Departamento de Ingeniería de Sistemas Industriales y Diseño

Departamento de Ingeniería Mecánica y Construcción

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE COMPARTICIÓN E INFERENCIA DE CONOCIMIENTO BASADO EN EL MARCO FBS A TRAVÉS DE LA INGENIERÍA ONTOLÓGICA

Programa de doctorado

Proyectos de innovación tecnológica en ingeniería del producto y del proceso

Memoria de tesis presentada por

David Cebrián Tarrasón

para optar al grado de doctor, dirigida por

Dra. Rosario Vidal Nadal

Castellón de la Plana (España). Diciembre 2013

Quiero, y así soy.

Me equivoco, para crecer.

Pertenezco a este instante

y moriré,

cuando ya no sienta.

Agradecimientos

Hay caminos que llenan los sueños de quienes los tienen. Esta es la historia de una persona que imaginó un día con poder conocer el mundo de la investigación y que sin tantas personas que se han cruzado en su camino no habría sido posible haber llegado hasta aquí.

Por dónde empezar es casi imposible, pues primero debería agregar a todas las personas que me he cruzado en este tiempo en cuatro lugares que me han llenado mucho: Instituto de Terapia Gestalt, Cátedra INCREA, AEIPRO y GID.

Del Instituto de Terapia Gestalt debería nombrar a todos los miembros de un grupo que paralelamente al trabajo realizado en esta tesis, me han permitido conocer un mundo maravilloso que ya me ha abierto muchas puertas y que me seguirá vislumbrando otras tantas :-).

De la Cátedra INCREA pues me permitió a la vez que iniciaba parte de esta investigación conocer el mundo de la innovación y principalmente el de la creatividad, que luego me ha brindado maravillosas experiencias tanto como docente como investigador. De esa etapa quiero nombrar a José Ruiz y a Jordi Olucha por todas las experiencias que compartí con ellos y que me enseñaron sobre el trabajo y la vida.

AEIPRO, ¡realmente quien me iba a decir que iba a llegar a Coordinador de la sección joven en aquel primer congreso internacional celebrado en Lugo y que luego me ha permitido conocer Turquía, Australia, Grecia, Holanda, Italia y Portugal! En este proceso nombrar al Dr. Salvador Capuz y fundamentalmente al Dr. Daniel Collado-Ruiz, con quien he vivido grandes aventuras ;-).

Y por último, cómo no, el GID, ese Grupo de Ingeniería de Diseño con ese proyecto del MEC que me ha proporcionado muchas alegrías y muchas penas por el esfuerzo realizado durante estos años, en el cual he aprendido tanto de valores personales como profesionales de grandes personas y actualmente ya doctores Dr. Daniel Justel, Dr. Carlos Muñoz, Dr. Daniel Garraín, Dr. Enrique Moliner y Vicente Franco (próximamente) y Dr. Vicent Chulvi como así Marta Royo. Y a la colaboración de la todos aquellos estudiantes que han ido pasando por el grupo y que puntualmente me han brindado su colaboración y sus enseñanzas como es el caso de Jessica Abad-Kelly (en el apartado inicial relativo al modelado de CAD del TULUM®).

Personalmente también quiero agradecer a José Sánchez Moreno, su inestimable colaboración en el apartado referido al KSS 2.0, y apoyo moral durante esos meses de trabajo que realizamos juntos.

Luego, no podría olvidar a toda mi familia por todo el tremendo apoyo en los momentos buenos y malos: a mis padres Manuel y Montse, a mi tía Rosalía, a mi prima Elisa, a mis abuelos Manuel y María, Joaquín y Felicitas y a toda aquella parte cercana que ha estado curiosa preguntando cómo me iba yendo ☺. Y a todos mis amigos (no hacen falta nombres ni me cabrían aquí ;-) que me apoyan incondicionalmente en mi día a día.

Cómo no, a mi directora de tesis, Sari, que después de este largo proceso, acompañado de ideas y venidas, pacientemente haya aguardado a que este proyecto estuviera gratinado y en su punto.

Y finalmente agradecer a todos los que estáis leyendo esto, que significará que por algún motivo, grande o pequeño os interesa este trabajo. Y eso, al fin y al cabo, es lo que importa, que el tiempo que he ocupado sirva para que otras personas como tú puedan aprovecharse y mejorar la labor realizada por el bien de la ciencia y la investigación.

Índice

Índice de tablas

Índice de figuras

Índice de reglas

Índice de gráficos

Abreviaturas

ACV	Análisis del Ciclo de Vida
AI	Artificial Intelligence
AJAX	Asynchronous Javascript XML
API	Application Program Interface
APO	Agentive Physical Object
A-QB	Action –QB[a] (*Cube*)
AsD	Assembly Design
B-Cube	Behavior-Cube
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAs	Consumer Attributes
CAx	Computer Aided
CommonKADS	Common Knowledge Acquisition Documentation System and KBS Analysis and Design Support
CSS	Cascading Style Sheets
DAML	DARPA Agent Markup Language
DiDeas	Distributed Design Assistant
DfX	Design for
DO	Design Ontology
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DPs	Design Parameters
EDIT	Engineering Design Integrated Taxonomy
EIS	Enterprise Information System
FaBES	Function-action-Behavior Environment Structure
FAST	Function Analysis System Technique
FEBS	Function-Environment-Behavior-Structure modeling
FBRL	Function and Behavior Representation Language
FB	Functional Basis
FBS	Function Behavior Structure
FCBS	Function-Cell-Behaviour-Structure
FKC	Functional Knowledge Cell
FPPT	Function, Physical Principle and Technology
FR	Formal theory of objects and object functionalities
FRs	Functional Requirements
GCI	General Concept Inclusions
GUI	Graphical User Interface
HQL	Hibernate Query Language

[a] Pronunciado “cu-be”, es decir, cubo, del inglés *Cube*

HSA	Heuristic State-Space Approach
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICARE	Illustration, Constraint, Activity, Rule, Entity
JDBC	Java Database Connectivity
Java SE	Java Platform, Standard Edition
Java EE	Java Platform, Enterprise Edition
Java ME	Java Platform, Micro Edition
JRE	Java Runtime Environment
KBE	Knowledge Based Engineering
KBS	Knowledge Based System
KSS	Knowledge Sharing System
MCRDR	Multiple Classification Ripple Down Rules
MEC	Ministerio de Educación y Ciencia
MOKA	Methodology and tools Oriented to Knowledge Acquisition
NAPO	Non-Agentive Physical Object
OIL	Ontology Inference Layer
OntoFaBES	Ontología basada en el marco FaBES
OntoRFB	Ontological Reconciled Functional Basis
ORM	Object/Relational Mapping
OWA	Open World Assumption
OWL	Web Ontology Language
PDA	Personal Data Assistant
PDF	Personal Document File
PLM	Product Lifecycle Management
PVs	Process Variables
PYME	Pequeña y Mediana Empresa
QFD	Quality Function Deployment
R.A.E.	Real Academia Española
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RDFS	RDF Schema
RFB	Reconciled Functional Basis
RIA	Rich Internet Application
RML	Rule Markup Language
RPC	Remote Procedure Call
RSS	Really Simple Syndication
SaaS	Software as a Service
SoA	State of Affairs
SOAP	Simple Object Access Protocol

SQL	Structured Query Language
SQWRL	Semantic Query-Enhanced Web Rule Language
STEP	Standard for the Exchange of Product model data
SWRL	Semantic Web Rule Language
TIC	Tecnologías de la Información y Comunicación
UDDI	Universal Description, Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Markup Language

Resumen

En un mundo donde la economía está globalizada y la información tiende a desarrollarse de diversas formas y en base a comunidades, la gestión del conocimiento es un tema recurrente. Además, en los últimos tiempos el conocimiento intangible ha pasado a considerarse un recurso de vital importancia para las grandes empresas. Bajo esta perspectiva, el conocimiento es un valioso elemento necesario para mantenerse competitivo. Es por ello que, de acuerdo a numerosos trabajos de investigación y experiencias industriales, la efectiva gestión del conocimiento representa un importante factor de competencia al mismo nivel que el entorno de la web 2.0 y 3.0 (o también llamada *web semántica*).

En el campo del diseño, los diseñadores dedican aproximadamente un 25% de su tiempo en la adquisición de información y conocimiento, tiempo excesivo que sugiere la necesidad de investigar en nuevas estrategias para gestionar el conocimiento del diseño. Además, el enfoque perseguido en esta búsqueda de nuevas estrategias debe estar guiado por la facilidad de interacción con el usuario y por su integración en la gestión del conocimiento global de la empresa.

Desde esta perspectiva, los sistemas expertos o sistemas KBS *(Knowledge Based Systems – Sistemas basados en el conocimiento)* y específicamente los sistemas KBE *(Knowledge Based Engineering – Ingeniería basada en el conocimiento)* pueden verse como herramientas que facilitan los procesos de diseño y desarrollo de productos en entornos participativos y de colaboración en el campo de la ingeniería y han resuelto en gran medida dichas necesidades. Sin embargo aún no han alcanzado un estado óptimo de desarrollo y accesibilidad para ser aplicados e implementados en el diseño y desarrollo de nuevos productos en las pequeñas y medianas empresas, PYMEs. Por tal motivo, se hace necesaria mayor investigación que permita su desarrollo, aplicación e implementación.

Los actuales sistemas KBE no han conseguido implantarse en las pequeñas y medianas empresas por la extremada complejidad del software, por el desproporcionado nivel de abstracción y conocimientos informáticos que requieren los diseñadores y por un coste excesivo. Otro inconveniente de estos programas de KBE se debe a que aunque utilizan el conocimiento, realmente no lo gestionan y exigen de un enorme esfuerzo para capturarlo. Sin embargo, al no estar convenientemente gestionado, este esfuerzo resulta poco útil y además no permite su integración con modelos de síntesis para diseñar nuevos productos ni tampoco asistir al diseñador durante las fases iniciales del diseño.

Posteriormente una evolución de los sistemas KBE fue desarrollada por la ingeniería ontológica en el caso de tratamiento de amplias bases de datos de información y se generaron algunos modelos funcionales también con una gran complejidad y coste, en el que no se trataba la posibilidad de inferir conocimiento a partir de la información recopilada. Además que en el ámbito del diseño funcional, las ontologías existentes eran complejas y de difícil acceso para la reutilización de conocimiento.

Esta tesis aporta KSS (*Knowledge Sharing System – Sistema de compartición del conocimiento*) 2.0 un sistema de reutilización, gestión e inferencia del conocimiento. Evolución del modelo KSS, integra el modelo MOKA de metodología de gestión del conocimiento para sistemas KBE con los avances más recientes en el ámbito de la ingeniería ontológica. Para lo cual, hace uso de las nuevas tecnologías web disponibles en torno al concepto comúnmente conocido como Web 2.0.

Esta plataforma busca repercutir directamente en la competitividad de las PYMEs, mejorando su rendimiento en el cumplimiento de los requisitos del cliente, a la vez que acortando el ciclo de diseño de productos (en su fase de re-diseño) y el tiempo de puesta en el mercado para poder reducir costes.

El sistema KSS 2.0 explicita el conocimiento del diseño de productos a través del nuevo marco de diseño funcional FaBES (*Function – action – Behavior – Environment – Structure | función – acción – comportamiento – entorno – estructura*). Para ello se plantean las definiciones de cada una de las capas del citado marco, acompañadas de varios ejemplos que aclaran su organización y labor.

Se adapta dicho modelo al ámbito de la ingeniería ontológica a través de la ontología OntoFaBES, ontología que formaliza el conocimiento sobre un producto con el objetivo de inferir diferentes estructuras de dicho

producto a partir de los requerimientos funcionales propuestos por el usuario, con el objetivo de que el conocimiento creado se pueda reutilizar y compartir entre diferentes aplicaciones.

Para evaluar la consistencia de KSS 2.0 y por tanto de OntoFaBES se considera el ejemplo real de una cámara submarina que recibe el nombre comercial de TULUM®, instrumento que sirve para inspeccionar el casco de veleros sin necesidad de sacarlo del agua. Para poder entender la aplicación informática, primero se determina el análisis de funciones y comportamientos haciendo uso del marco FaBES. Posteriormente, se describe el proceso de obtención de información del diseño del TULUM utilizando el programa de CAD (*Computer Aided Design - diseño asistido por ordenador*) *Solidworks*®, su posterior formalización a través de la macro KSS-*Solidworks*, edición en la aplicación web KSS 2.0 y conversión a la ontología OntoFaBES.

Finalmente, se muestran los resultados obtenidos al aplicar las reglas de inferencia definidas en OntoFaBES.

Resum

En un món on l'economia està globalitzada i la informació tendeix a desenvolupar-se de forma diverses i sobre la base de comunitats, la gestió del coneixement és un tema recurrent. A més, sent que en els últims temps el coneixement intangible hagi passat a considerar-se un recurs de vital importància per a les grans empreses. Sota aquesta perspectiva, el coneixement és un valuós element necessari per mantenir-se competitiu. És per això que, d'acord amb nombrosos treballs d'investigació i experiències industrials, l'efectiva gestió del coneixement representa un important factor de competència al mateix nivell que l'entorn de la web 2.0 i 3.0 (també coneguda como *web semàntica*).

Al camp del disseny, els dissenyadors dediquen aproximadament un 25% del seu temps en la consecució d'informació i coneixement, temps excessiu que suggereix la necessitat d'investigar en noves estratègies per gestionar el coneixement del disseny. A més, l'enfocament perseguit en aquesta recerca de noves estratègies ha d'estar guiat per la facilitat d'interacció amb l'usuari i per la seva integració en la gestió del coneixement global de l'empresa.

Des d'aquesta perspectiva, els sistemes experts o sistemes KBS (*Knowledge Based Systems*) i específicament els sistemes KBE (*Knowledge Based Engineering*) es poden veure com a eines que faciliten els processos de disseny i desenvolupament de productes en entorns participatius i de col·laboració al camp de la enginyeria i que han resolt en gran mesura aquestes necessitats. No obstant això encara no han arribat a un estat òptim de desenvolupament i accessibilitat per ser aplicats i implementats en el disseny i desenvolupament de nous productes en les petites i mitjanes empreses, PIMEs. Per tal motiu, es fa necessària més investigació que permeti el seu desenvolupament, aplicació i implementació.

Els actuals sistemes KBE no han aconseguit implantar-se a les petites i mitjanes empreses per l'extremada complexitat del programari, pel desproporcionat nivell d'abstracció i coneixements informàtics que requereixen els dissenyadors i per un cost excessiu. Un altre inconvenient d'aquests programes de KBE és que encara que utilitzen el coneixement, realment no ho gestionen i exigeixen d'un enorme esforç per capturar el coneixement. No obstant, en no estar convenientment gestionat, aquest esforç és poc útil i a més no permeten la seva integració amb models de síntesi per dissenyar nous productes ni tampoc assistir al dissenyador durant les fases inicials del disseny.

Posteriorment una evolució dels sistemes KBE va ser desenvolupada per l'enginyeria ontològica al cas de tractament d'àmplies bases de dades d'informació i es van generar alguns models funcionals també amb una gran complexitat i cost, al qual no es tractava la possibilitat d'inferir coneixement a partir de la informació recopilada. A més que en l'àmbit del disseny funcional, les ontologies existents eren complexes i de difícil accés per a la reutilització de coneixement.

Aquesta tesi aporta KSS (*Knowledge Sharing System)* 2.0 un sistema de reutilització, gestió i inferència del coneixement. Evolució del model KSS, integra el model MOKA de metodologia de gestió del coneixement per sistemes KBE amb els avanços més recents en l'àmbit de l'enginyeria ontològica. Per tal motiu, fa ús de les noves tecnologies web disponibles al voltant del concepte popularment conegut com Web 2.0.

Aquesta plataforma busca repercutir directament a la competitivitat de les pimes, millorant el seu rendiment en el compliment dels requisits del client, alhora que escurçant el cicle de disseny de productes (en la seva fase de re-disseny) i el temps de posada en el mercat per poder reduir costos.

El sistema KSS 2.0 explicita el coneixement del disseny de productes a través del nou marc de disseny funcional FaBES (*Function-action-Behavior-Environment-Structure)*. Per això es plantegen les definicions de cadascuna de les capes de l'esmentat marc, acompanyades de diversos exemples que aclareixen la seva organització i tasca.

S'adapta el ja anomenat model a l'àmbit de l'enginyeria ontològica a través de l'ontologia OntoFaBES, ontologia que formalitza el coneixement sobre un producte amb l'objectiu d'inferir diferents estructures d'aquest producte a partir dels requeriments funcionals proposats per l'usuari, amb l'objectiu de que el coneixement creat es pugui reutilitzar i compartir entre diferents aplicacions.

Per avaluar la consistència de KSS 2.0 i per tant de OntoFaBES es considera l'exemple real d'una càmera submarina que rep el nom comercial de TULUM ®, instrument que serveix per inspeccionar el casc de velers sense necessitat de treure'l de l'aigua. Per poder entendre l'aplicació informàtica primer es determina l'anàlisi de funcions i comportaments fent ús del marc FaBES. Posteriorment, es descriu el procés d'obtenció d'informació del disseny del TULUM utilitzant el programa de CAD *Solidworks*®, la seva posterior formalització a través de la macro KSS-*Solidworks*, edició a l'aplicació web KSS 2.0 i conversió a l'ontologia OntoFaBES.

Finalment, es mostren els resultats obtinguts en aplicar les regles d'inferència definides en OntoFaBES.

Abstract

In a world where the economy is globalized and the information tends to develop communities in different ways based on communities, knowledge management is a recurring topic. Moreover, considering that in recent time, intangible knowledge has come to be considered a critical resource for large companies. In this perspective, knowledge is a valuable element necessary to stay competitive. This is why, according to many researches and industrial experiences, effective knowledge management is an important factor of competition at the same level as the environment of Web 2.0 and 3.0 (also known as *semantic web*).

In the field of design, designers spend approximately 25% of their time in the acquisition of information and knowledge. This is an excessive time that suggests the need for researching on new strategies in order to manage design knowledge. Moreover, the approach pursued in this search for new strategies should be guided by the ease of user interaction and integration into the global knowledge management company.

From this perspective, expert systems or KBS (*Knowledge Based Systems*) and specifically KBE (*Knowledge Based Engineering*) can be seen as tools that facilitate the processes of design and product development in participatory and collaborative environments in the field of engineering and have largely solved these needs. However, they have not yet reached an optimum stage of development and accessibility to be applied and implemented in the design and development of new products in small and medium enterprises, SMEs. For that reason, more research is needed to enable its development, application and implementation.

KBE current systems have failed to be implemented in SMEs by the extreme complexity of the software, a disproportional level of abstraction and computer skills needed by designers and by excessive costs. Another disadvantage of these KBE programs is that are not really managing knowledge even they use it and they require a huge effort to capture knowledge. However, as cited knowledge is not being properly managed, this effort is of little use and it does not allow integration with synthesis models to design new products nor assist the designer during the early stages of design.

Afterwards an evolution of KBE systems was developed by ontological engineering for treatment of large information databases and several functional models were also generated with a great complexity and cost, in which was not possible to infer knowledge from the information collected. Also in the field of functional design, existing ontologies were complex and difficult to reuse knowledge.

This thesis provides KSS (Knowledge Sharing System) 2.0 as a reuse, management and inference knowledge system. Evolution of KSS model, integrates MOKA model of knowledge management methodology for KBE systems with the latest developments in the field of ontology engineering. For that aim, uses available new web technologies around the concept commonly known as Web 2.0.

This platform aims to directly impact the SMEs competitiveness, improving its performance in meeting customer requirements, while shortening the product design cycle (at the stage of re-design) and the time-to-market in order to reduce costs.

The KSS 2.0 system makes explicit the product design knowledge through FaBES (Function-action-Behavior-Environment-Structure) new functional design framework. To do this the definitions of each of the layers of said framework are considered, complemented by a number of examples which clarify its organization and work.

The model is adjusted to the field of ontology engineering through OntoFaBES ontology, formalizing knowledge of a product in order to infer different structures of that product from functional requirements given by the user, with the aim of that the created knowledge could be reused and shared between different applications.

To evaluate the consistency of KSS 2.0 and thus OntoFaBES is considered the actual example of an underwater camera that receives TULUM ® trade name, instrument to inspect the sailboat hull without removing it from the water. In order to understand the computer application, first a function and behavior analysis is determined using the FaBES framework. Subsequently, the process of obtaining information

from TULUM design is described using *Solidworks*® CAD (*Computer Aided Design*) program, later formalized through KSS-*Solidworks* macro, editing in KSS 2.0 web application and conversion to OntoFaBES ontology.

Finally, the results of applying inference rules defined in OntoFaBES are shown.

Capítulo 1. Introducción

1.1 Objeto y alcance

El objeto de estudio de esta tesis se centra en el campo de la gestión del conocimiento aplicada al ámbito del diseño funcional a través de la ingeniería ontológica.

Esta tesis doctoral se enmarca dentro del proyecto ya concluido "Sistema integrado de eco-innovación de producto en PYMEs basado en el conocimiento" con referencias DPI2006-15570-C02-01 y DPI2006-15570-C0-02. El proyecto abordó el diseño y desarrollo de un sistema de gestión del ciclo de vida del producto basado en el conocimiento, enfocado a la optimización y automatización de las decisiones en las fases de diseño de producto en PYMEs (Pequeñas Y Medianas Empresas) por medio de técnicas y herramientas basadas en el conocimiento.

La arquitectura estuvo basada en el marco FBS (Función Comportamiento Estructura, del inglés *Function-Behavior-Structure*) y en el uso de ontologías con el objetivo de que el conocimiento pudiera ser reutilizado y compartido por diferentes aplicaciones.

El modelo de síntesis funcional dispuso de un módulo basado en conocimiento, KBE, para el enlace de aplicaciones de diseño. Esta plataforma buscaba repercutir directamente en la competitividad de las PYMEs, mejorando la calidad de las mismas en cumplimiento de los requisitos del cliente, a la vez que acortando el ciclo de diseño de productos y reduciendo costes en el proceso de rediseño.

1.2 Justificación

El término "**gestión del conocimiento**" ha cobrado especial relevancia en los últimos tiempos debido en gran parte al impacto que está causando el hecho de que, a pesar de que el conocimiento es intangible, haya pasado a considerarse un recurso de vital importancia en todo tipo de organizaciones. Bajo esta perspectiva, el conocimiento es un valioso ingrediente necesario para alcanzar objetivos como la ventaja competitiva e incluso la supervivencia de organizaciones. Es por ello que, de acuerdo con numerosos trabajos de investigación y experiencias industriales, la efectiva gestión del conocimiento representa un importante factor de competitividad de la misma manera que lo es la gestión de la información.

En el campo del diseño, los diseñadores dedican aproximadamente un 25% de su tiempo en la consecución de información y conocimiento, tiempo excesivo que sugiere la necesidad de investigar en nuevas estrategias para gestionar el conocimiento en el diseño. Además, el enfoque perseguido en esta búsqueda de nuevas estrategias debe estar guiado por la facilidad de interacción con el usuario y por su integración en la gestión del conocimiento global de la empresa.

Desde esta perspectiva, los sistemas expertos o sistemas KBS (Sistemas Basados en el Conocimiento, del inglés *Knowledge Based Systems*) y específicamente los sistemas KBE (Ingeniería Basada en el Conocimiento, del inglés *Knowledge Based Engineering*) pueden verse como herramientas que facilitan los procesos de diseño y desarrollo de productos en entornos participativos y de colaboración en el campo de la ingeniería y han resuelto en gran medida dichas necesidades. Sin embargo, aún no han alcanzado un estado óptimo de desarrollo y accesibilidad para ser aplicados e implementados en el diseño y desarrollo de nuevos productos en las PYMEs, por tal motivo se requiere más investigación que permita su desarrollo, aplicación e implementación.

Los actuales sistemas KBE como ICAD® o los incluidos en programas de CAD (Diseño Asistido por Ordenador, del inglés *Computer Aided Design*) como CATIA® no han conseguido implantarse en las pequeñas y medianas empresas por la excesiva complejidad del software, por el excesivo nivel de abstracción y conocimientos informáticos que requieren los diseñadores y por un coste excesivo. Otro inconveniente de estos programas de KBE se debe a que aunque utilizan el conocimiento, realmente no lo gestionan. Exigen un enorme esfuerzo para capturar el conocimiento, pero al no estar convenientemente gestionado, este esfuerzo resulta poco útil. Además no permiten su integración con modelos de síntesis para diseñar nuevos productos, ni tampoco asistir al diseñador durante las fases iniciales del diseño.

1.3 Objetivos

La tesis doctoral se incluye dentro de un proyecto que abordó el diseño y desarrollo de un sistema de gestión del ciclo de vida del producto basado en el conocimiento, enfocado a la optimización y automatización de las decisiones en las fases de diseño de producto en PYMEs por medio de técnicas y herramientas basadas en el conocimiento.

El núcleo central de la tesis es un sistema de compartición e inferencia de conocimiento que opera sobre un nuevo marco de diseño funcional operado desde una ontología, que permite además de recopilar el conocimiento disponible, asistir en el rediseño de productos ya existentes. Los objetivos concretos son:

1. La mejora, desarrollo e implementación de una ontología para el diseño de objetos basada en el marco de diseño funcional FaBES.
2. Desarrollar e implementar un sistema de compartición e inferencia de conocimiento basado en el marco FaBES a través de la ingeniería ontológica.
3. Aplicar el sistema de compartición e inferencia de conocimiento a un caso práctico a nivel industrial. Concretamente se aplica al caso de una cámara submarina.

1.4 Hipótesis

Esta tesis pretende corroborar las siguientes hipótesis de partida:

- El marco FaBES es compatible con las reglas lógicas de una ontología siendo una especificación formal y explícita de una conceptualización compartida sobre el ámbito del diseño conceptual.
- Una ontología desarrollada en lenguaje OWL y SWRL puede explicitar e inferir conocimiento del diseño conceptual de un producto a partir del marco FBS y la metodología MOKA.
- Se puede disponer de un sistema online y semi-automático de compartición e inferencia de conocimiento del diseño CAD de un producto que utilice la metodología MOKA e ingeniería ontológica para optimizar el rediseño de productos.

1.5 Metodología

De cara a comprobar la hipótesis de partida y lograr el cumplimiento de los objetivos planteados, la metodología empleada en esta investigación ha seguido las etapas que se describen a continuación.

En primer lugar, se revisa la literatura sobre las metodologías del diseño funcional y su aplicación al ámbito de los KBE y de la ingeniería ontológica. De este análisis se observan las bondades y los procedimientos válidos y aplicables para el desarrollo de una nueva metodología.

Posteriormente se desarrolla un nuevo marco de diseño funcional para poder ser aplicado a la ingeniería ontológica, mostrando varios ejemplos de trabajo.

A continuación, se explica un nuevo modelo de compartición del conocimiento basado en la estructura de la web 2.0.

Finalmente, se plantea el caso de estudio aplicado a una cámara submarina.

1.6 Estructura del documento

El presente trabajo está dividido en seis capítulos, contando este primero como introductorio.

En el capítulo 2 se se describe el estado del arte sobre el diseño funcional, el concepto de ontologías y su aplicación en el campo de la ingeniería del diseño. Primero se introducen los conceptos de función, comportamiento y estructura, elementos claves para la comprensión adecuada del diseño funcional. A continuación, se establece una revisión de las investigaciones más importantes en relación con el marco FBS, metodología para el análisis del proceso de diseño. Finalmente se trata el apartado referido al ámbito de la ingeniería basada en el conocimiento como elemento introductor al campo de las ontologías.

En los capítulos 3 y 4 se muestran los aportes principales de la tesis: el marco de diseño funcional FaBES, la ontología OntoFaBES y el KSS 2.0 (Sistema de compartición del conocimiento 2.0, del inglés *Knowledge Sharing System* 2.0) (Figura 1).

En el capítulo 3 se aporta FaBES, marco de diseño funcional, y OntoFaBES, ontología que basada en FaBES, formaliza el conocimiento sobre un producto para inferir diferentes estructuras de dicho producto a partir de los requerimientos funcionales propuestos por el usuario, con el objetivo de que el conocimiento creado se pueda reutilizar y compartir entre diferentes aplicaciones.

El KSS 2.0 se presenta en el capítulo 4. Se trata de una evolución del modelo KSS (Sánchez-Moreno 07), que integra el modelo MOKA (Metodología y herramientas orientadas a la adquisición del conocimiento, del inglés *Methodology and tools Oriented to Knowledge Acquisition*) (Stokes, 2001) de metodología de gestión del conocimiento para sistemas KBE con los avances más recientes en el ámbito de la ingeniería ontológica. En dicho capítulo, se detalla la estructura de KSS 2.0 para después indicar su arquitectura.

Una vez planteada la ontología OntoFaBES e introducido el KSS 2.0, en el capítulo 5 se evalúa su consistencia. Para ello, se considera el ejemplo real de una cámara submarina que recibe el nombre comercial de TULUM®, instrumento que sirve para inspeccionar el casco de veleros sin necesidad de sacarlo del agua.

Finalmente, en el capítulo 6, en primer lugar se estructuran y presentan las conclusiones del trabajo realizado a partir de los resultados y las discusiones, justificando el cumplimiento de las hipótesis y los objetivos. Posteriormente se realiza un análisis de las perspectivas y futuros trabajos que pueden desarrollarse a partir de este trabajo. También se presenta la difusión que se ha dado hasta el momento del trabajo realizado mediante publicaciones, ya sea en revistas, capítulos de libro, congresos nacionales e internacionales y documentos científico-técnicos.

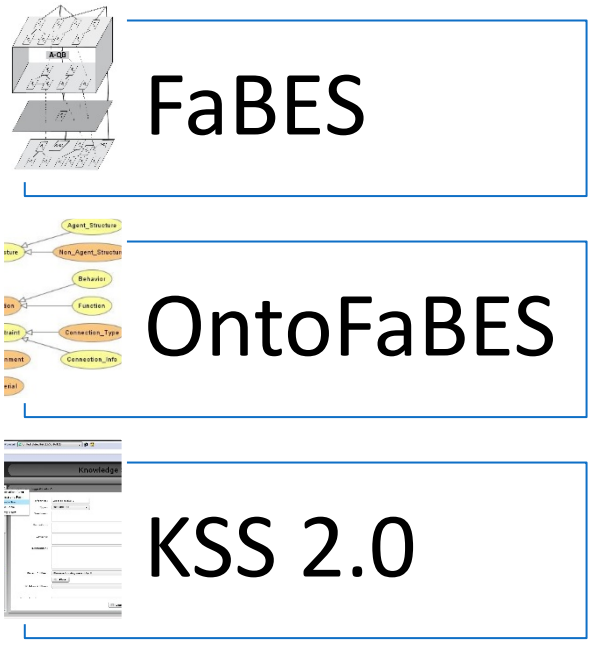


Figura 1: Aportes principales de la tesis

Capítulo 2. Antecedentes

En el presente capítulo se describe el estado del arte en los campos del diseño funcional, ingeniería basada en el conocimiento, el concepto de ontologías y su aplicación en el campo de la ingeniería del diseño para una comprensión adecuada de los aportes realizados en esta tesis.

Primero, se introducen los conceptos de función, comportamiento y estructura, elementos claves para la comprensión adecuada del diseño funcional. Eso permite establecer una revisión de las investigaciones más importantes en relación al marco FBS, modelo del conocimiento requerido durante la fase de diseño conceptual. Además se presenta el modelo B-Cube (Chulvi, 2013) identificando una serie de deficiencias en éste. Todo ello va a permitir comprender la necesidad de generar un marco nuevo de formalización del diseño conceptual.

A continuación, se introducen una serie de conceptos básicos referidos al ámbito de la arquitectura cliente/servidor dado que se va a tratar posteriormente de un sistema online para la gestión del conocimiento del diseño de productos y se ha considerado conveniente explicar el funcionamiento informático de la aplicación informática que se aporta en esta tesis. Por tal motivo, se introduce el significado de los términos clave referidos a los paquetes de programación, aplicaciones web, servidor y arquitectura.

Posteriormente, se trata el apartado referido al ámbito de la ingeniería basada del conocimiento (KBE) como elemento introductor al campo de las ontologías. El objetivo es poder conocer la base teórica que sustentan los sistemas KBE de la misma forma como la necesidad de utilizar una ontología como mejora de dichos sistemas pues la aplicación informática que se aporta está conformada por un KBE y una ontología. Para ello, se va a explicar desde la importancia del conocimiento en la fase de diseño conceptual de productos, seguido por la definición de la KBE, MOKA, metodología utilizada para la adquisición del conocimiento del diseño de productos así como el KSS (Moreno, 07) identificando aquellos aspectos con margen de mejora. Y para acabar el apartado, se explica el contexto actual en el cual se encuentra la gestión del conocimiento ante el auge de Internet y cómo eso lleva a la evolución del KBE a la ingeniería ontológica.

Finalmente, se explica en qué consiste una ontología, las características que la definen y se profundiza en aquellas que tratan el ámbito de la ingeniería de diseño, concretamente en DOLCE (Ontología descriptiva para la ingeniería cognitiva y lingüística, en inglés *Descriptive Ontology for Linguistic and Cognitive Engineering*) (Masolo, 2003) y OntoRFB (Base funcional reconciliada ontológicamente, en inglés *Ontological Reconciled Functional Basis*) (Garbacz, 2006), de las cuales parte el desarrollo teórico de OntoFaBES, la ontología que se presenta en esta tesis doctoral.

2.1 Diseño funcional

Esta tesis aporta un nuevo marco de diseño funcional (Capítulo 3.1). Para poder comprender la necesidad de su creación, se establece una revisión respecto a los términos función, comportamiento y estructura. Luego, en base a dichos términos se explica con detalle el estado del arte en el ámbito del marco funcional FBS. Y se para finalizar se analiza el modelo B-Cube como introducción al modelo de acciones que se presenta en los apartados 3.1.3 y 3.1.4.

2.1.1 Función, Comportamiento y Estructura

Una función es un elemento que dispone de diferentes interpretaciones dependiendo del enfoque utilizado (Borgo, 2009; Srinivasan, 2012). Según la R.A.E. (Real Academia Española) la palabra función dispone de 14 acepciones diferentes. Para el propósito que atañe a este estudio, la más próxima por sentido es la primera acepción:

1. f. Capacidad de actuar propia de los seres vivos y de sus órganos, y de las máquinas o instrumentos.

A nivel general, una función se puede considerar como la relación de entrada-salida prevista de un sistema cuyo propósito es realizar una tarea (Pahl & Beitz, 1984). Por ello, el análisis funcional tiene por objeto

traducir la estructura de un producto (piezas, procesos, o materiales, entre otros) en una estructura de palabras. De esta forma se construye un diagrama funcional. Cada elemento se representa por una combinación de palabras verbo-nombre. Por tanto, el hecho de analizar las funciones de un producto es totalmente equivalente a analizar el producto.

De esa forma, haciendo uso del FAST (Técnica de sistema de análisis de funciones, del inglés *Function Analysis System Technique,* Bytheway, 1971) las funciones obtenidas para un objeto se pueden estructurar en función objetivo, función básica y cuatro tipos de funciones de apoyo: asegurar fiabilidad, aumentar el valor, asegurar comodidad de uso y complacer los sentidos. Sin embargo, este método aunque es útil sólo permite al diseñador expresar una solución, con lo que se impediría el mantener todas las soluciones potenciales propuestas en cada estadio sin llegar a estar definidas (Costadoat, 2010).

En el campo de la ingeniería de diseño, se plantean una gran diversidad de definiciones tanto en el apartado de función como en el de comportamiento debido a la recopilación de diferentes modelos y teorías provenientes de la experiencia práctica (Garbacz, 2005). Y tal como indica Chandrasekaran (2005), esta variedad de enfoques alimenta un cierto desorden. Así mismo Crilly (2013) recalca la importancia de relativizar los tipos de funciones en los proyectos de diseño complejos sugiriendo una revisión profunda en su planteamiento.

De la misma forma, Vermaas (2012) discute que en los métodos de diseño en que los flujos de energía, materiales y señales se consideran como funciones, éstas no pueden establecer una relación formal de pertenencia con sus subfunciones en la descripción funcional de artefactos al no mantenerse la anti-simetría, es decir, haciendo uso de una lógica formal se puede demostrar que una función tiene subfunciones pero no lo contrario. Por ello, la utilización de un marco común, como es el marco FBS basado en criterios de lógica formal, permite establecer un cierto orden en la citada cuestión.

En ingeniería, el comportamiento es el eje sobre el que se estructura una serie de metodologías de diseño permitiendo conectar las descripciones de la estructura física de objetos a las descripciones de sus funciones técnicas.

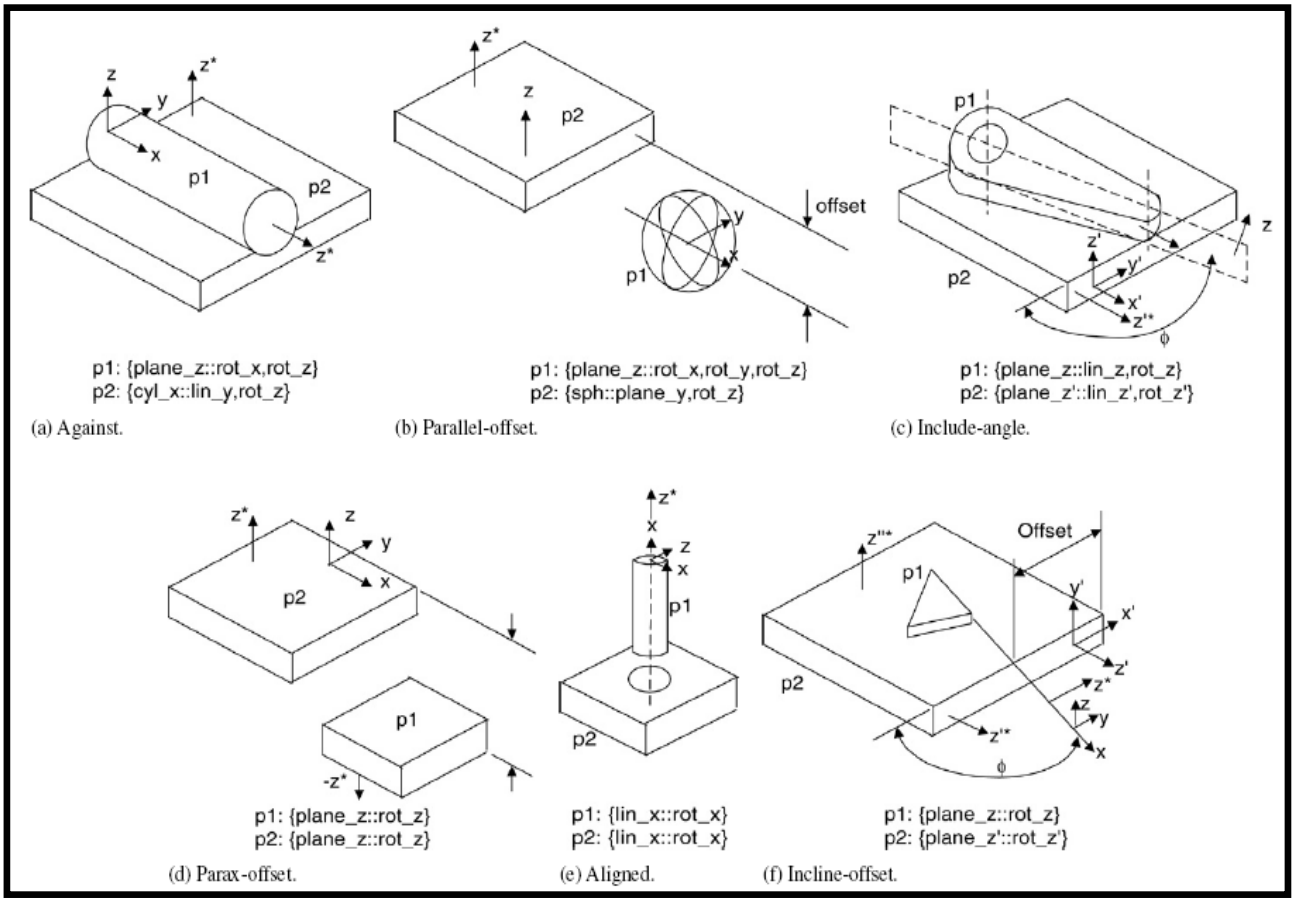


Figura 2: Seis tipos de relaciones espaciales (Ambler & Popplestone, 1975).

Desde una concepción lógica, Borgo et al. (2006) captura y relaciona las diferentes posturas y establece el comportamiento como una relación entre el objeto y el evento en el cual participa. Sin embargo, a veces se establece una cierta confusión para comprender las diferencias entre funciones como comportamientos o como unidades físicas en un producto (Andreasen 2011).

En una estructura, un elemento se considera como cada una de las partes diferenciadas, aunque vinculadas, en que puede ser dividida a efectos de su diseño. Dichos elementos en un conjunto pueden ser divididos en partes o combinarse para formar ensamblajes. Al respecto, hay una gran unicidad en su representación espacial. Ambler & Popplestone (1975) y Rihtarsic (2008) estudian el tipo de conexión que existe entre dos partes, dividiéndose entre *Against, Parallel-offset, Include-angle, Parax-offset, Aligned* e *Incline-offset* (Figura 2).

Finalmente, el concepto de *affordance* (permiso[b]) ha cobrado especial interés en el diseño funcional porque captura bastante bien la relación entre los usuarios humanos y los artefactos diseñados. *Affordances* en diseño son las posibilidades de acción de un usuario cuando éste interactúa con un artefacto (Gibson, 1977; Maier 2009a; Maier 2009b).

2.1.2 Marco FBS

Aunque los términos *función*, *comportamiento* y *estructura* ya habían sido utilizados en el campo del diseño industrial, no es hasta 1990 cuando se clarifican y se usan para definir un marco donde modelar y representar la funcionalidad de un sistema: el marco FBS (Gero, 1990; Umeda, 1990). En dicho marco, *función* representa las tareas que el sistema lleva a cabo, *estructura* representa los elementos físicos de la solución y *comportamiento* actúa como la relación entre *función* y *estructura*. En la síntesis de diseño, el *comportamiento* se deriva de una funcionalidad intencionada con el fin de obtener una solución de ello. Cuando se define una solución, su *comportamiento* se deduce de dicha solución para evaluar si ésta alcanza la funcionalidad pretendida. Entonces, el *comportamiento* se relaciona con el estado físico del diseño, el cual puede ser estático o variable con el tiempo.

A partir de la revisión del estado del arte, en la Tabla 1 se conforma un listado de las investigaciones realizadas sobre el diseño funcional (Garbacz, 2005; Camelo, 2007; Erden, 2008).

Tabla 1: Investigaciones relacionadas con el marco FBS.

Abreviatura	Fuente	Descripción
A-Design	Campbell et al. (2003)	Agent-Based Approach to Conceptual Design in a Dynamic Environment
Axiomatic Design	Suh (1990)	Axiomatic theory of design
B-Cube	Chulvi et Vidal (2013)	Behavior Cube
B-FES	Tor et al. (2000, 2002)	Behavior- Function Environment Structure
FABM	Li et al. (2012)	Function Action Behavior Mechanism
FBOS	Montecchi et al. (2011)	Function/Behavior-Oriented-Search
FCBS	Chao-Chen et al. (2012)	Function–Cell–Behavior–Structure model
FBRL	Sasajima et al. (1995)	Function and Behavior Representation Language
FBS	Umeda et al. (1990)	Theory of objects described in terms of Function, Behavior and Structure

[b] Considerando permiso como la capacidad de que un instrumento pueda permitir realizar una acción a su usuario.

Abreviatura	Fuente	Descripción
FBS	Gero et al. (1990)	Theory of objects described in terms of Function, Behavior and Structure
FEBS	Deng et al. (1999)	Function-Environment-Behavior-Structure modeling
FPPT	Klein Meyer et al. (2007)	Function, Physical Principle and Technology
FR	Chandrasekaran et al. (2005)	Formal theory of objects and object functionalities
Reconciled Functional Basis	Hirtz et al. (2002)	Taxonomy of flows and functions
HSA	Zhang et al. (2002)	Heuristic State-Space Approach
NIST Design Repository	Szykman et al. (1999)	Formal model of objects from object-oriented perspective
FB	Pahl and Beitz (1984)	General theory of design activity
PFM	Roy et al. (2002)	General theory of objects
RFB	Hirtz et al. (2002)	Reconciled Functional Basis
RFBS	Christophe et al. (2010)	Requirement-Function-Behavior-Structure model
Theory of Technical Systems	Hubka and Eder (1988)	General non-formal theory of objects
VFS	Kim et al. (2009)	Value Function Structure

A grandes rasgos se podrían distinguir dos enfoques en el esquema FBS (Chandrasekaran et al., 2005). En el primero, se relacionan las *funciones* con los *comportamientos* de un elemento, y entonces se relacionan los citados *comportamientos* con las descripciones físico-estructurales de los elementos. Este enfoque fue desarrollado por Gero (1990), quien propuso el modelo de diseño *Function-Behavior-Structure (Función - Comportamiento - Estructura)*, y por Umeda et al. (1990), que propusieron el modelo de diseño *Function-Behavior-State (Función - Comportamiento - Estado)*.

Este primer enfoque considera el *comportamiento* como un concepto clave al sugerir una ordenación ontológica clara: los objetos tienen su propia estructura física. Esta estructura, en interacción con un entorno físico, suscita los *comportamientos* de los objetos, y entonces, dichos *comportamientos* determinan de alguna manera las *funciones* de los objetos (Borgo, 2006).

Diversas investigaciones se han desarrollado con respecto a este enfoque. Mizoguchi et al. (2003) utiliza el modelo FBRL (Lenguaje de representación de función y comportamiento, en inglés *Function and Behavior Representation Language*) basado en el trabajo de Sasajima et al. (1995) y expresa el *comportamiento* como una conceptualización del cambio de los valores en el espacio sobre el tiempo. Además, consideran que una *función* es una interpretación teleológica del *comportamiento* bajo un objetivo dado.

Klein Meyer et al (2007) proponen la teoría FPPT (Función, principio físico y tecnología, en inglés *Function, Physical Principle and Technology*) la cual establece, con respecto al análisis funcional y la selección de tecnología, que los conceptos de modelado se basan en el enfoque FBS con el objetivo de permitir el análisis del *comportamiento* en el inicio del proceso de diseño, cuando aún no hay una geometría disponible. Y Cascini (2013) plantea un modelo en el que a partir de la crítica de la evolución del modelo de Gero (2004) se insertan la identificación de necesidades y la definición de requerimientos.

Las posibilidades de búsqueda, exploración, combinación y selección de sistemas basados en la representación FBS se potencian debido al modelo B-FES propuesto por Tor (2000, 2002). Este modelo es una extensión y un refinamiento del marco de modelado FEBS (Función Entorno Comportamiento Estructura, del inglés *Function Environment-Behavior-Structure*) de Deng (1999).

En el segundo enfoque, se modelan las *funciones* de los objetos en términos de entradas y salidas, y entonces se relacionan de forma directa estas *funciones* a las descripciones físico-estructurales de los objetos (Szykman, 1999). También se conoce como *Functional Modelling* (Modelado Funcional) ya que considera el *comportamiento* como una representación matemática de los estados de un mecanismo (Pahl, 1984; Hirtz, 2002).

El modelado funcional se crea con el objetivo de reducir la ambigüedad existente en el nivel de modelado de un objeto. Para ello, realiza una discriminación entre el significado de *función* y *flow* (flujo). Los *flujos* principales se dividen en tres tipos: material, energía y señal. La definición parte del trabajo previo del Análisis del Valor y de Pahl and Beitz (1986) al inspirar la clasificación de las *funciones* en clases. Así, el conjunto de *funciones* y *flujos* recibe el nombre de *functional basis* (base funcional). Más tarde, se perfeccionó recibiendo el nombre de *Reconciled Functional Basis* (RFB, Base funcional reconciliada) (Szykman, 1999; Hirtz, 2002). A partir de ahí, Sen (2011) a través de la generación de un protocolo para derivar definiciones formales basadas en lógica de verbos funcionales, realiza una crítica al modelo RFB por no poderse formalizar adecuadamente.

A partir de un estudio detallado sobre el marco FBS combinado con la base teórica del trabajo de Garbacz (2006) surge el modelo B-Cube (Chulvi, 2013). Lo que diferencia este modelo del resto es el hecho de que un comportamiento no queda definido por una palabra o taxón sino que el comportamiento viene definido como un vector tridimensional (x, y, z), determinado por sus características y cualidades.

Las últimas investigaciones al respecto ha corrido a cargo de Chao-Chen (2012), quienes han desarrollado el modelo FCBS (Función-célula-comportamiento-estructura, en inglés Function–Cell–Behavior–Structure) para una mejor representación y reutilización del conocimiento en el diseño conceptual a través de la FKC (célula de conocimiento funcional, en inglés Functional Knowledge Cell) como unidad funcional para codificar los principios de uso de diseño racionales. Igualmente Li (2012) plantea un esquema de modelado denominado Función-acción-comportamiento-mecanismo (FABM - function action behavior mechanism) propuesto para generar soluciones inteligentes a partir una función concreta, de acuerdo a las necesidades de los usuarios. Por otro lado, a través de Kannengieser (2012), quienes en base al esquema FBS se establecen tres tipos de *affordances* (reflexivas, reactivas y reflectadas) que varían en su habilidad de tratar con la dinámica de estas interacciones. En base a ello, el rango de uso de un diseño puede expandirse aparte de lo que fuera pretendido por el diseñador. Sin embargo, estos planteamientos indicados siguen siendo bastante teóricos y necesita más desarrollo para poder ser validado experimentalmente.

Dentro de los diferentes usos del esquema FBS, Tang (2011) compara los procesos de diseño de diseñadores en entornos de bocetos de forma tradicional y digital, donde el entorno digital emula el tradicional entorno de papel y lápiz, cara a cara. Por otra parte, Montecchi (2011) introduce un marco de búsqueda para asesorar el estado del arte de un producto o una tecnología y apoyar las actividades de transferencia de tecnología.

A partir de esta revisión, a priori se observa la gran cantidad de trabajos desarrollados en el ámbito de la ingeniería del diseño aplicados en torno al marco FBS. De la misma manera, se percibe la diversificación del enfoque entre las diversas investigaciones al respecto sin llegar a lograrse un consenso (Vermaas, 2012).

2.1.3 B-Cube

El modelo B-Cube (Chulvi, 2013) se basa en tres ejes diferenciados a partir de cuatro términos: P (*perdurant*) que constituye el eje Y, y PQ (Cualidad Física, del inglés *Physical Quality*) estableciendo el eje X; y TQ (cualidad temporal, del inglés *Temporal Quality*) formando el eje Z (Figura 3), definiciones de los cuales están extraídos del trabajo OntoRFB (Aptdo. 2.5.3).

Según Chulvi (2013) para la matriz final los P quedan definidos como:

- ***Achievement***: el comportamiento es no-acumulativo y atómico.
- ***Accomplishment***: el comportamiento es no-acumulativo y no-atómico.
- ***State***: el comportamiento es acumulativo y homeomérico.
- ***Process***: el comportamiento es acumulativo y no-homeomérico.

Las PQ se definen como:

- ***Spatial Location***: la posición que ocupa el elemento físico en el espacio.
- ***Topological Connectedness***: el tipo de conexión topológica que cumple el elemento físico.
- ***Energy***: relativo al estado energético en que se encuentra el elemento físico.
- ***Magnitude***: magnitud física afectada por el elemento físico.
- ***Signal***: relativo al efecto del elemento físico sobre la señal.

Las TQ relacionan como afecta el P a las PQ, por lo que se distribuyen de la siguiente manera:

- **Initial SoA**[c]: La PQ se encuentra al principio y el P actúa eliminándola o reduciéndola.[d]
- **Immutable SoA**: P no varía la PQ, ésta se mantiene en su estado.
- **Final SoA:** La PQ se consigue al final como consecuencia de P.

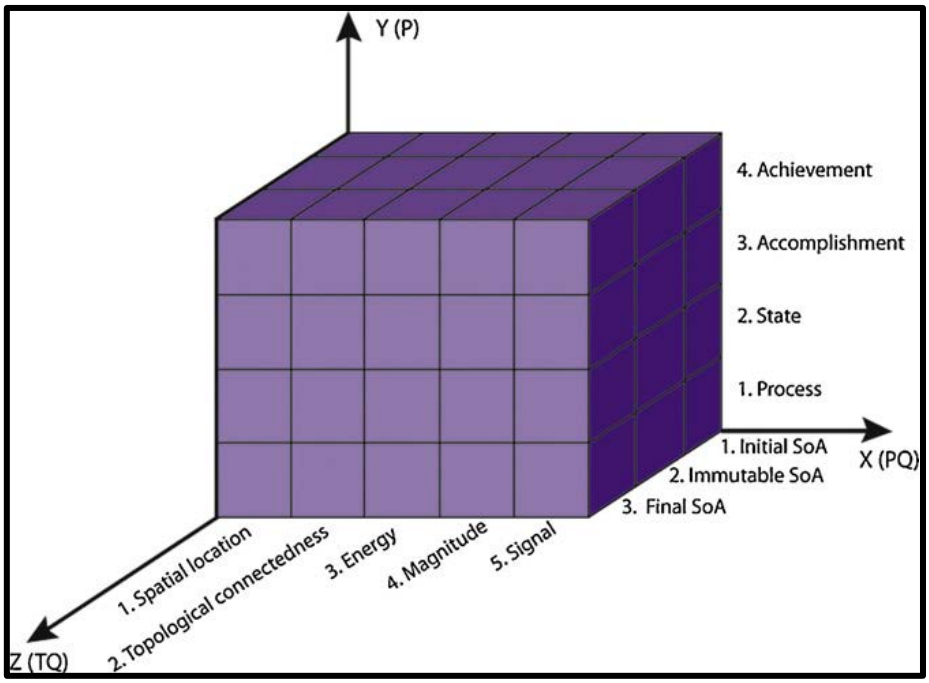


Figura 3: Modelo B-Cube (Chulvi, 2013)

Sin embargo, surgen una serie de cuestiones para mejorar en este modelo:

- El modelo B-Cube define comportamientos, sin profundizar en cómo se aplicaría al ámbito de las funciones siendo que originalmente la ontología OntoRFB, de la cual hereda parte de su estructura básica, es una ontología para generar una coherencia lógica sobre el modelo RFB (Hirtz, 2002).
- Al adaptar el eje Z, referido a las cualidades temporales, o también llamadas TQ, al ámbito formal, es complicado definir cuál es la diferencia entre la clase *Initial SoA* y la clase *Final SoA*. De hecho, en este trabajo se han definido las mismas restricciones para ambas clases. En futuras investigaciones, se espera poder definir de manera concreta este apartado.
- A partir de la aplicación en casos prácticos destaca el hecho que todos los comportamientos que son *Perdurant* State, se identifiquen en el eje Z con la cualidad *Immutable SoA.* Este hecho puede indicar que hayan combinaciones para las cuales no exista ningún comportamiento concreto.
- Con el B-Cube se gana en margen de definición, pero la estructura en forma de cubo, podría transformarse en un árbol de funciones y comportamientos de forma jerárquica tal como se ha constituido la metaontología DOLCE (Aptdo. 2.5.2). Eso permitiría la inclusión de los diferentes tipos de conexiones topológicas.

[c] El término SoA (State of Affairs) se explica en profundidad en el apartado 2.5.3.
[d] Excepto en la PQ "Spatial location", que queda definida por convenio con las demás.

- Con el fin de mejorar el gráfico del B-Cube, se podría modificar el eje Z, para que Immutable SoA tuviera el valor **0**, Final SoA **1** e Initial SoA **-1**. Esa modificación permitiría hacer la identificación de comportamientos más comprensible, ya que según el propio B-Cube, los Immutable SoA son *perdurants* que mantienen su estado, que se mantienen inmutables y de ahí el valor 0; los Final SoA son *perdurants* en los que la cualidad física se consigue como consecuencia, proporcionándole un sentido positivo, de ahí el valor +1; y los Initial SoA debido a que los *perdurants* actúan eliminando o reduciendo la cualidad física en la mayor parte de los casos, es más coherente que se definan por el valor -1.

2.2 Arquitectura cliente/servidor

Con el fin de poder entender la el concepto arquitectura cliente/servidor, estructura informática en la cual se basa el sistema de gestión del conocimiento que se presenta en el Capítulo 4 se van a introducir una serie de nociones básicas sobre la terminología referida a los paquetes de programación, aplicaciones web, arquitectura y servidor disponibles.

Se va a incidir concretamente sobre el lenguaje Java y las aplicaciones web tipo RIA (Aplicación Enriquecida de Internet del inglés, *Rich Internet Application*) pues la aplicación informática que se aporta está basada en dicho lenguaje y tipo de aplicación, respectivamente.

2.2.1 Paquetes de programación

En el ámbito informático, la arquitectura es la estructura lógica y física de los componentes de un sistema. Este capítulo presenta los conceptos relativos al tipo de arquitectura cliente/servidor ya que es el tipo de arquitectura que se utiliza en el desarrollo de la aplicación informática que aporta esta tesis. Dicho sistema requiere de un paquete de desarrollo de programación, para lo cual es necesario conocer las opciones disponibles.

2.2.1.1 Paquetes de desarrollo de programación

Combinando la información de diferentes páginas web, DedaSys (2012) indica que los 10 lenguajes de programación más utilizados actualmente (en orden alfabético) son: C, Java, C++, PHP, JavaScript, Python, C#, Perl, SQL (Lenguaje de consultas estructuradas, del inglés *Structured Query Language*), y Ruby (Figura 4).

De esa lista cabe remarcar Python, Java y *Visual Basic*[e]. Todos ellos son gratuitos o existen en versiones libres, y que pese a que Python es un lenguaje más eficiente (Taft, 2011), hay un mayor uso de aplicaciones desarrolladas en Java. Python y Java son independientes del sistema operativo mientras que *Visual Basic* sólo se ejecuta en máquinas Windows. Adicionalmente, Java es de código abierto. La Tabla 2 contiene una comparativa resumida (Rashed, 2012).

Igualmente Java y Python permiten una mayor utilización comparados con C++ y .NET. Eso es debido principalmente a que estén disponibles vía código abierto reduciendo su coste para las empresas y manteniendo el mismo nivel de eficacia (Mathisen, 2008).

Respecto a *Visual Basic* cabe indicar que también posee una curva de aprendizaje muy rápida y es ampliamente utilizada en la programación de macros para extender y automatizar funcionalidades en documentos, hojas de cálculo, bases de datos así como en programas de CAD (CATIA, *Solidworks*, AutoCAD, entre otros). En el siguiente punto se habla de la utilización de *Visual Basic* en *Solidworks*.

[e] Debido a la terminología informática establecida en inglés se van a utilizar muchos términos sin traducción actual al castellano. Se intentará en la mayor parte de los casos indicar una traducción aproximada. Igualmente se puede buscar su significado en el glosario que se adjunta al final.

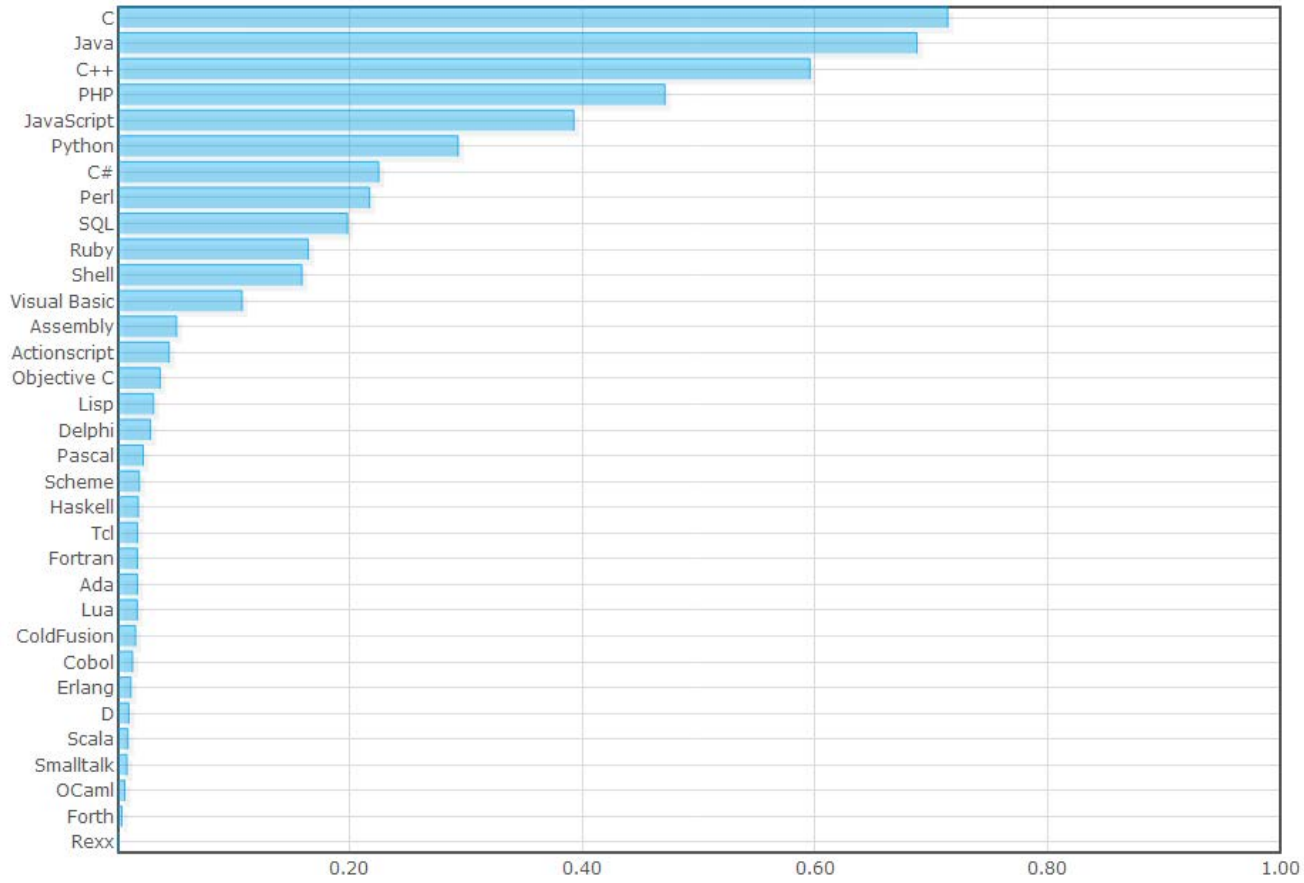


Figura 4: Comparativa de los lenguajes de programación más utilizados (DedaSys, 2012).

Tabla 2: Comparación entre lenguajes de programación

	Java	***Visual Basic***	**Python**
Plataformas	Todas	Windows	Todas
Gratuito	Sí	Sí	Sí
Código abierto	Sí	No	Sí

2.2.1.2 VBa: *Visual Basic* para aplicaciones en *Solidworks*

Microsoft VBa (*Visual Basic* para aplicaciones en inglés, *Visual Basic for applications*) es el lenguaje de macros de Microsoft *Visual Basic* utilizado en la interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) de *Solidworks*.

Una API es una interfaz implementada debido a la interacción entre dos programas informáticos de la misma manera que una interfaz del usuario facilita la interacción entre humanos y ordenadores. Las APIs se implementan para aplicaciones, bibliotecas de programación[f] y sistemas operativos con el fin de determinar el vocabulario y convenciones de llamadas que el programador debería emplear para usar sus servicios. Puede incluir especificaciones para rutinas, estructuras de datos, clases de objetos y protocolos usados para comunicarse entre el consumidor y el implementador de la API.

A continuación se indica cómo se aplica en el Modelo de objetos *Solidworks* API, es decir, en las propiedades de los objetos utilizados por la API de *Solidworks*. Así, un modelo de objetos a nivel general es la colección de objetos o clases por las cuales un programa puede examinar y manipular algunas partes específicas de su mundo.

[f] Una biblioteca de programación es un conjunto de subprogramas utilizados para desarrollar software.

2.2.1.3 Modelo de objetos *Solidworks* API

La API de *Solidworks* se utiliza para personalizar el software de *Solidworks* a través del modelo de objetos *Solidworks* API (Figura 5). El objeto *SldWorks* es el objeto de la aplicación de nivel superior en la jerarquía de objetos de *Solidworks* API. Todos los demás objetos de *Solidworks* se encuentran por debajo del objeto *SldWorks* en la jerarquía de objetos y se puede acceder directamente o indirectamente.

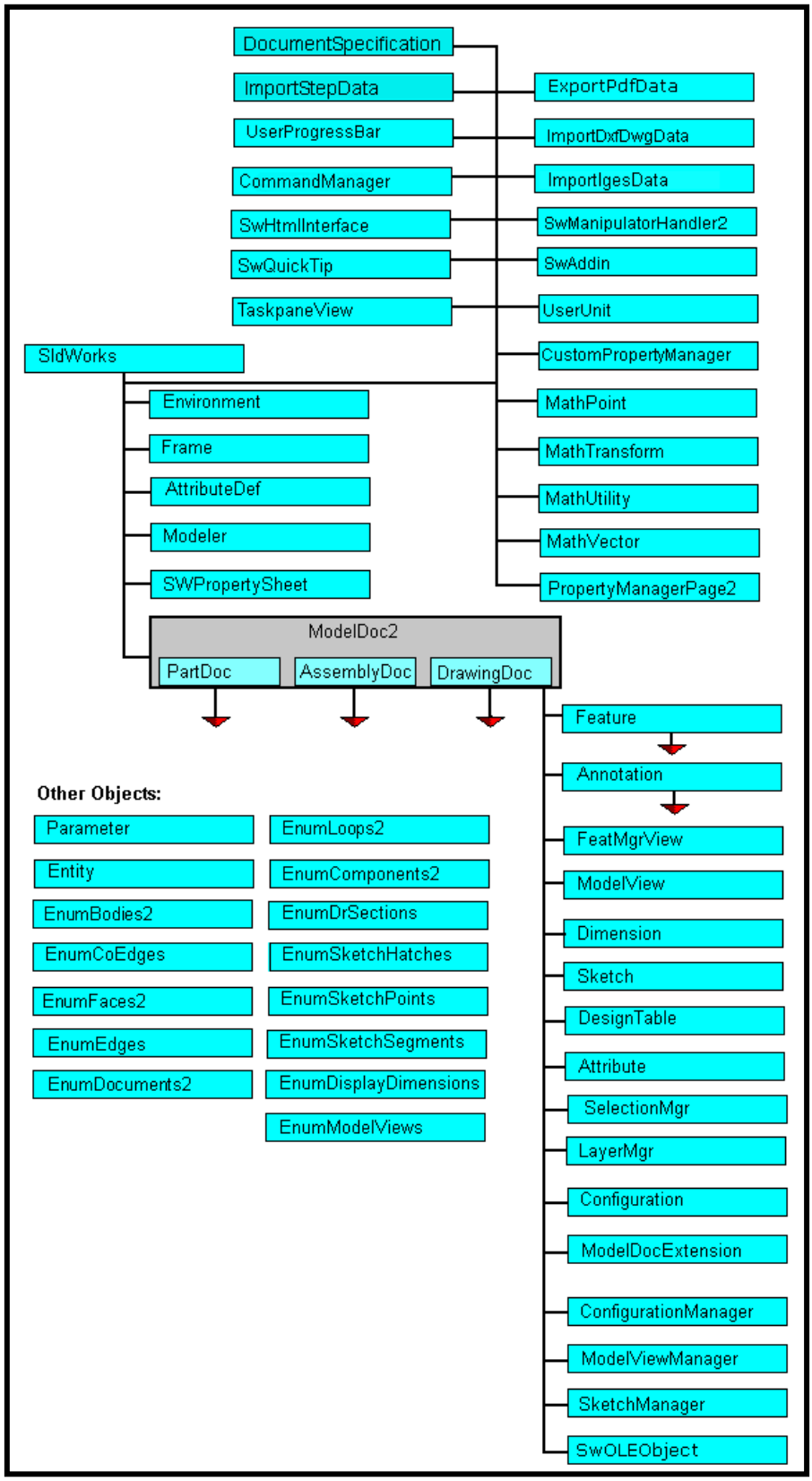


Figura 5: Modelo de objetos *Solidworks* API

Si un objeto sólo se puede acceder indirectamente, debe hacer referencia al objeto de otro objeto más alto en la jerarquía de objetos. Por ejemplo, el objeto *SketchSpline* sólo se puede acceder indirectamente, no puede existir por sí mismo. Debe hacer referencia al objeto *SketchSegment* porque existe sólo dentro del contexto del objeto *SketchSegment*.

La mayoría de los objetos de *Solidworks* API corresponden a la funcionalidad de la interfaz de usuario, sin embargo, varios objetos de la API de *Solidworks* proporcionan una funcionalidad que sólo se puede acceder a través de la API de *Solidworks*. Por ejemplo, el objeto *Feature* (función) le permite acceder al árbol del diseño del *FeatureManager* (gestor de funciones), pero el objeto *Attribute* (atributo) es un objeto de *Solidworks* API sólo porque no hay una interfaz de usuario correspondiente funcionalidad.

A continuación se presenta los conocimientos necesarios referidos a la Plataforma *Java* ya que la aplicación web que se aporta en esta tesis está programada en dicho lenguaje.

2.2.1.4 Plataforma Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel. Su habilidad más conocida es la de poderse ejecutar en navegadores web y ganar así una independencia real de las plataformas sobre las que se ejecute. La conveniencia de ejecutar *applets*[g] Java directamente de la web, sin tener ningún conocimiento sobre lo que es un compilador o intérprete, ya que está embebido en el navegador, lo hizo muy popular en Internet. Es considerado a menudo como el "lenguaje de Internet" (Georgatos, 02).

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java, u otros lenguajes que compilen a *bytecode*[h], y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de las aplicaciones, y un conjunto de bibliotecas estándar que ofrecen una funcionalidad común.

Se definen tres plataformas en un intento por cubrir distintos entornos de aplicación:

- **Java SE**: Plataforma Java, Edición Estándar (*Java Platform*, *Standard Edition*), (antes J2SE) para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.
- **Java EE**: Plataforma Java, Edición Empresa (*Java Platform, Enterprise Edition*) (antes J2EE) orientada a entornos distribuidos empresariales o de Internet.
- **Java ME**: Plataforma Java, Edición Micro (*Java Platform, Micro Edition*), (antes J2ME) orientada a entornos de limitados recursos, como teléfonos móviles o PDAs (Asistente digital personal, del inglés *Personal Digital Assistant*).

Para poder entender la plataforma Java EE es necesario introducir el concepto de aplicaciones web, el que permite explicar la arquitectura cliente/servidor utilizada.

2.2.2 Aplicaciones web

Una aplicación web consiste básicamente en al menos dos partes, llamadas capas (*tier*): un servidor, en la que los principales componentes se están ejecutando y un cliente, donde se muestra la interfaz de usuario, generalmente en un navegador web, cuyo usuario no posee control sobre los recursos, sino que es el servidor el encargado de manejarlos. Una vez que el servidor acepta el pedido la información requerida es

[g] Un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

[h] El bytecode es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

suministrada al cliente que efectuó la petición, siendo este último el responsable de proporcionar los datos al usuario con el formato adecuado (Roth, 2006).

Con las últimas tecnologías, una aplicación web puede proporcionar la misma funcionalidad que un software que el usuario instale y ejecute en su máquina local. Las ventajas de una aplicación web son los siguientes (Díaz Ortuño, 2003):

- No se consume casi ningún espacio en el disco duro (excepto el caché y el contenido *offline* que es guardado por el navegador web).
- No hay conflictos con otras aplicaciones en la máquina cliente.
- Accesible desde cualquier lugar.
- No necesita instalación.
- Las actualizaciones se manejan de forma central.
- Los datos son los mismos para todos los usuarios.

La interfaz de usuario de las aplicaciones web se crea normalmente con páginas escritas con el lenguaje HTML (Lenguaje de marcación de hipertexto, del inglés *Hypertext Markup Language*) y la comunicación entre el cliente y el servidor funciona con las peticiones y respuestas HTTP. HTTP (Protocolo de transferencia de hipertexto del inglés, *Hypertext Transfer Protocol*) es uno de los protocolos de Internet que puede ser utilizado para fines de comunicación entre *hosts*[i].

Otra aproximación dentro del ámbito de las aplicaciones web es utilizar Adobe Flash Player (reproductor multimedia) o Java applets para desarrollar parte o toda la interfaz de usuario. Como casi todos los navegadores incluyen soporte para estas tecnologías (usualmente por medio de plug-ins), las aplicaciones basadas en Flash o Java pueden ser implementadas con aproximadamente la misma facilidad. Dado que ignoran las configuraciones de los navegadores, estas tecnologías permiten más control sobre la interfaz, aunque las incompatibilidades entre implementaciones Flash o Java puedan crear nuevas complicaciones, debido a que no son estándares. Por las similitudes con una arquitectura cliente/servidor existen discrepancias sobre el hecho de llamar a estos sistemas “aplicaciones web”, un término alternativo es RIA.

2.2.2.1 Aplicación Enriquecida de Internet

Las aplicaciones RIA son un nuevo tipo de aplicaciones que reúnen una serie de ventajas frente a las aplicaciones web pues utilizan un navegador web estandarizado para ejecutarse y por medio de plug-ins[j] o independientemente una máquina virtual[k], lo cual provoca que desde el principio se cargue toda la aplicación y sólo se produzca comunicación con el servidor cuando se necesitan datos externos, es decir, cuando el cliente solicite aquellos datos que requiere.

Además, las RIA, como herramientas que surgen durante el auge de la web 2.0, aprovechan las interacciones con el usuario para ser más dinámicas y útiles. Permite, por tanto, agilizar su acceso, garantizando el acceso a la aplicación desde cualquier ordenador en cualquier lugar del mundo. Y a pesar de que su desarrollo es más complejo que otro tipo de aplicaciones de escritorio, los esfuerzos se justifican por varios motivos (El-Hadik, 2009):

- Mejora la conectividad permitiendo un despliegue instantáneo de la aplicación.
- Gran capacidad multimedia gracias a que estos entornos tienen reproductores internos ofreciendo así al usuario una mayor satisfacción.
- No necesitan instalación (sólo es necesario mantener actualizado el navegador web).

---

i Un host ("anfitrión", en español) es todo equipo informático que posee una dirección IP y que se encuentra interconectado con uno o más equipos.

[j] Un complemento (*plug-in)* es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como *plug-in* (del inglés "enchufable"), add-on (agregado), conector o extensión.

[k] Una máquina virtual es un software que emula a una computadora y puede ejecutar programas como si fuese una computadora real.

- Las actualizaciones se realizan de forma automática.
- Se pueden utilizar desde cualquier ordenador con una conexión a Internet sin depender del sistema operativo que éste utilice.
- Generalmente es menos probable la infección por virus, que utilizando por ejemplo programas ejecutables.
- Dispone de una mayor capacidad de respuesta, ya que la interacción con el servidor se realiza directamente lo cual incrementa la productividad y el rendimiento.
- Ofrecen aplicaciones interactivas que no se pueden obtener utilizando solo HTML (exceptuando la última revisión HTML5 [Pilgrim, 2012]), incluyendo arrastrar y pegar, cálculos en el lado del cliente sin la necesidad de enviar la información al servidor.

2.2.2.2 Tipos de RIA

Hay muchas herramientas para la creación de entornos RIA. Dichos entornos deben ser instalados en el ordenador antes de ejecutar la aplicación. La propia plataforma ya se ocupa de descargar, actualizar, verificar y ejecutar el RIA. Algunos ejemplos de plataformas son: *Adobe Flex* y *Adobe AIR* de Adobe, *AJAX*, *Curl*, *ExtJs*, *ICE faces*, *Open Laszlo*, *Oracle ADF Faces*, *Silverlight* de Microsoft, *JavaFX* Script de Sun Microsystems, *Bindows*, *GWT*, *Javascript MVC*, *Javascript* y *ZK Framework*.

En la Tabla 3 se establece una comparativa con las distintas RIA existentes en el mercado actualmente indicando los siguientes valores comparativos:

- **Desarrollador**: Empresa que ha programado la aplicación.
- **Lenguaje**: Lenguaje de programación que utiliza la RIA.
- **Entorno de ejecución**: Es el sistema que suministra servicios de software para procesos o programas mientras que el ordenador se está ejecutando.
- **Código Abierto**: Si el software es distribuido y desarrollado libremente.
- **Otras características**: Recoge otras propiedades relevantes en la RIA.

Dentro de las distintas posibilidades disponibles, una de las opciones mejores por funcionalidad y criterio estético es *Adobe Flex.* Además Flex es muy prominente y familiar para los desarrolladores profesionales pues consiste en un conjunto de productos diseñados de manera efectiva sobre la base de la prestación eficiente de alto rendimiento (Fraternali et al., 2010).

Por otra parte, las aplicaciones Web se han convertido en pocos años en complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, dando servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta. Esto exige reflexiones sobre la mejor arquitectura y las técnicas de diseño más adecuadas.

Por ello es necesario conocer el diseño conceptual a través del cual se ha constituido y la estructura operacional fundamental del sistema, o en otras palabras, lo que es llamado arquitectura de la aplicación. Concretamente en el capítulo 2.2.4 se tratará el caso de las arquitecturas cliente/servidor, debido a su trascendencia en la comprensión del proyecto. Antes conviene introducir el concepto de servidor para poder comprender el citado tipo de arquitectura.

Tabla 3: Comparativa sobre entornos de RIA

Nombre	Desarrollador	Lenguaje	Entorno de ejecución	Código Abierto	Otras características
Java Applets	Sun Microsystems	Java	Java Runtime Environment (JRE)	Parcialmente	Normalmente se ejecuta en un navegador.
JavaFX	Sun Microsystems	JavaFX Script	JavaFX Runtime(viene con jre)	Parcialmente	Puede ser usado en el escritorio, con un navegador y en móviles.
Adobe Flash	Adobe Systems	Action Script	Plugin de Flash Player		Se centra en audio, video y animaciones

Nombre	Desarrollador	Lenguaje	Entorno de ejecución	Código Abierto	Otras características
Adobe FLEX	Adobe Systems	MXML, ActionScript, Spark	Flex builder	No	Entorno de ejecución basado en eclipse y permite crear *GUIs*.[l]
Adobe Integrated Runtime (AIR)	Adobe Systems	Flash, Flex, HTML y Ajax	Adobe Integrated Runtime	No	Acceso sin necesidad de navegador. Permite archivos PDF.
Microsoft Silverlight (Linux: Moonlight)	Microsoft	Lenguajes .NET, lenguajes Scripting, XAML	Plugin Silverlight	Parcialmente (Moonlight)	Acceso sin conexión. Incluye reproductor multimedia.
Open Laszlo	Laszlo Systems	LZX, JavaScript	Flash, JavaScript	Sí	Realmente Código Abierto. Similar a Adobe Flex
UniPaaS	Magic Software	UniPaaS	UniPaaS	No	Permite intercambiar entre cliente Full y aplicación RIA.
Curl	Curl Inc.	Curl	Curl Surge RTE	No	*Sandbox* [m]para aplicaciones "sin conexión".
Omnis	Tiger Logic Corp	Omnis	Omnis web client	No	Entorno de desarrollo integrado: "Omnis Studio"

2.2.3 Servidor

Este apartado tiene una importancia relevante para la posterior comprensión de la descripción informática de la aplicación web que se aporta en esta tesis en el capítulo 4.5. Para ello inicialmente se indica la definición de servidor, y concretamente de un servidor de aplicaciones, para poder comprender las características de un servicio web y el lenguaje de programación de base de datos MySQL.

2.2.3.1 Definición

En informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan hacer uso de ellos.

En Internet, un servidor web o servidor HTTP es un ordenador que usa el protocolo http para enviar páginas web al ordenador de un usuario cuando el usuario las solicita. Dichos servidores junto a los referidos al apartado de correo y base de datos, constituyen el mayor número de accesos por parte de los usuarios de Internet. Para ello se utiliza un programa que procesa cualquier aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

Entre los servidores web más conocidos se encuentran *4D WebSTAR*, *AOLserver*, *Apache*, *BadBlue*, *Baikonur Web App Server*; *Covalent Enterprise Ready Server*, *ESAWEB*, *Go Ahead WebServer*, *Java Server, Jigsaw,*

[l] GUI (Interfaz gráfica del usuario, del inglés *Graphical User Interface*): Es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

[m] Sandbox: En el contexto de desarrollo de software o desarrollo web, entorno de pruebas que aísla los cambios en el código, fruto de la experimentación, del propio entorno de producción o entorno de edición.

RapidSite, *RoxenWebServer*, *Sambar Server*, *Savant*, *Servertec Internet Server*, *SimpleServer*: *WWW*, *vqServer*, *WN* o *Zeus Web Server* (Roth, 2006).

Sin embargo, cabe distinguir la diferencia entre un servidor web y un servicio web que es una parte de un programa informático que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Dentro del ámbito del lenguaje de programación Java, uno de los servidores de aplicaciones para servicios web más conocido es JBoss, servidor de aplicaciones J2EE *Open Source*[n] de Red Hat inc. Y el API de Java para crear servicios web es JAX-WS (API Java para servicios Web XML, del inglés Java API XML – Web Service) (Labarga et al., 2007).

2.2.3.2 Servidor de aplicaciones

Como consecuencia del éxito del lenguaje de programación Java, el término servidor de aplicaciones usualmente hace referencia a un servidor de aplicaciones Java EE. Un servidor de aplicación funciona como un componente de la capa intermedia en las aplicaciones Java EE. Es el componente que permite el carácter distribuido de una aplicación con sus servicios. Éste cuenta con servicios para la comunicación entre el cliente y un sistema de información como una base de datos. Estos servicios incluyen funciones de transacciones, persistencia, así como las funciones de seguridad y otros (Roth, 2006).

Un servidor de aplicaciones tiene otro componente importante, un servidor web. Hay muchas implementaciones de servidores de aplicaciones apoyando Java EE, tal como se comentará en el punto siguiente.

Los servidores de aplicaciones típicamente incluyen también *middleware* (o software de conectividad) que les permite intercomunicarse con variados servicios, para efectos de confiabilidad o seguridad entre otros. Los servidores de aplicación también brindan a los desarrolladores una API, de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación web moderna.

Un ejemplo común del uso de servidores de aplicación (y de sus componentes) son los portales de Internet, que permiten a las empresas la gestión y divulgación de su información, y un punto único de entrada a los usuarios internos y externos. Teniendo como base un servidor de aplicación, dichos portales permiten tener acceso a información y servicios (como servicios Web) de manera segura y transparente, desde cualquier dispositivo.

2.2.3.3 Tipos de servidores

La elección de un servidor de aplicaciones adecuado es una cuestión tan importante como que el servidor es un componente esencial en una aplicación Java EE. Hay muchas implementaciones de servidores de aplicaciones que construyen un entorno de ejecución para Java EE. Para elegir un servidor de aplicaciones apropiado es importante definir una selección de criterios aceptable. Para este trabajo, los siguientes puntos son decisivos (Roth, 2006):

1. Las tecnologías integradas deben estar actualizadas.
2. El precio es un factor crítico porque los costos de algunos servidores de aplicaciones son bastante altos. Un servidor de aplicaciones libres es claramente preferible.
3. Las características especiales del servidor deben ayudar con el desarrollo y la implementación de la aplicación.
4. Cuestiones menores como la simplicidad de la instalación o manejo del servidor.
5. Apoyo comunitario y la cuota de mercado del servidor.

Las ventajas principales del servidor JBoss son que es un servidor popular y reconocido. Además tiene un mantenimiento y dispone de servicio de atención de confianza. Tal como se ha indicado se ejecuta en cualquier plataforma que soporte Java implementando todo el paquete de servicios de J2EE. El único

[n] *Open source* o código abierto en castellano se conoce por la expresión referida al software distribuido y desarrollado libremente.

inconveniente que se puede observar es que su configuración es compleja (Labarga et al., 2007). A menudo es considerado como uno de los servidores de aplicaciones más importantes, además de *Websphere* de IBM o *WebLogic* de Oracle, que son productos comerciales. Los desarrolladores de JBoss principales son empleados actualmente por Red Hat, una de las mayores compañías de software del mundo de código abierto. En el campo industrial, JBoss se está volviendo más y más popular (Fleury & Revelbel, 2003).

2.2.3.4 Interfaz del Servicio Web

Actualmente los servicios Web proporcionan un marco sistemático y extensible para la interacción de aplicación a aplicación, construido en la cima de los protocolos de Internet existentes, y basado en estándares abiertos XML.

Los servicios Web definen un mecanismo estandarizado para describir, localizar y comunicarse con las aplicaciones en línea. Esencialmente, cada aplicación se convierte en un componente de servicios Web accesibles que describe el uso de estándares abiertos.

El marco de servicios Web se divide en tres áreas - los protocolos de comunicación, descripción de servicios, y el descubrimiento de servicios - y hay especificaciones desarrolladas para cada uno. Las especificaciones que se encuentran las más destacadas y estables en cada área son:

_ SOAP (Protocolo de Acceso de Objetos Simples, del inglés *Simple Object Access Protocol*) que permite la comunicación entre los servicios Web;

_ WSDL (Lenguaje de Descripción de Servicios Web, del inglés *Web Services Description Language*), describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios web, y

_ UDDI (Descripción, descubrimiento e integración universal, del inglés *Universal Description, Discovery and Integration*) que es un registro de descripciones de servicios web.

Concretamente, SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML para mensajes y llamadas a RPC (procedimiento remoto, del inglés *Remote Procedure Call*). Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros y está actualmente bajo el auspicio de la W3C (Consorcio de la red informática mundial, del inglés World Wide Web Consortium). Es uno de los protocolos utilizados en los servicios Web (Curbera et al, 2002).

En una aplicación Java EE, una base de datos puede guardar información para otros componentes de un sistema. Esta información es persistida, es decir, mantenida en una estructura predefinida, para permitir a las partes involucradas del sistema acceder a los datos a través de diversas búsquedas. Sin embargo, cuando se trabaja con diferentes bases de datos en la que los datos tienen relación entre diferentes tablas es necesario un RDBMS (sistema de gestión de forma relacional, del inglés *Relational Database Management System*). Hay varios sistemas disponibles, como por ejemplo Oracle, Microsoft SQL Server o MySQL, que es uno de los programas más utilizados bajo la filosofía de código abierto para sitios web.

2.2.3.5 MySQL

MySQL es un RDBMS que se ejecuta en un servidor permitiendo un acceso multiusuario a un número determinado de bases de datos. MySQL archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido (MySQL, 2012).

MySQL está escrito en C y C + +, se ejecuta en varios sistemas operativos y ofrece muchas API para permitir el acceso a los lenguajes de programación: C, C + +, Eiffel, Java, Perl, PHP, Python y Ruby.

A continuación, se cambia de ámbito realizando una breve descripción de las arquitecturas más comunes que se pueden encontrar.

2.2.4 Arquitectura

Una arquitectura es una estructura organizativa de un sistema de software compuesta por diferentes bloques constituyentes, sus propiedades externas y las relaciones entre sí y con el entorno. Está diseñada para que la estructura del sistema permita las funcionalidades deseadas a través del criterio de integración, evolución y mantenimiento de éste.

La arquitectura de los sistemas ha ido evolucionando con el tiempo, acompañando los cambios tecnológicos y paradigmas del diferente software creado. En este caso, a continuación se van a tratar la arquitectura de los sistemas cliente/servidor y concretamente los basados en Java ya que la aplicación web Adobe Flex utilizada en la aplicación web que aporta en esta tesis se basa en este tipo en concreto.

2.2.4.1 Arquitectura cliente-servidor

La arquitectura cliente/servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Se especializa para aquellos sistemas operativos que se encuentran distribuidos a través de una red de ordenadores (Orfali et al., 2009).

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes ordenadores aumentando así el grado de distribución del sistema.

De ahí que una red cliente/servidor sea aquella red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.

2.2.4.2 Modelos y tipologías de arquitecturas cliente/servidor

El esquema de la arquitectura cliente/servidor se basa en capas, es decir, esquemas abstractos de aplicaciones distribuidas genéricas que se corresponden con las funciones típicas en un sistema.

Hay principalmente 3 capas relevantes:

1. **Capa de presentación (interfaz de usuario)**: Se ocupa de la interacción con el usuario, presenta los datos y recibe las entradas.

2. **Capa de aplicación/negocio (lógica de negocio)**: Se responsabiliza de las tareas propias de la aplicación concreta y aplica las reglas de negocio[o] sobre los datos y las entradas de usuario.

3. **Capa de datos (almacenamiento y acceso a datos)**: Es responsable de la gestión y almacenamiento permanente de los datos.

Cada tipo de sistema cliente/servidor distribuye esas capas de modo distinto entre los componentes cliente y servidor, pudiendo estar conformado en 2, 3 o n-capas en función de la ubicación física de las distintas funcionalidades.

[o] La lógica de negocio es un término técnico utilizado generalmente para describir los algoritmos funcionales que gestionan el intercambio de información entre una base de datos y la interfaz del usuario. Consiste en el procesamiento de los datos capturados por un programa informático como entrada y la posterior entrega de resultados al usuario por medio de la interfaz con dicho usuario.

En las arquitecturas multicapa (n-capas) la lógica de negocio se reparte en diferentes capas/niveles ubicadas entre el cliente y los datos mientras que las capas intermedias se proporcionan servicios entre sí ya que cada nivel se comunica sólo con los niveles contiguos a través de interfaces bien definidos.

Esta es la estructura típica que tienen los sistemas basados en componentes distribuidos (objetos distribuidos). Un ejemplo de este tipo de sistema es la plataforma Java EE.

2.2.4.3 Java Enterprise Edition

Java EE, tal como se ha comentado en el apartado 2.2.1.4 es una plataforma de programación— elemento conformante de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java con arquitectura de *n* capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

La programación de Java EE proporciona un mecanismo de muy bajo nivel para la comunicación e intercambio de datos entre cliente y servidor. El protocolo de comunicación entre ambos, determina el establecimiento de la conexión, para lo cual es necesario un protocolo común. En la programación de Java EE, la comunicación es en ambos sentidos a la vez entre cliente y servidor; siendo responsabilidad del sistema el llevar los datos de una máquina a otra, dejando al programador el proporcionar significado a esos datos. Parte de la información que fluye entre las dos máquinas es, pues, para implementar el protocolo, y el resto son los propios datos que se quieren transferir (Singh, 2002).

Además, Java EE ofrece funcionalidades para la portabilidad, robustez, escalabilidad y seguridad en aplicaciones de servidor Java. Es básicamente una extensión de Java SE, pero define varias clases adicionales y APIs que proporcionan los mecanismos necesarios para estas funcionalidades. Para ello se requiere una clase especial de servidor, un servidor de aplicaciones Java EE que implemente dichas clases (Roth, 2006).

A pesar de una aplicación Java EE pueda consistir en una arquitectura de tres o cuatro niveles, las aplicaciones Java EE de varios niveles se consideran normalmente aplicaciones de tres niveles, ya que se distribuyen en tres lugares: las máquinas cliente, la máquina de servidor Java EE, y la base de datos o las máquinas de legado en la parte final (*back end*). Las aplicaciones de tres niveles que se ejecutan de esta forma extienden el modelo estándar de dos niveles cliente/servidor mediante la colocación de un servidor de aplicaciones multiproceso entre la aplicación cliente y el almacenamiento del *back end*.

Las aplicaciones Java EE están formadas por componentes. Un componente Java EE es una unidad autónoma de software funcional que se ensambla en una aplicación Java EE con sus clases y archivos relacionados y que se comunica con otros componentes. Así, los componentes Web son la API preferida para la creación de un programa de cliente web, porque no hay plug-ins o los archivos de políticas de seguridad necesarios en los sistemas cliente. Además, los componentes web permiten un diseño más limpio y más modular de aplicaciones ya que proporcionan una manera para separar las aplicaciones de programación de diseño de páginas web. El personal involucrado en el diseño de la página web por tanto no necesita comprender la sintaxis del lenguaje de programación Java para hacer su trabajo.

2.2.4.4 Java EE: Clientes

Un cliente Java EE puede ser un cliente web o una aplicación cliente. Consiste en dos partes:

1. Las páginas web dinámicas que contienen distintos tipos de lenguaje de marcado (HTML, XML, etc.), que son generados por los componentes Web que se ejecutan en la capa web.

2. Un navegador web, que permite que se ejecuten las páginas recibidas desde el servidor.

Un cliente web es a veces llamado un cliente ligero[p]. Los clientes ligeros no suelen consultar bases de datos, ejecución de reglas de negocio complejas, o conectarse a las aplicaciones heredadas. Por tanto, las

[p] Un cliente ligero es un ordenador o programa informático en una arquitectura de red cliente-servidor que depende primariamente del servidor central para las tareas de procesamiento, y principalmente se enfoca en transportar la

operaciones complejas se ejecutan en el servidor Java EE, donde se aprovechan la seguridad, velocidad, servicios y confiabilidad de las tecnologías disponibles (Nieh, 2000).

En cambio, una aplicación cliente se ejecuta en una máquina cliente y proporciona una forma para que los usuarios puedan manejar tareas que requieren una interfaz de usuario más rica de la que pueda ser proporcionada por un lenguaje de marcas. Sin embargo, en la plataforma Java EE, la lógica de negocio, se gestiona a través de la capa EIS (Sistema de información de empresa del inglés, *Enterprise Information System*), apartado que se comentará a continuación.

2.2.4.5 Java EE: La Capa EIS

La capa del sistema de información de empresa se encarga del software utilizado y que incluye sistemas de infraestructura empresarial, tales como la planificación de recursos empresariales, procesamiento de transacciones principales, sistemas de bases de datos y otros sistemas de información heredados. Por ejemplo, los componentes de aplicación Java EE pueden tener acceso a los sistemas de información de la empresa para la conectividad de base de datos.

Para la gestión adecuada de esta capa es necesario un servidor de aplicaciones apropiado. Se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones. Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones (Sharma et al, 2002).

Para conectar la capa de negocio y la capa EIS, se puede hacer con un ORM (mapeo objeto/relacional del inglés, *Object/Relational Mapping*), una técnica que mapea una representación de datos a partir de un modelo de objetos para obtener de esa manera un modelo de datos relacional. El objetivo es lograr la persistencia de los datos, es decir, guardar en un modelo de datos para poder restaurarse en un momento posterior con todas sus propiedades y relaciones.

Una de las herramientas actuales de ORM más popular para las aplicaciones Java es Hibernate.

2.2.4.6 Hibernate

Este software gratuito de código abierto regula el mapeo de clases Java a tablas de bases de datos y en un paso posterior, los tipos de datos Java de los tipos de datos SQL. Además de esta asignación, Hibernate también ofrecen consulta y recuperación de datos. Con estas características se puede simplificar el proceso de desarrollo y reducir la cantidad de tiempo que un desarrollador tendría que gastar en el manejo de los problemas de SQL y JDBC (Conectividad de bases de datos Java, del inglés *Java Database Connectivity*) si tuviera que poner en práctica todo manualmente. Una gran ventaja es el HQL (lenguaje de consultas de Hibernate del inglés, *Hibernate Query Language*) que permite al desarrollador acceder a bases de datos de diferentes proveedores, con un lenguaje unificado en lugar de tener que aplicar las específicas de su proveedor de código SQL. HQL puede hacer frente a los diferentes tipos de SQL, gracias a varios dialectos HQL (Sharma et al, 2002).

Una vez que un objeto se guarda en un modelo de datos debe restaurarse en un momento posterior en el tiempo con todas sus propiedades y relaciones. Un objeto que cumple con estas condiciones se llama persistente.

---

entrada y la salida de datos entre el usuario y el servidor remoto. En contraste, un cliente pesado procesa la mayor cantidad de información posible y pasa solamente los datos para las comunicaciones y el almacenamiento al servidor (Nieh, 2000).

2.2.4.7 Arquitectura de Adobe Flex

Las aplicaciones Flex se aprovechan de Adobe Flash Player y esto permite al desarrollador extender sus capacidades permitiendo al cliente tener aplicaciones más ricas en términos reales. Además representa un nuevo nivel de presentación en las aplicaciones de arquitectura de varios niveles y permite una fácil integración con la funcionalidad del lado del servidor (Preciado et al., 2007).

La Figura 6 muestra el diagrama de la arquitectura de Flex incidiendo en la parte lógica mientras muestra de izquierda a derecha el procesamiento de la información. Tiene 3 capas en el lado del servidor: capa de datos que es la menos expuesta; capa de la aplicación (indicada en la figura por la lógica de negocio) que procesa los datos capturados en el servidor y los enlaza con la siguiente capa que es la interfaz de presentación que es mostrada al cliente.

La RIA toma lugar al lado del cliente (Capítulo 2.2.2). El entorno de ejecución de la interacción con el usuario (IU) permite el acceso a la información desde cualquier ordenador conectado a Internet y gracias al motor HTML se pueden ejecutar en un navegador web.

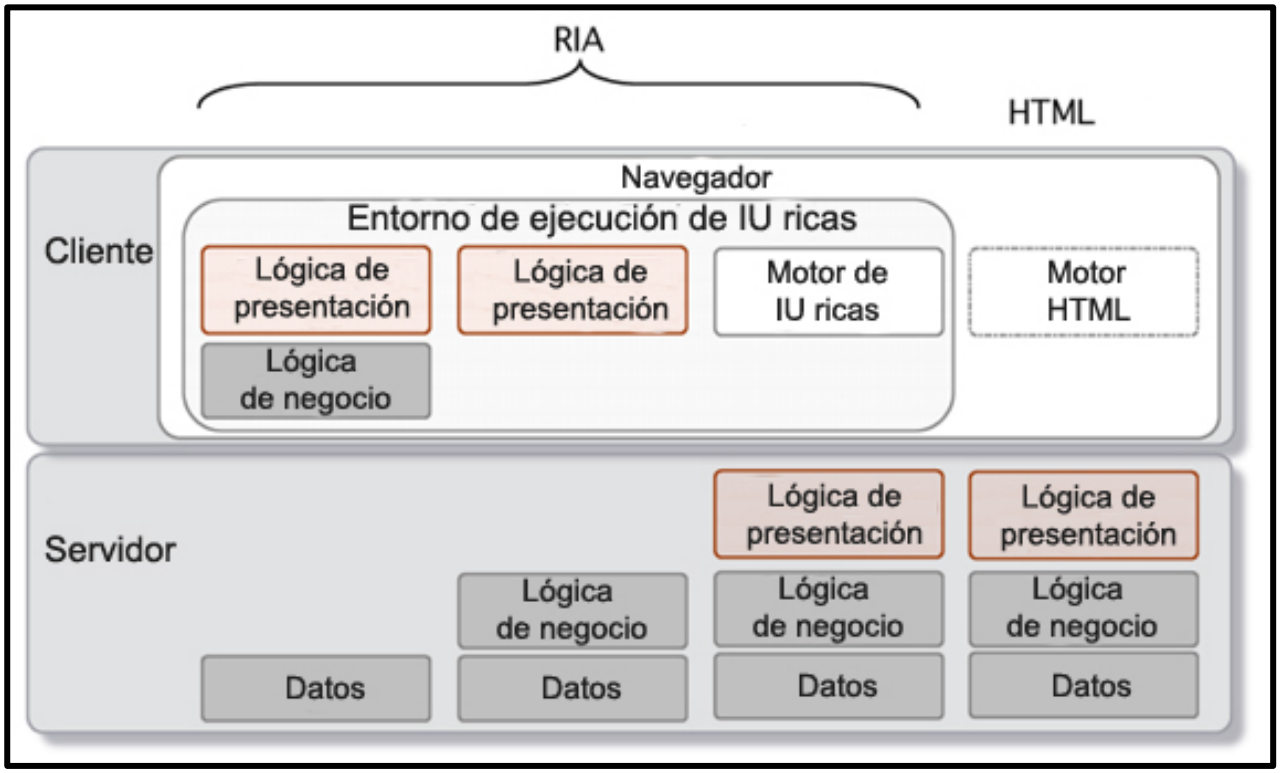


Figura 6: Diagrama de la arquitectura de FLEX

A continuación se explicará con detalle en qué consiste la Ingeniería Basada en el Conocimiento y concretamente la metodología MOKA; lo cual sirve para entender el programa informático KSS (Sistema de Compartición del Conocimiento, del inglés *Knowledge Sharing System*) descrito en el apartado 2.3.4. Finalmente se indica la relevancia que tienen la web 2.0 y la web semántica o web 3.0.

2.3 Ingeniería Basada en el Conocimiento

En el rediseño de productos el conocimiento es imprescindible para poder optimizar costes de tiempo y producción. No obstante, hasta la actualidad ha habido ciertos condicionantes, como la falta de comunicación entre software existente, que han impedido formalizar dicho conocimiento y que éste pueda estar disponible de forma unificada y extendida.

A mediados de los años 80 surgen los primeros esfuerzos para formalizar la información de un diseño a partir de sistemas basados en AI, (Inteligencia Artificial, en inglés *Artificial Intelligence*). El objetivo de estos sistemas era que un ordenador fuera capaz de resolver problemas que podrían ser solucionados por una persona. La ventaja inicial es que diseños similares con una geometría distinta pudieran ser generados más

rápidamente que con un sistema tradicional. Sin embargo, dichos sistemas eran realmente complicados de mantener, pues si había una pequeña alteración en el proceso, el sistema debía modificarse para mantener su funcionalidad (Chapman, 1999).

En los años 90, el avance de la informática permitió la posibilidad de crear macros (macro-instrucciones) que automatizan tareas rutinarias y que son fáciles de compartir entre diferentes aplicaciones a la vez que se introducían nuevas facilidades de interconexión y herramientas gráficas simples para el uso de la red, iniciándose el auge que actualmente se le conoce a Internet. Un caso de ello, conocido ampliamente, son las macros de Microsoft Office. Así las macros comienzan a utilizarse en los programas de CAD que permiten el diseño de productos primero en dos dimensiones (2D) y luego posteriormente en tres dimensiones (3D). Este hecho unido al desarrollo de los KBS, sistemas expertos que formalizaban el conocimiento recurriendo a expertos en la materia, permitió la creación de la KBE.

Sin embargo, dichos sistemas de conocimiento acarrean una serie de inconvenientes. Uno de ellos se debe a su proceso de abstracción pues almacenan la información de la manera más formalizada posible, resultando en muchos casos complejos para su utilización por parte de un diseñador o un operario. Un segundo caso que puede ocurrir es la necesidad de la determinación de un gran número de reglas formales a utilizar por parte del trabajador para transferir su conocimiento, lo cual dificulta su rentabilidad para la empresa. Y otra cuestión aparte, es cuando el conocimiento que se quiere extraer es el referido al trabajo de un diseñador industrial. En ese caso, aquellas actividades dependientes de un proceso creativo, actualmente no se pueden automatizar.

En los siguientes apartados, se explica la definición de KBE y sus fundamentos, la metodología MOKA, y el sistema KSS con el fin de comprender las bases metodológicas utilizadas al respecto en la aplicación web que se aporta en esta tesis.

2.3.1 Definiciones en relación al conocimiento

Con el objetivo de poder entender algunos términos en las explicaciones posteriores, se muestran una serie de definiciones (Kitamura, 2003):

- **Conocimiento**: Conjunto de datos que modelan de forma estructurada la experiencia que se tiene sobre un cierto dominio o que surgen de interpretar los datos básicos. Por el contrario, la información es el conjunto de datos básicos, sin interpretar, que se usan como entrada del sistema.
- **Gestión de conocimiento**: Se puede considerar como el proceso de integrar la información sobre un dominio concreto, extraer el sentido de la información que se encuentra incompleta y actualizarla. Los sistemas que realizan gestión del conocimiento necesitan diferentes tipos de conocimiento que no suelen estar disponibles en bases de datos y otras fuentes clásicas de información como es el conocimiento sobre los objetos en un entorno y posibles relaciones entre ellos y el conocimiento difícil de representar de manera sencilla, como intencionalidad, causalidad, objetivos, información temporal, conocimiento que para los humanos es de sentido común.
- **Formalización del conocimiento**: La formalización del conocimiento consiste en la transformación de la información en conocimiento, interpretando el conjunto de datos para que explícito y gestionable por un sistema informático.
- **Inferencia de conocimiento**: Es la obtención de nuevo conocimiento que a priori se desconoce desde información previa incluida en el dominio de trabajo. Para ello es necesario un lenguaje que permita el tratamiento del conocimiento.
- **Transmisión del conocimiento**: Consiste en la comunicación del conocimiento adquirido sobre un dominio de un sistema a otro.
- **Persistencia del conocimiento**: Es el conocimiento almacenado previamente sobre un dominio para poder conocer el modo de funcionamiento de éste.

2.3.2 Definición de KBE

La KBE surge como solución de los KBS en su aplicación al ámbito de la ingeniería del diseño debido a dos problemas principalmente: la complejidad de las metodologías existentes, tales como CommonKADS (Sistema de documentación y adquisición de conocimiento común y soporte al diseño y análisis de los

sistemas basados en el conocimiento, del inglés *Common Knowledge Acquisition Documentation System and KBS Analysis and Design Support*); y las formas de adquisición de conocimiento de las que disponía (Studer, 1998).

La KBE es una disciplina basada en los sistemas CAx (Asistidos por Ordenador, del inglés *Computer Aided*) y los KBS con diversas definiciones y funciones dependiendo del contexto de su aplicación (Penoyer, 2000). En el ámbito de la ingeniería del diseño, la KBE se puede definir como el desarrollo de una estructura, a partir de la cual se puede implementar un diseño automático, haciendo uso del conocimiento experto sobre el ciclo de vida de un producto (Vidal and Mulet, 2006). Igualmente, la KBE debe ser capaz de resolver problemas de diseño concretos y a partir del conocimiento adquirido en forma de modelos solucionar cuestiones que puedan surgir en ámbitos similares. (Ammar-Khodja, 2008).

Dentro de las diversas metodologías que surgieron para los sistemas KBE cabe destacar MOKA (Stokes, 2001). La metodología MOKA se propuso para tratar los aspectos metodológicos durante el desarrollo de los sistemas KBE. Fue resultado del proyecto MOKA ESPRIT cuyo objetivo era crear una metodología estándar para el desarrollo y mantenimiento de aplicaciones KBE siendo una de sus principales ventajas la clara distinción entre el modelo formal e informal separando el proceso de adquisición del conocimiento de la integración del conocimiento (Stokes, 2001).

No obstante, a pesar de los estudios llevados a cabo para la aplicación formal del conocimiento en el proceso de diseño, se han creado pocas soluciones eficientes de sistemas KBE que permitan integrar la aplicación del conocimiento con las funciones modernas de los sistemas CAx. Este tipo de aplicaciones se considerarían sistemas abiertos, independientes de cualquier otra aplicación (Vidal and Mulet, 2006) aunque hay un grupo de sistemas CAx que tienen la posibilidad real de representar y almacenar el conocimiento en forma de funciones listas para usar. La principal base para su aplicación es la conexión perfectamente establecida entre el conocimiento obtenido de procesos particulares a través de reglas, entidades estructurales y funcionales del producto de diseño y sus restricciones (Skarka, 2007).

Además, en el ámbito de las PYMEs surgen una serie de inconvenientes en su aplicación (Lovett 2000):

- La falta de personal con experiencia en KBE es un factor clave pues para el desarrollo de nuevos sistemas es necesaria la ayuda de una organización externa.
- El limitado número de personas con quien el desarrollador puede trabajar en una PYME puede presentar dificultades. Por ejemplo, que el conocimiento sobre un aspecto de un sistema KBE provenga de una única fuente de información. Esto puede conducir a errores, malentendidos y desinformación.
- Las pequeñas compañías tienen una economía más limitada que sus grandes competidores. Consecuentemente, no pueden afrontar la dedicación a demasiados proyectos especulativos.

Y otro de los problemas existentes deviene porque los usuarios finales no se suelen involucrar en el desarrollo del sistema experto. En los casos en que el experto de dominio y el usuario final no es necesariamente la misma persona, al experto del dominio usualmente, pero no siempre, le es realmente complicado conocer los requerimientos de los usuarios del sistema (Fasth, 2000).

Igualmente en las tecnologías KBE se puede encontrar una falta de métodos integrados para la identificación, obtención y almacenamiento del conocimiento en el sistema, para poder ser usado de una manera formal (Skarka, 2007). Estos aspectos se podrían concretar en que el uso de estas tecnologías tiene una serie de limitaciones ya que imposibilitan una reutilización y compartición del conocimiento; se halla una la falta de conocimiento común a partir del cual se pueda crear una base de conocimiento y el limitado éxito de metodologías para la extracción de éste (Mizoguchi, 2003). En el campo de la ingeniería del diseño, cada vez se está prestando una mayor atención en el desarrollo de ontologías[q] como una posible solución de las deficiencias mencionadas anteriormente (Montiel-Posoda, 2011).

[q] El apartado referido a las ontologías se trata en profundidad en el apartado 2.4.

En el siguiente apartado se trata la base tecnológica de un KBE, incidiendo en los paquetes de desarrollo de programación y concretamente en los lenguajes de programación *Visual Basic* y Java.

2.3.3 MOKA

MOKA describe en términos de reglas, procesos, técnicas de modelado y definiciones, las etapas necesarias para la especificación de sistemas KBE. Además provee un marco tanto para la captura como para la representación del conocimiento. El citado marco trabaja a dos niveles: nivel informal y nivel formal. El primero es relativamente simple y orientado para representar y formalizar el conocimiento en un lenguaje que sea entendido por los expertos sin ser especialistas en lenguajes formales. La ventaja de este nivel es que hace posible la validación del conocimiento adquirido (Stokes, 2001).

El segundo nivel es más formal y su objetivo es representar y almacenar el conocimiento de una forma codificada con el fin de conectarlo a nivel informático.

Dos aspectos que diferencian MOKA de otras metodologías similares es la estrategia de desarrollo entre los diferentes roles y los formularios ICARE (Ilustración, restricción, actividad, regla, entidad, del inglés *Illustration, Constraint, Activity, Rule, Entity*), que logran definir el dominio de información.

2.3.3.1 MOKA y los roles del diseñador

Con el fin de facilitar la comunicación en el sistema, la metodología MOKA identifica los siguientes roles en el desarrollo de aplicaciones KBE: expertos, ingenieros del conocimiento, ingenieros de software y usuarios finales.

Los papeles del ingeniero del conocimiento y experto toman una relevancia fundamental en la implantación de la metodología MOKA dentro de una aplicación de KBE. Los expertos (también denominados expertos del dominio) son aquellas personas que suministran la principal fuente de conocimiento mediante especificaciones sobre el diseño del producto. Los ingenieros del conocimiento son los que organizan de forma apropiada la captura de este conocimiento así como proponen las actividades para su formalización. Los desarrolladores de software son los responsables de desarrollar el software basado en los requisitos proporcionados por los ingenieros del conocimiento.

2.3.3.2 Los formularios ICARE

El modelo informal de MOKA está compuesto por una serie de formularios enlazados que crean un marco para almacenar las unidades de conocimiento. Estos formularios son cinco:

- ***Illustration* (Ilustración)**

Las ilustraciones, formularios-I, se emplean para completar la información sobre los otros formularios, incluir históricos o referencias a diseños similares, pruebas, explicaciones complejas, entre otros. Normalmente se emplean para incluir esquemas, diseños o gráficos relacionados con el formulario al cual está enlazado.

- ***Constraint* (Restricción)**

Las restricciones, formularios-C, están relacionadas con las entidades y se emplean para capturar y almacenar conocimiento sobre limitaciones del producto o de algún paso del diseño del producto.

- ***Activity* (Actividad)**

Las actividades, formularios-A, describen los procesos de diseño requeridos para el producto en cuestión.

- ***Rule* (Regla)**

Las reglas, formularios-R, tienen varios propósitos, desde generar la salida de la entrada (cálculos, métodos básicos...), como filtro o diagnóstico o para seleccionar las actividades determinadas a través de un proceso.

- ***Entity* (Entidad)**

Las entidades, formularios-E, describen objetos del dominio de conocimiento del producto que se está diseñando como pueden ser componentes, ensamblajes, partes y características. Estos objetos pueden ser físicos o conceptos abstractos.

2.3.3.3 MOKA en la Web 3.0

Actualmente, se puede afirmar que MOKA sigue siendo una metodología vigente en la transición de los sistemas KBE a la web 3.0 o web semántica. Un ejemplo de ello puede ser las investigaciones realizadas por Skarka (2007), Moreno (2007), Ammar-Khodja (2008) o Torres (2013).

A partir del MOKA, Moreno (2007) desarrolla KSS, arquitectura para la captura y adquisición del conocimiento de los procesos de diseño de producto que permite la integración del diseñador en la KBE. Además de esto, incorpora un desarrollo ontológico que permite asistir al diseñador en el proceso de diseño del producto. La herramienta está pensada para que los diseñadores la utilicen de forma autónoma dentro de su puesto de trabajo, conectándose de forma remota a un servidor de base de datos centralizada que albergará las bases de conocimiento que estarán a disposición del mismo. No obstante, pese a las características del proyecto, éste se quedó en fase de prototipo y nunca se probó su usabilidad a nivel práctico.

Otro autor que también utiliza la metodología MOKA y en particular su modelo de conocimiento informal para la representación del conocimiento en un software CAE (Ingeniería asistida por ordenador, del inglés *Computer Aided Engineering*) es Skarka (2007). Para ello formaliza la información a través de una ontología escrita en OWL (Lenguaje de ontologías web, del inglés *Web Ontology Language*) y desarrollada en Protégé (editor libre y abierto de ontologías) con el fin de disponer de un conocimiento coherente, verificado y estructurado. Sin embargo, la ontología creada es muy simple y no establece inferencias de información.

Ammar-Khodja (2008) presenta una implementación del proceso de estimación del conocimiento basado en el enriquecimiento de la metodología MOKA para apoyar la integración del conocimiento del proceso de planificación de partes mecánicas en un sistema CAD. El objetivo es ayudar a los diferentes actores a trabajar colaborativamente proponiendo una visión de referencia del dominio, del contexto y de los objetivos asumiendo que eso les ayudará a tomar una mejor toma de decisiones. Sin embargo, amplía los formularios de MOKA a dos ámbitos más: recurso y función ya que el caso de estudio que propone no es un producto final sino un proceso. No obstante, considera los formularios ICARE como una ontología en sí misma, para el cual representa el conocimiento mediante árboles y diagramas sin llegar a verificar mediante un lenguaje ontológico, como podría ser OWL, la consistencia de la ontología y la posibilidad posterior de la inferencia de información adicional.

De la misma manera Torres (2013) integra el modelo MOKA en la plataforma CATIA para el diseño de una placa de presión de un embrague de fricción y de una aeronave. Sin embargo, no llega a integrar la información en un sistema de formalización del conocimiento para verificar la consistencia de los datos recogidos, como podría ser una ontología y por tanto, tampoco se puede plantear una posible inferencia del conocimiento adquirido.

2.3.4 KSS

Mediante el modelo informal que ofrece la metodología MOKA, KSS proporciona una interfaz gráfica que permite al diseñador la captura y representación del conocimiento del producto (Moreno, 2007).

Dicha interfaz gráfica del KSS permite dar de alta, modificar y eliminar formularios ICARE, respetando el estándar que sugiere MOKA. De la misma forma, establece las distintas relaciones que puedan existir entre ellos y visualizaciones en forma de grafo así como el orden de creación de los mismos. Y se incorpora también un histórico de formularios, que también de forma gráfica, permite al diseñador visionar cuál ha sido el orden en el que se han ido cumplimentando o editando los formularios.

KSS tiene como unidad de trabajo los denominados *proyectos de diseño*, que no son más que agrupaciones lógicas de formularios ICARE, formularios todos ellos relacionados con un proyecto de diseño en concreto. Un formulario ICARE puede incluirse a más de un proyecto de diseño, de forma que se reutiliza el conocimiento entre los diferentes proyectos disponibles.

Otra utilidad que incorpora KSS es la posibilidad de visionar la ontología de dominio que se incluye en la que se representa un proyecto concreto, permitiendo al diseñador tomar decisiones del proceso de diseño y del diseño del producto mediante su empleo. Asimismo, KSS proporciona también otra interfaz que

permite la edición de la ontología, así como una herramienta para importar y exportar ontologías en formato OWL de una forma rápida y sencilla.

KSS mantiene también la estructura diferenciada de roles que se constata en MOKA (Capítulo 2.3.3.1): los expertos e ingenieros del conocimiento son los encargados de mantener y validar la base del conocimiento, esto es, tanto la ontología genérica como los formularios ICARE; los diseñadores, asistidos por la ontología y por los formularios, van tomando decisiones sobre los problemas de diseño que van apareciendo para alcanzar una solución de diseño en el menor tiempo y coste posible.

Como desventajas cabe indicar que el KSS depende de la información suministrada directamente al sistema, sin estar enlazado con ningún software externo. Además, sólo es posible la lectura de la ontología de dominio sin posibilidad de modificarla directamente.

En el siguiente apartado se indica cómo la gestión del conocimiento en la fase de diseño de productos se influencia por la aparición de Internet.

2.3.5 La gestión del conocimiento en la Web 2.0

En la evolución de la gestión del conocimiento de los KBE, surge un aporte radical que transforma tanto el apartado profesional como el social: Internet. Inicialmente conocida como Web 1.0, pues la información relativa a una empresa o una persona que se provee es completamente estática al conformarse por planos virtuales. Evoluciona a través de la inclusión de contenido audiovisual y ahora se transforma en la archiconocida Web 2.0 o también conocida como web social, que centra el conocimiento en la interacción entre sus usuarios.

El término Web 2.0 hace referencia a una segunda evolución de la Web, basada fundamentalmente en dos aspectos: Web como plataforma de acceso a los distintos servicios y en la participación activa de los usuarios como fuente de contenido e información. Web 2.0 no es precisamente una tecnología, representa una actitud, una forma de desarrollar nuevas aplicaciones Web (Oreilly, 2007).

Algunas de las características principales de las aplicaciones Web 2.0:

- **Web como plataforma**: servicios accesibles vía web.
- **Participación:** la participación de los individuos de forma activa es la razón de existencia de muchos nuevos servicios.
- **Contenido generado por el usuario:** la información generada, publicada y compartida por los individuos hace que surjan nuevos servicios basados principalmente en ese tipo de contenidos
- **Software social:** herramientas que basan su existencia en las necesidades o fines de comunicación de las personas y que normalmente forman una comunidad con intereses comunes
- **Filtrado colaborativo**: cuando muchos usuarios expresan sus gustos sobre cierto tipo de contenido se pueden crear modelos y predicciones individuales basados en la opinión del colectivo o ciertos grupos de interés
- **Interfaces ricas**: formas avanzadas de que un usuario interactúe con una aplicación web, ofreciendo nuevas funciones y nuevas posibilidades
- **Folksonomías**: metodología de clasificación en las que los propios usuarios empleen tags o etiquetas de modo descentralizado sobre objetos diversos como fotografías, páginas, vídeos …
- **Movilidad**: posibilidad de acceder a un servicio aunque el usuario cambie de lugar de acceso o de dispositivo
- **Inteligencia colectiva**: ciertas estructuras sociales autorreguladas pueden mostrar comportamientos inteligentes en sí mismas, siendo más eficientes que sus miembros individualmente
- **Sistema flexible de licencias de derechos de autor** para trabajos creativos donde compartir es prioritario frente a restringir o limitar (*Creative Commons*).

Dentro de las tecnologías asociadas a la Web 2.0, se pueden considerar como principales: XHTML y CSS (Hojas de Estilo en Cascada, del inglés *Cascading Style Sheets*); Sindicación de contenidos como RSS (Sindicación realmente simple, del inglés *Really Simple Syndication*) y Atom; AJAX (XML Javascript

asíncrono, del inglés *Asynchronous Javascript XML*); Utilización de RIAs como Flex, Open Lazlo, utilización de SOAP o los *Mashup* (Aplicación Web Híbrida).

Por otro lado, las aplicaciones basadas en la Web 2.0 tienen las siguientes ventajas:

Extrapolación y sindicación absoluta: el hecho de que todas las aplicaciones se realicen sobre web, va a permitir que entre ellas se pueda compartir toda la información. Eso permite a su vez una propagación inmediata de contenido e información lo que permitirá un mejor desarrollo de la estructura de red a la vez que el uso de otras fuentes para desarrollar nuevas aplicaciones.

SaaS (Aplicaciones como servicio, del inglés *Software as a Service*) y no como producto, lo cual elimina el coste de acceso de las PYMES a la tecnología más moderna y permite la realización de una serie de innovaciones constantes.

Ubicuidad: La Web como canal de interoperabilidad por excelencia.

Según O'Reilly (2007) las aplicaciones de la web 2.0 son software que continuamente actualizan su servicio de manera que mejore el uso que le den las personas que lo usan, consumiendo y mezclando datos de recursos múltiples, incluyendo usuarios individuales, mientras proveen sus propios datos y servicios de una forma que se permite la remezcla por otros, creando efectos radiales a través de una arquitectura participativa.

Esto hace que en la web 2.0 converja un rango diverso de tecnologías. Las más utilizadas son los blogs, los wikis, podcasts y redes sociales. Igualmente, nuevas tecnologías van apareciendo constantemente tal como Internet va evolucionando (Chui 09). Por otro lado, se asocia con un fenómeno social, basado en la interacción que se logra a partir de diferentes aplicaciones web, que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en Internet (Web 2.0, 2012). Por ello, las herramientas que disponen de un mayor éxito están siendo aquellas que disponen de una versión gratuita o que ofrecen parte de sus servicios de esa forma.

Sin embargo, en la actualidad, en un mundo globalizado e *hiper-conectado* con la Web 2.0, la KBE ya no puede ofrecer una solución apropiada pues la información necesita estar disponible y ser compartida al instante y la KBE plantea la división del conocimiento base de un sistema en áreas de experiencia donde un experto en el dominio es responsable en mantener ese subdominio, o en muchos casos, realizar la adquisición, mantenimiento y verificación y validación del conocimiento a menudo desempeñado por un ingeniero de conocimiento (Richards 2009). Lo cual dificulta un flujo directo de conocimiento entre los diferentes actores del proceso.

En ese ámbito Richards (2009) ofrece un sistema de colaboración en base a una Wiki para facilitar la captura del conocimiento en acción (*knowledge-in-action*) el cual abarca tanto el tipo de conocimiento explícito como implícito. Este enfoque extiende una técnica denominada MCRDR (reglas de ondas de clasificación múltiple, en inglés *Multiple Classification Ripple Down Rules*) de adquisición del conocimiento basada en casos y en una regla combinada para permitir a los usuarios múltiples una visión colaborativa, definir y refinar una base de conocimiento sobre el tiempo y el espacio. No obstante, el método que aplica necesita ser validado y le falta ser extendido al ámbito de la ingeniería ontológica.

El paradigma de la Web 2.0 modifica la gestión de la información al encontrarse ésta globalizada a la vez que dispersa. Ante esa situación, las empresas deben aprovechar las características de la red y adaptarse a los nuevos recursos disponibles. De ahí surge el concepto de *Enterprise 2.0 (empresa 2.0)* a partir de las nuevas formas de entender la empresa y su relación con el cliente en conexión con la web social emergente.

En la *empresa 2.0* la palabra "cliente" desaparece, pues la relación que se crea con la empresa se basa en sentimientos, en el trato que se puede ofrecer a cada usuario (Kemsley, 2010). Otro aspecto fundamental es el tratamiento de la privacidad de la información pues las personas implicadas en el ciclo de vida de un producto quieren "confiar" en éste y para ello quieren conocer la mayor cantidad de información posible. A este hecho se une la característica de que todo producto está siempre en fase beta, es decir, en proceso de mejora continua y a la vez, como ejemplo, las PYMEs 2.0 dedicadas al sector TIC (Tecnologías de la Información y Comunicación) ofrecen productos muy específicos cuyo negocio se basa en el servicio que ofrecen más que en la venta del producto en sí, conocido con el nombre de SaaS.

Igualmente, en la conjunción entre la evolución formal de la KBE y la referida a Internet, con el fin de simplificar los laboriosos algoritmos en diversos lenguajes para construir los citados sistemas y poder estructurar de manera lógica el ingente conocimiento disponible globalmente, surgen las ontologías, como especificación formal y explícita de una conceptualización compartida (Gruber, 1993). Las ontologías, siendo el sistema más usado de representación del conocimiento, constituyen la base de lo que se conoce como Web 3.0 o web semántica que está empezando a expandirse como modo eficiente de organizar el conocimiento en Internet (Hendler & Berners-Lee, 2010).

En su origen, las ontologías permitieron la inclusión de una nueva forma de gestión del conocimiento de forma jerárquica y lógica que se ha llegado a conocer como Ingeniería Ontológica (Mizoguchi, 2003). Esta formalización permite una claridad, eliminación de fallos estructurales y de la redundancia de información como también la inferencia de nuevo conocimiento a partir de la generación de reglas lógicas, es decir, nuevas relaciones obvias a nivel formal pero de nivel complejo a nivel de comprensión humana.

Por tanto, bajo el planteamiento de la situación de una PyME cuyo objetivo es el rediseño de productos, donde la detección de mejoras, nuevas necesidades y percepciones del usuario son signos claros que el diseñador debe atender para cumplir la tarea asignada, la deslocalización de los recursos, costes, globalización y formalización de la información se vuelven elementos necesarios a considerar. Cabe plantearse si se pueden aunar las principales ventajas de cada una de las etapas planteadas minimizando los inconvenientes, adaptándose a la Enterprise 2.0 y generar un sistema de compartición de conocimiento útil, eficiente y óptimo.

2.3.6 Evolución del KBE a la Ingeniería Ontológica

La Ingeniería Ontológica se puede definir como una metodología de investigación la cual proporciona el diseño racional de una base de conocimiento, del núcleo conceptual del tema de interés y las restricciones semánticas de los conceptos utilizados junto a teorías sofisticadas y tecnologías que permiten la acumulación del conocimiento el cual es imprescindible para el procesado del conocimiento en el mundo real (Mizoguchi, 2003; Fernández-López, 2010).

Las ontologías se pueden basar en una sola taxonomía o en varias siendo conceptos y relaciones que se organizan jerárquicamente y cuyos conceptos pueden ser ordenados como clases e instancias (Gómez-Pérez, 2004). Y por ello, la estructura de una ontología debe estar basada en una taxonomía que tenga en cuenta el modelado de un sistema basado en ciertas descripciones funcionales (Garbacz, 2006). De este modo, se han modelado una gran diversidad de metodologías.

Hacia los modelos en que el conocimiento está disponible en un entorno global, se debe ser consciente de que éste está distribuido por lo que se deben aplicar una serie de reglas. Así, una herramienta de gestión del conocimiento 2.0 requiere (Wang, 2004):

1. Construir un repositorio compartido el cual logra una comprensión común y misma estructura con un medio de averiguar en qué proyectos o problemas se están trabajando (proveyendo oportunidades para compartir y reutilizar);
2. Permitir a la gente crear contenido en su propio modo usando sus propios términos y conceptos;
3. Desarrollar un mapa de conocimiento *top-down* en la forma de una ontología o mapa conceptual para asistir a la gente para definir y estructurar sus propios conceptos con la referencia mayor posible;
4. Permitir secciones de conocimiento para emerger de abajo a arriba (bottom-up) o en sentido inverso (top-down) donde sea apropiado de una manera adecuada, inmediata y espontánea.
5. Apoyar un rango de niveles de experiencia, vistas del conocimiento y derechos de acceso;
6. Proveer un proceso de revisión en los cuales los usuarios pueden indicar su aprobación o desacuerdo;
7. Testear y mantener el contacto de la consistencia entre todos los elementos del sistema de conocimiento y notificar a los usuarios cuando aparecen los conflictos;
8. Ser compatible con un amplio rango de sistemas existentes y fuentes de conocimiento;

9. Proveer un ciclo de mantenimiento del conocimiento estructurado intuitivo, simple y ya estructurado;
10. Apoyar una propiedad primaria y gestión del conocimiento del dominio por usuarios del dominio, no por terceras partes como puede ser un ingeniero del conocimiento.
11. Ser configurable para cada empresa permitiendo cambios en la ontología, en el flujo de trabajo de adquisición del conocimiento y la interfaz del usuario sobre la vida del sistema.

Finalmente, en cualquier sistema de conocimiento es crucial siempre ser consciente de la usabilidad del programa creado. Esta afirmación es de gran importancia pues si no se aprecia una interfaz clara por parte de los usuarios finales o es difícil de usar, no importará cuán inteligente sea el sistema de conocimiento, porque fallará como sistema para ser usado regularmente (Fasth 2000).

2.4 Ontologías

En esta sección se describen los antecedentes en el campo de las ontologías referidos al ámbito de la semántica en un entorno informático ya que uno de los aportes fundamentales de esta tesis es una ontología.

Primero se indica una visión general del concepto de la web semántica o web 3.0. A continuación, se muestra un resumen de las tecnologías anteriores que realizaron las primeras investigaciones sobre la compartición eficiente de datos. A partir de dicho resumen, se trata en profundidad el XML (lenguaje de marcas extensible, en inglés *eXtensible Markup Language*) y su rol como sintaxis fundacional. A partir de ahí, se introduce el RDF (marco de descripción de recursos, en inglés *Resource Description Framework*) y su función como un mecanismo de enlace en la descripción de recursos. Después, se explica la definición y el rol de las ontologías así como su lenguaje de marcas: OWL. Por último, se discute el SWRL (lenguaje de reglas de la Web Semántica, en inglés *Semantic Web Rule Language*).

2.4.1 Definiciones

De la misma forma que en el capítulo anterior, se muestran una serie de definiciones para entender mejor las explicaciones posteriores (Suárez-Figueroa, 2011):

- **Clase:** También llamada concepto, en un sentido amplio de la palabra. Son el centro de la mayoría de las ontologías. Describen conceptos de un dominio de conocimiento determinado usualmente organizadas en taxonomías a través de las cuales se pueden aplicar mecanismos de inherencia. Se interpretan como conjuntos que contienen instancias.
- **Relaciones:** Representan un tipo de asociación entre conceptos de un dominio. Se definen formalmente como cualquier subconjunto de un producto de n conjuntos. Las ontologías generalmente contienen relaciones binarias. El primer argumento es conocido como el *dominio* de la relación, y el segundo argumento como su *rango*. Pueden ser propiedades de objeto, o extrínsecas, o propiedades de tipo de datos, o intrínsecas. Pueden tener diferentes roles y rangos según el ámbito de aplicación. El rango es la instancia dentro de una clase que "recibe" una propiedad, considerándose como *objeto* de la propiedad. Y el *rol* indica la cualidad específica de la propiedad.
- **Axiomas formales**: De acuerdo a Gruber (1993), sirven para modelar sentencias que son siempre verdad. Se utilizan normalmente para representar aquel conocimiento que no puede ser formalmente definido por los otros componentes. Además, los axiomas formales sirven para verificar la consistencia de la ontología o la consistencia del conocimiento guardado en una base de conocimiento. Los axiomas formales son muy útiles para la inferencia de nuevo conocimiento.
- **Instancia:** Representan objetos determinados en el dominio en el cual se trabaja. También recibe el nombre de individuo.
- **Ontología:** Es una descripción explícita y formal de clases en un dominio de discurso, con propiedades de cada clase describiendo varias características y atributos de la clase, y con restricciones sobre dichas propiedades. En términos prácticos, desarrollar una ontología incluye: definir clases en la ontología, organizar las clases en una jerarquía taxonómica, definir propiedades

y describir valores permitidos para dichas propiedades, y llenar los valores de las propiedades para las instancias (Studer et al., 1998).

2.4.2 La Web Semántica o Web 3.0

La visión original de Tim Berners-Lee's sobre la WWW iba más allá de la actual incluyendo meta-datos, es decir, código de máquina interpretable (W3C, 1992). Esta era una visión de la Web Semántica como esquema común que permite la compartición y reutilización de datos a través diferentes aplicaciones, empresas o comunidades. En otras palabras, la Web Semántica o Web 3.0 es la WWW con capacidades de inferencia. La finalidad de la Web Semántica no es sólo crear tanto unas aplicaciones eficaces, como también plasmar datos eficaces (Daconta, 2003).

La información puede ser más eficaz a través del uso de tecnologías semánticas, tales como mapas conceptuales u ontologías. En tales mapas conceptuales y ontologías, los ordenadores encontrarían el significado a los datos semánticos a partir del seguimiento de los hipervínculos hacia definiciones de términos clave y reglas para el razonamiento sobre ellas de manera lógica. Este conjunto de documentos y datos enlazados a través de hipervínculos promovería el desarrollo de servicios automatizados y agentes de software pues el mapeado de términos colocaría a documentos y datos en un contexto específico (Berners-Lee, 2001).

El problema a la hora de desarrollar tareas inteligentes, tal como podrían ser los servicios web o la compartición de modelos en ingeniería de diseño, es que deben recaer en lenguajes de servidores y tecnologías que desarrollen una lógica empresarial. El objetivo de la Web Semántica es proveer un mecanismo a través del cual los ordenadores puedan desarrollar inferencias en documentos y datos con una menor intervención humana. Con la Web Semántica, los ordenadores pueden hacer inferencias basadas en vocabularios con hipervínculos que definen de manera explícita conceptos usados en documentos. Estas inferencias de información permitirían a los ordenadores ir más allá de los simples análisis lingüísticos utilizados por parte de los motores de búsqueda actuales en la WWW (Alesso, 2004).

En la actualidad, el avance de la Web Semántica está ya proveyendo de una cantidad de información útil para diferentes aplicaciones. Permite también un amplio desarrollo de las ontologías explotando el acceso de conocimiento a larga escala. Además se ha convertido además en el paradigma predominante para los sistemas basados en el conocimiento (Suárez-Figueroa, 2011).

El núcleo de la Web Semántica se halla en el uso de sintaxis utilizando URIs (identificadores uniformes de recursos, del inglés *Uniform Resource Identifier*), que enlazan recursos y términos entre documentos. El significado de ello es que los conceptos no son sólo palabras dentro de documentos, sino recursos enlazados que sólo proveen el contexto para los conceptos pertenecientes al documento. La estructura enlazada provee el mecanismo básico para que los ordenadores infieran hechos sobre un documento, incluso si dichos hechos no están creados de manera explícita por el autor. La Figura 7 representa la típica pirámide de los lenguajes semánticos de marcas recomendados por el W3C mostrando los diferentes lenguajes semánticos.

A partir de los modelos de conocimiento dentro de la ingeniería de diseño que se puedan construir para la Web Semántica, los ingenieros serían capaces de compartir el conocimiento de productos de manera más efectiva y fiable. Si los miembros de una cadena de suministro usaran diferentes vocabularios para describir sus datos, las ontologías proveerían un mecanismo para que el ordenador discerniera la equivalencia de términos. Esto implicaría que dichos modelos pudieran ser compartidos con casi ninguna intervención humana y con menor pérdida de información durante el proceso de transferencia (Kim, 2006).

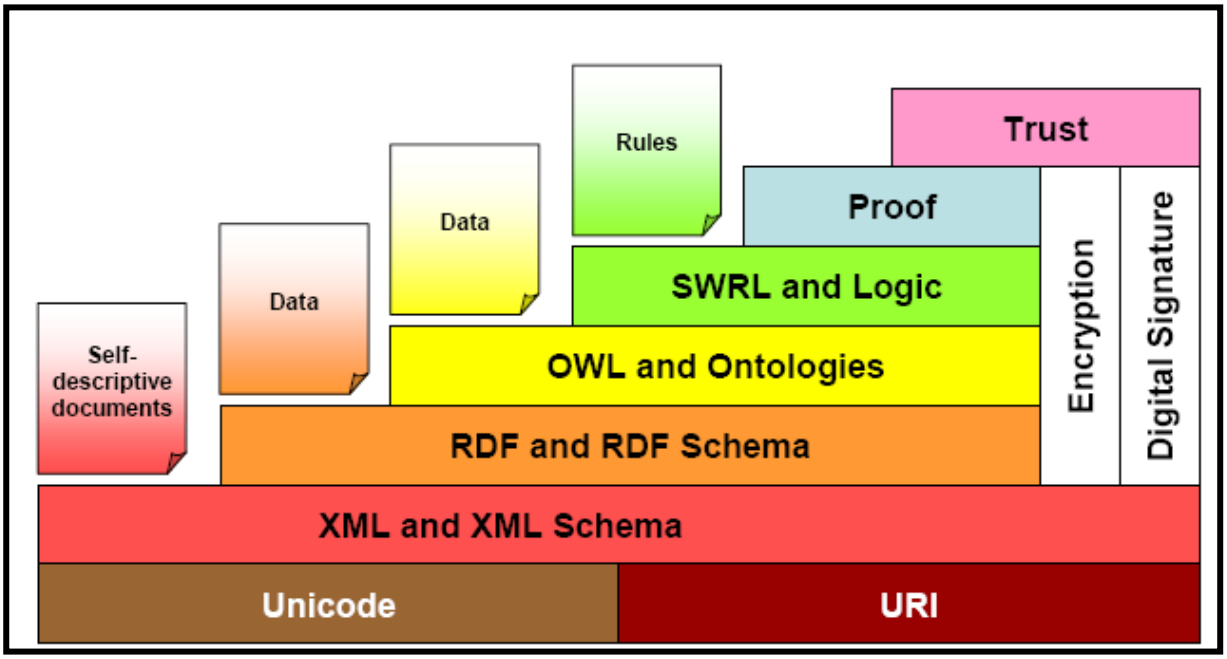


Figura 7: Pirámide de lenguajes de la Web Semántica

2.4.3 Lenguajes ontológicos principales

Los lenguajes ontológicos existentes tienen diferentes mecanismos de expresividad e inferencia, ya que los paradigmas de representación del conocimiento bajo todos esos lenguajes son diversos. Además, una de las decisiones claves para tomar en el proceso de desarrollo de ontologías es la selección del lenguaje (o conjunto de lenguajes) en el cual la ontología será implementada.

A continuación, se presenta un resumen introductorio de las especificaciones actuales para lenguajes de ontologías desarrollados en el ámbito de la actividad de web semántica de la W3C (2004): XML, RDF, OWL, OWL 2 y SWRL.

2.4.3.1 Esquema XML

El esquema XML es un estándar abierto que permite intercambiar datos entre aplicaciones y bases de datos sobre la WWW. Es un puente interoperable entre distintos marcos, tales como *.NET* de Microsoft y *J2EE* de Sun (Alesso, 2004).

Ya que es posible que los datos sean usados en una variedad de aplicaciones y programas, es necesario, por ello, que se represente en una estructura genérica. Así, XML permite a los usuarios añadir una estructura arbitraria en sus documentos a través del uso de *tags*, pero no muestran ninguna información al respecto sobre lo que la estructura significa. Quedan claras sus ventajas ya que permiten a los usuarios definir sus propias etiquetas añadiendo así una estructura a sus datos. Sin embargo, el programa receptor debe conocer al inicio cómo han sido usadas las etiquetas en el documento para así codificar de manera apropiada el programa. Así, el programa receptor debe conocer el contexto de los datos antes que pueda empezar a utilizarlo. Dicha situación muestra que mientras XML habilita el intercambio de datos, carece de semántica. Por tanto, los datos formateados en XML pueden utilizarse para un propósito determinado.

El esquema XML es un conjunto de reglas que documentos XML referenciados deben seguir. Un esquema XML define la estructura y el orden de documentos XML así como los tipos de datos que se permite utilizar. Estos son metadatos para el documento XML que son usados para comprobar si el documento es válido antes de la transmisión y que la compartición de datos tome lugar. Si un documento XML es válido, los programadores pueden con seguridad analizar la sintaxis del documento XML con éxito.

Si dos organizaciones sólo desean compartir datos entre sus aplicaciones utilizando la sintaxis XML, se requiere que como primer paso se pongan de acuerdo en una estructura de datos común (conjunto y orden de las etiquetas XML) antes que el intercambio de datos y el procesamiento de información comiencen. Si

las organizaciones “evolucionan” sus datos a RDF, las dos aplicaciones pueden compartir datos usando dos estructuras de datos diferentes, pero manteniendo una transferencia de datos con éxito haciendo uso del concepto de equivalencias.

2.4.4 Esquema RDF

RDF es un lenguaje de marcas que permite la interconexión de recursos distribuidos y que se construye a raíz de XML. A pesar que XML representa los datos en una estructura de árbol, RDF los representa en la forma de un número variable de tripletes (Figura 8). Un triplete contiene un sujeto, predicado y objeto, y así mismo, cada triplete representa una sentencia o hecho sobre algo. Resumiendo, los tripletes RDF plantean sentencias sobre recursos donde éstos son cualquier tipo de ítems con una URI o dirección asociada. De esta forma, un recurso se enlaza vía alguna propiedad (que puede ser otro recurso) a un valor específico o a otro recurso. La Figura 9 representa en un gráfico la estructura básica de un triplete RDF.

La estructura en triplete de RDF es simple, además de muy eficaz para enlazar recursos distribuidos. Mientras que en XML los datos deben residir en una estructura en forma de árbol, los tripletes RDF pueden residir en cualquier orden, manteniendo siempre la coherencia incluso en un triplete individual (Tabla 10).

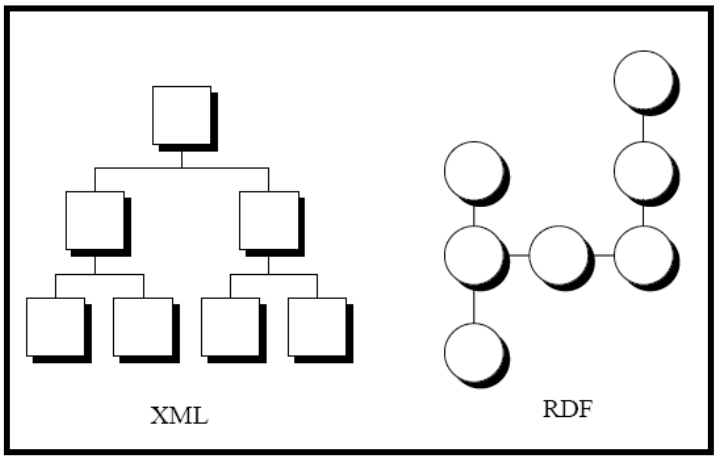


Figura 8: Estructura de XML vs estructura RDF

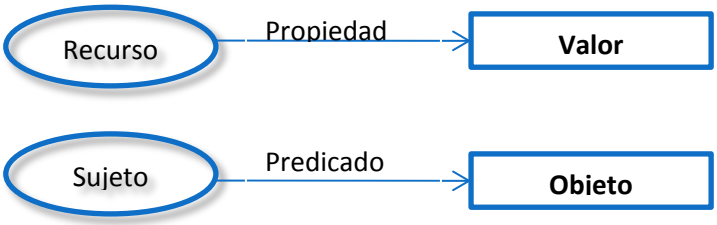


Figura 9: Estructura de un triplete RDF

Las siguientes cuatro sentencias son elementos a tener en cuenta sobre los tripletes RDF (Alesso, 2004):

- Cada triplete RDF está formado por un sujeto, predicado y objeto.
- Cada triplete RDF es un hecho completo y único.
- Cada triplete RDF puede ser unido a otros tripletes RDF, pero aún mantiene su propio valor único a pesar de su complejidad.
- Cada triplete RDF es una tupla [r]triple cuyo sujeto es una referencia o nulo, el predicado es una referencia, y el objeto es una referencia, nulo o valor literal.

[r]Tupla: Es una secuencia ordenada de objetos, esto es, una lista con un número limitado de objetos. Las tuplas se emplean para describir objetos que son capaces de ser descompuestos en un cierto número de componentes.

RDF provee de una sintaxis para la representación de la semántica básica de datos de una manera interoperable estandarizada. Para representar el significado de los datos, el W3C desarrolló el RDFS (esquema RDF, del inglés *RDF Schema*). El propósito del RDFS es definir relaciones de clases entre vocabularios. Y así, tales relaciones de clases facilitan la búsqueda de segmentos específicos de datos y, de esta forma, le proporcionan a los datos un contexto. Las clases le dicen al usuario o a la aplicación qué información está representada. Es posible para un elemento ser miembro de muchas clases y no por ello mantener una relación jerárquica. Esta capacidad hace a RDFS un lenguaje mínimo para la representación de ontologías.

Aunque RDF y RDFS son pilares básicos para la definición de un lenguaje de marcas de la Web Semántica, carecen de poder expresivo. Por ejemplo, no pueden definir:

- Las propiedades de otras propiedades. Ej.: Una propiedad de la clase “Padre” es “tiene hijos”, pero dicha propiedad no puede definir otra propiedad a su vez.
- Condiciones necesarias y suficientes para una pertenencia de una clase, es decir, restricciones que definen.
- Clases equivalentes y disjuntas[5].

Además, las únicas restricciones expresables son las restricciones de dominio y de rango en las propiedades. Como resultado, la semántica se especifica de forma imprecisa. Cada URI define un nuevo recurso. Así, hay muchos recursos que representan el mismo concepto. Las especificaciones RDF no definen un mecanismo para establecer la equivalencia de los recursos, es decir, que los recursos múltiples representan el mismo mapeado conceptual. Esto deja lugar a capas más elevadas del lenguaje, tales como OWL (Alesso, 2004).

En resumen, RDF y RDFS se construyen a partir de XML y proveen un medio para enlazar recursos dispares los cuales pueden representar cualquier cosa con una URI. RDF va más allá de la estructura en árbol de XML y representa sentencias en forma de tripletas que se entrelazan con otras. RDFS aplica una semántica básica a RDF a través de la definición de clases y propiedades para recursos. Aunque esto es el punto inicial para el lenguaje de una ontología, le falta expresividad para definir las condiciones para que las clases sean equivalentes y disjuntas.

Con respecto a la ontología presentada en este trabajo, RDF provee el mecanismo de enlace para segmentos de datos en un documento. Por ejemplo, los componentes de una estructura particular pueden residir en diferentes bases de datos distribuidas de manera dispersa. El uso de RDF puede enlazar diferentes componentes en un modelo pudiendo también ser usado para enlazar entidades geométricas en una estructura.

2.4.5 Las ontologías y el lenguaje OWL

Un problema clave en lograr la interoperabilidad en la WWW es reconocer cuando dos segmentos de datos se relacionan con la misma entidad, incluso si diferentes terminologías se están usando. OWL puede ser utilizado para cubrir esta diferencia terminológica (Alesso, 2004). Por ejemplo, un programa cuyo propósito es comparar información a partir de dos bases de datos debe conocer que dos términos están referenciando a la misma entidad. Idealizando, el programa debe tener un modo para descubrir significados comunes para todo aquello que la base de datos encuentra.

Una ontología se puede describir como una especificación explícita de una conceptualización compartida, la cual se puede basar en una taxonomía o en axiomas (Gruber, 1993). Una conceptualización se refiere a una perspectiva abstracta y simplificada del conocimiento que tenemos del "mundo", y que por un motivo determinado se quiere representar. Especifica un vocabulario, incluyendo términos claves, sus interconexiones semánticas, y algunas reglas de inferencia. Además, una ontología es un modelo computacional de algunos elementos que a menudo se representan en la forma de una red semántica (un gráfico cuyos nodos representan conceptos u objetos individuales y cuyos arcos representan relaciones o

[5]Una clase disjunta es aquella que no puede tener ninguna instancia en común con otra clase.

asociaciones entre los conceptos). Desde un punto de vista colaborativo, una ontología es un acuerdo sobre una conceptualización compartida que incluye marcos para el modelado de un dominio del conocimiento. Resumiendo, las ontologías facilitan "conversaciones" entre aplicaciones de software a través de Internet (Singh, 2005). Las ontologías proveen un medio para programas dispersos para encontrar significados comunes entre diferentes vocabularios.

Las ontologías típicas consisten en una taxonomía de clases y un conjunto de reglas de inferencia. Una regla puede describir una conclusión que parte de una premisa o también puede ser una sentencia procesada por un mecanismo o un proceso de cálculo que puede inferirse a partir de una regla genérica dada. A través de la utilización de procesadores de reglas se pueden hacer inferencias sobre conocimiento existente, es decir, nuevo conocimiento derivado de uno ya existente. Permitiendo a los ordenadores inferir nuevo conocimiento es posible liberar de ese conocimiento a los usuarios humanos ya que no necesitarán procesar esa información. De todas formas, el ordenador en realidad no entiende la información tal como lo hacen los humanos, sin embargo, a través del lenguaje OWL los ordenadores la pueden manipular de forma eficiente.

El lenguaje recomendado de marcas para definir ontologías para su uso en la Web Semántica es OWL (W3C-OWL 2004). OWL creció a partir de un lenguaje ontológico previo, DAML+OIL (*DARPA Agent Markup Language + Ontology Inference Layer*). A grandes rasgos, OWL es una estandarización de esfuerzos de representación de conocimiento previos usados para extender RDF con un vocabulario mayor para que se pudieran crear relaciones lógicas más enriquecedoras entre conceptos.

Las ontologías OWL difieren de los esquemas XML, ya que las ontologías OWL pueden modelar el conocimiento en un dominio donde los esquemas XML son sólo sintaxis y formato para mensajes. Las sentencias XML por sí solas no permiten plantear conclusiones sobre otras sentencias XML. Las ontologías OWL difieren de los esquemas RDF en que las ontologías OWL son mucho más expresivas. OWL está construido a partir de RDF y así utiliza todas las construcciones de RDF/RDFS, pero OWL además añade vocabulario adicional en orden de representar relaciones lógicas, tales como propiedades de otras propiedades, condiciones de pertenencia a clases, restricciones cardinales, y clases equivalentes/disjuntas.

OWL por sí mismo sólo es un formato de datos que utiliza reglas y lógica descriptiva para relacionar términos. Con el fin de utilizar los datos en OWL para una aplicación debe construirse para aceptar datos en formato OWL. Sin embargo, ya que los datos en el formato OWL están bien marcados, es posible inferir nuevo conocimiento desde el conocimiento ya existente sin intervención humana. Para hacer esto, se utiliza un motor de inferencia (*inference engine*) para derivar nuevos hechos. La Figura 10 presenta la arquitectura básica de una aplicación que utilice datos explícitos en OWL como hechos inferidos.

Con el fin llegar a determinar la reusabilidad de una ontología se hace uso de las métricas. El tema de reutilización de ontologías es un tema que se aborda en Lozano Tello (2004), mediante la aplicación de una métrica denominada *Ontometric*. Ésta métrica toma como base diferentes criterios: versión del lenguaje, madurez de la metodología, objetivos del proyecto, entre otros y está destinada para ingenieros o gestores de proyectos. En el siguiente apartado se trata con profundidad las características de dicha métrica.

Según Fortuna et al. (2011) las aproximaciones de las evaluaciones de ontologías se encuentran dentro de las siguientes categorías:

a) Aquellas basadas en la comparación de la ontología con una ontología estándar de referencia, (comparación a nivel léxico y conceptual);

b) Basadas en el uso de la ontología dentro de una aplicación para la posterior evaluación de resultados. Para ello se considera su expresividad y el recuento de axiomas de las clases, propiedades de objeto y datos, individuos y anotaciones;

c) Basadas en comparaciones con una fuente de datos (una colección de documentos) acerca del dominio que cubre la ontología; y,

d) Basadas en criterios de expertos en ontologías.

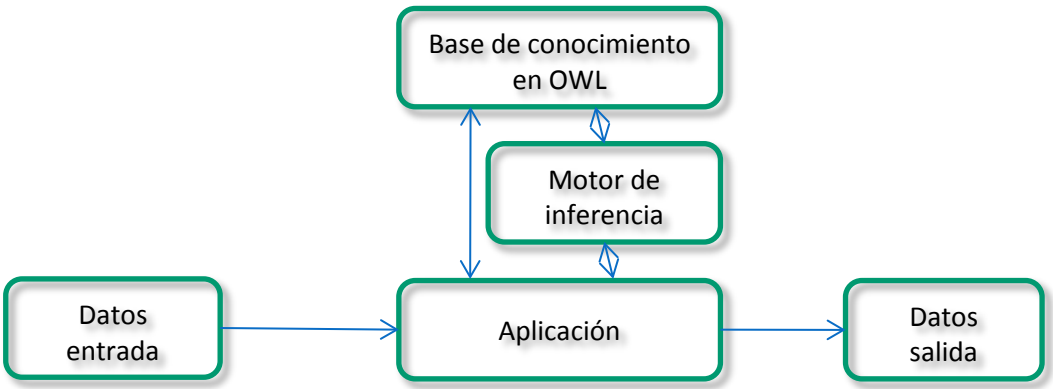


Figura 10: Arquitectura básica de una aplicación utilizando OWL (Alesso, 2004).

2.4.5.1 Ontometric

Ontometric es una métrica realizada por Lozano Tello (2004) para medir la idoneidad de ontologías existentes con respecto al sistema donde se utiliza. Pese a su antigüedad, se mantiene completamente vigente y se establecido como uno de las métricas más fiables (Sicilia, 2012). Se basa principalmente en una serie de factores estructurados en las siguientes dimensiones:

- **Herramienta**: Uno de los procesos que más tiempo consume al intentar usar una ontología es la adaptación de los términos de la ontología a las necesidades del proyecto. No es usual que al utilizar una ontología coincidan al cien por cien los conceptos, sus definiciones formales, las relaciones, los atributos, los axiomas, etc. Los entornos software pueden ser de gran ayuda para realizar la adaptación, y el usuario debe estudiarlos con detenimiento, pues favorecerán o dificultarán la integración de la ontología en el proyecto. En gran parte, la posibilidad de utilizar estas herramientas software está supeditada, principalmente, al lenguaje en el que está implementada la ontología. Estos tratan: las prestaciones del entorno, aspectos de visualización de la ontología, de edición, de interacción con otras aplicaciones software, metodológicos, cooperativos, de traducción y de integración. En la Tabla 4 se indican los ítems resultantes de cada uno de los factores.
- **Lenguaje**: El lenguaje en el que se encuentra implementada la ontología constituye también uno de los principales criterios que deben ser examinados para seleccionar una ontología. Por un lado, si para el proyecto van a necesitarse métodos de resolución de problemas que realicen inferencias con los conocimientos almacenados, es fundamental que el lenguaje tenga las capacidades deductivas requeridas. Además, si se necesitan añadir nuevos términos a la ontología o aumentar la granularidad de los existentes, el usuario se debe asegurarse que el lenguaje de implementación tiene ese potencial expresivo. Eso hace que se caracterice esta dimensión por conceptos/instancias/hechos/peticiones, atributos, facetas, relaciones, taxonomías, axiomas y reglas productivas. En la Tabla 5 se indican los ítems resultantes de cada uno de los factores.
- **Contenido**: En esta dimensión se consideran los términos que incluye la ontología y cómo éstos están organizados. En el proceso de valoración de las ontologías es fundamental examinar esta dimensión, para valorar qué grado de eficiencia existe entre el contenido que necesita el proyecto y el que contiene la ontología analizada. En este apartado se incluyen, además de las características descriptivas sobre el contenido, características que se refieren a los conceptos identificados en la ontología, la taxonomía que forma la jerarquía de conceptos, las relaciones[t] (y funciones) existentes, y los axiomas. En la Tabla 6 se indican los ítems resultantes de cada uno de los factores.
- **Metodología**: El desarrollo de ontologías debe ser realizado mediante metodologías adecuadas, diseñadas para este fin, del mismo modo que se siguen metodologías específicas para el desarrollo de cualquier componente software. Una ontología desarrollada con una metodología será más fácil

[t] En este caso hay un ítem en el factor de relaciones que es la aridad entendido como el número de relaciones y complejidad de argumentos necesarios para que se pueda constituir la ontología.

de modificar que otra desarrollada sin ella. Por el contrario, una metodología con actividades difíciles de seguir o mal definidas puede retrasar considerablemente el proceso de adaptación de la ontología al nuevo sistema. Eso hace que esta dimensión esté caracterizada por la precisión, su usabilidad y la madurez de la ontología. En la Tabla 7 se indican los ítems resultantes de cada uno de los factores.

Tabla 4: Características relativas a la dimensión "Herramienta" (Lozano Tello, 2004)

FACTORES	VALORES
HABILIDADES	**(muy bajo, bajo, medio, alto, muy alto)**
Uso local	(no soportado, soportado)
Uso en red	(no soportado, soportado)
Uso basado en Internet	(no soportado, soportado)
Claridad de la interfaz del usuario	(muy bajo, bajo, medio, alto, muy alto)
Tiempo de respuesta	(muy bajo, bajo, medio, alto, muy alto)
Fiabilidad	(muy bajo, bajo, medio, alto, muy alto)
VISUALIZACIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
El navegador muestra la completa información de los términos	(no soportado, soportado)
El navegador permite la selección a nivel de detalle	(no soportado, soportado)
El navegador muestra la taxonomía	(no soportado, soportado)
El navegador muestra relaciones *ad hoc*	(no soportado, soportado)
EDICIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
La herramienta construye en el mismo lenguaje	(no soportado, soportado)
La herramienta permite la edición en cualquier momento	(no soportado, soportado)
La herramienta muestra la taxonomía gráficamente	(no soportado, soportado)
La herramienta permite la definición de nuevas relaciones	(no soportado, soportado)
INTERACCIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
La herramienta permite un uso independiente	(no soportado, soportado)
La herramienta suministra interfaces de acceso	(no soportado, soportado)
Existe documentación sobre las interfaces de acceso	(muy bajo, bajo, medio, alto, muy alto)
Las interfaces de acceso son *Open Source*	(no soportado, soportado)
Documentación sobre interfaces de programación de acceso	(muy bajo, bajo, medio, alto, muy alto)
ASPECTOS METODOLÓGICOS	**(muy bajo, bajo, medio, alto, muy alto)**
La herramienta soporta el ciclo de vida completo	(no soportado, soportado)
La herramienta apoya actividades de desarrollo importantes	(no soportado, soportado)
La herramienta suministra documentación sobre productos creados	(no soportado, soportado)
La herramienta chequea la consistencia	(no soportado, soportado)
ASPECTOS COOPERATIVOS	**(muy bajo, bajo, medio, alto, muy alto)**
La herramienta crea grupos de trabajo	(no soportado, soportado)
La herramienta permite trabajo simultáneo	(no soportado, soportado)
La herramienta mira ontologías editadas	(no soportado, soportado)
La herramienta mira términos editados	(no soportado, soportado)
La herramienta notifica los cambios al grupo	(no soportado, soportado)
La herramienta identifica los cambios del usuario	(no soportado, soportado)
TRADUCCIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
La herramienta importa datos de otras lenguas	(no soportado, soportado)
La herramienta importa datos de otras lenguas de marcado	(no soportado, soportado)
La herramienta exporta a otras lenguas	(no soportado, soportado)
La herramienta exporta a otras lenguas de marcado	(no soportado, soportado)
Las traducciones pierden un mínimo de semántica	(no soportado, soportado)
La traducción es supervisada	(no soportado, soportado)
INTEGRACIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
Facilidad de integración	(muy bajo, bajo, medio, alto, muy alto)
Dificultad de referir a nuevos términos	(muy bajo, bajo, medio, alto, muy alto)
La herramienta permite la selección de términos para su integración	(no soportado, soportado)
La herramienta chequea la consistencia en su integración o fusión	(no soportado, soportado)
Asistencia para fusión manual	(no soportado, soportado)
Fusión semi-automática	(no soportado, soportado)

- **Costes**: En proyectos software, normalmente de índole comercial, el estudio de requisitos debe contemplar las estimaciones de gastos que va a ocasionar el desarrollo del proyecto. En los proyectos que usan ontologías, uno de los apartados que debería reflejar el estudio son las estimaciones de gastos relacionados específicamente con la elección de una determinada ontología. Las características que se recogen en esta dimensión se contabilizarán de forma aditiva, y el valor obtenido guardará una proporción inversa con la idoneidad en esta dimensión; es decir, independientemente del presupuesto con el que cuente el proyecto, a mayor coste de uso, menos idónea será la ontología en esta dimensión. Esta dimensión está caracterizada por tanto, por el uso de las licencias de la ontología, costes estimados del hardware y software, coste de las interfaces de acceso, y el uso de las licencias. Para su evaluación se utiliza una escala de 5 puntos.

Tabla 5: Características relativas a la dimensión "Lenguaje" (Lozano Tello, 2004)

FACTORES	VALORES
CONCEPTOS/INSTANCIAS/HECHOS/PETICIONES	**(muy bajo, bajo, medio, alto, muy alto)**
Permite las instancias de clases	(no soportado, soportado)
Tiene metaclases	(no soportado, soportado)
Puede definir clases sin metaclases	(no soportado, soportado)
Claridad de la interfaz del usuario	(no soportado, soportado)
Permite hechos	(no soportado, soportado)
Permite peticiones	(no soportado, soportado)
ATRIBUTOS	**(muy bajo, bajo, medio, alto, muy alto)**
Puede definir atributos de clases	(no soportado, soportado)
Puede definir atributos de instancias	(no soportado, soportado)
Puede definir atributos locales	(no soportado, soportado)
Puede definir atributos globales	(no soportado, soportado)
Puede definir atributos polimórficos	(no soportado, soportado)
Puede definir atributos sobre excepciones	(no soportado, soportado)
FACETAS	**(muy bajo, bajo, medio, alto, muy alto)**
Tiene valores de atributos por defecto	(no soportado, soportado)
Tiene tipos de atributos	(no soportado, soportado)
Puede definir la cardinalidad de los atributos	(no soportado, soportado)
Permite la definición de conocimiento sobre el procedimiento	(no soportado, soportado)
Permite nuevas facetas	(no soportado, soportado)
RELACIONES	**(muy bajo, bajo, medio, alto, muy alto)**
Permite la definición de funciones	(no soportado, soportado)
Hay relaciones arbitrarias enésimas	(no soportado, soportado)
Se permite la definición de relaciones *ad hoc*	(no soportado, soportado)
Se puede restringir el tipo en las relaciones	(no soportado, soportado)
Se puede restringir el valor en las relaciones	(no soportado, soportado)
Se definen las operaciones a realizar	(no soportado, soportado)
Se pueden declarar las propiedades en las relaciones	(no soportado, soportado)
TAXONOMÍAS	**(muy bajo, bajo, medio, alto, muy alto)**
Contienen la relación: *Subclase de*.	(no soportado, soportado)
Contienen la relación: *No es subclase de*.	(no soportado, soportado)
Múltiples subclases de clases	(no soportado, soportado)
Múltiples instancias derivadas de otras instancias	(no soportado, soportado)
AXIOMAS	**(muy bajo, bajo, medio, alto, muy alto)**
Permite los axiomas embebidos en términos	(no soportado, soportado)
Permite axiomas independientes	(no soportado, soportado)
Permite axiomas en lógica de primer orden	(no soportado, soportado)
Permite axiomas en lógica de segundo orden	(no soportado, soportado)
REGLAS DE PRODUCCIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
Permite reglas disyuntivas en las reglas de producción	(no soportado, soportado)
Permite reglas conjuntivas en las reglas de producción	(no soportado, soportado)
Cada regla tiene definido un mecanismo de encadenamiento	(no soportado, soportado)
Cada regla tiene definida una prioridad	(no soportado, soportado)
Procedimientos en los resultados en la reglas de producción	(no soportado, soportado)

Tabla 6: Características relativas a la dimensión "Contenido" (Lozano Tello, 2004)

FACTORES	VALORES
CONCEPTOS	**(muy bajo, bajo, medio, alto, muy alto)**
Conceptos esenciales	(muy bajo, bajo, medio, alto, muy alto)
Conceptos esenciales en los niveles superiores	(muy bajo, bajo, medio, alto, muy alto)
Los conceptos se describen apropiadamente en lenguaje natural	(muy bajo, bajo, medio, alto, muy alto)
La especificación formal de los conceptos coincide con el lenguaje natural	(muy bajo, bajo, medio, alto, muy alto)
Los atributos definen los conceptos	(muy bajo, bajo, medio, alto, muy alto)
Número de conceptos	(muy bajo, bajo, medio, alto, muy alto)
RELACIONES	**(muy bajo, bajo, medio, alto, muy alto)**
Relaciones esenciales	(muy bajo, bajo, medio, alto, muy alto)
Las relaciones corresponden conceptos apropiados	(muy bajo, bajo, medio, alto, muy alto)
La especificación formal de las relaciones coincide con el lenguaje natural	(muy bajo, bajo, medio, alto, muy alto)
Aridad especificada	(muy bajo, bajo, medio, alto, muy alto)
Propiedades formales de las relaciones	(muy bajo, bajo, medio, alto, muy alto)
Número de relaciones	(muy bajo, bajo, medio, alto, muy alto)
TAXONOMÍA	**(muy bajo, bajo, medio, alto, muy alto)**
Perspectivas diversas	(muy bajo, bajo, medio, alto, muy alto)
Apropiadas *no-subclases-de*	(muy bajo, bajo, medio, alto, muy alto)
Particiones exhaustivas apropiadas	(muy bajo, bajo, medio, alto, muy alto)
Particiones disjuntas apropiadas	(muy bajo, bajo, medio, alto, muy alto)
Máxima profundidad	(muy bajo, bajo, medio, alto, muy alto)
Media de subclases	(muy bajo, bajo, medio, alto, muy alto)
AXIOMAS	**(muy bajo, bajo, medio, alto, muy alto)**
Los axiomas resuelven cuestiones	(muy bajo, bajo, medio, alto, muy alto)
Los axiomas infieren conocimiento	(muy bajo, bajo, medio, alto, muy alto)
Los axiomas verifican la consistencia	(muy bajo, bajo, medio, alto, muy alto)
Los axiomas no se enlazan a conceptos	(muy bajo, bajo, medio, alto, muy alto)
Número de axiomas	(muy bajo, bajo, medio, alto, muy alto)

Tabla 7: Características relativas a la dimensión "Metodología" (Lozano Tello, 2004)

FACTORES	VALORES
PRECISIÓN	**(muy bajo, bajo, medio, alto, muy alto)**
Delimitación de fases	(muy bajo, bajo, medio, alto, muy alto)
Especificación de actividades por fases	(muy bajo, bajo, medio, alto, muy alto)
Especificación de personal por fases	(muy bajo, bajo, medio, alto, muy alto)
Especificación de técnicas por fases	(muy bajo, bajo, medio, alto, muy alto)
Especificación de productos finales por fases	(muy bajo, bajo, medio, alto, muy alto)
USABILIDAD	**(muy bajo, bajo, medio, alto, muy alto)**
Claridad de actividades y descripción de técnicas	(muy bajo, bajo, medio, alto, muy alto)
Calidad de manuales	(muy bajo, bajo, medio, alto, muy alto)
Manuales con ejemplos completos	(muy bajo, bajo, medio, alto, muy alto)
MADUREZ	**(muy bajo, bajo, medio, alto, muy alto)**
Número de ontologías desarrolladas	(muy bajo, bajo, medio, alto, muy alto)
Número de diferentes dominios	(muy bajo, bajo, medio, alto, muy alto)
Importancia de las ontologías desarrolladas	(muy bajo, bajo, medio, alto, muy alto)

2.4.6 OWL 2

OWL 2 (Motik, 2009) es una extensión y revisión de OWL que añade nueva funcionalidad con respecto a OWL, alguna de las nuevas características incrementan su estructura sintáctica mientras otras ofrecen nueva expresividad. OWL 2 incluye tres perfiles diferentes igual que el lenguaje OWL.

OWL 2 proporciona dos formas alternativas de asignar significado a las ontologías OWL 2: la semántica directa que asigna significado directamente a las estructuras ontológicas y la semántica basada en RDF que asigna el significado directamente a los gráficos RDF.

2.4.7 Hipótesis de un mundo abierto

Un aspecto que se debe comentar, con el fin de comprender la aplicación de las ontologías basadas en el lenguaje OWL, es la OWA (hipótesis de un mundo abierto, del inglés *Open World Assumption*).

Dicha hipótesis proveniente del campo de la lógica formal indica que el valor real de una sentencia es independiente de si cualquier observador o agente *sabe* que es cierto. Es el opuesto de la hipótesis de mundo cerrado el cual mantiene que cualquier sentencia que no se sabe que es cierta, es falsa. La OWA es utilizada en representación del conocimiento para codificar la noción informal que, en general, ningún agente u observador tiene un conocimiento completo, y por ello no se puede aceptar la hipótesis de un mundo cerrado. Aunque el OWA limita los tipos de inferencia y deducciones que un agente puede hacer a sólo aquellas sentencias que el agente sabe que son ciertas.

Desde el punto de vista heurístico, la OWA se aplica cuando se representa el conocimiento en un sistema a la vez que se descubre, y donde no se puede garantizar si se ha descubierto o se descubrirá la información completa. En la OWA, las sentencias sobre el conocimiento que no se incluyen o no se infieren del conocimiento explícito grabado en el sistema pueden considerarse desconocidas, más que ciertas o falsas.

En las ontologías basadas en OWL la ausencia de una sentencia particular significa en principio que la sentencia todavía no es explícita, independiente de si sería cierta o no, y de si creemos (creeríamos) que es (o sería) cierta o no. De forma resumida, desde la ausencia de una sentencia sola, un razonador deductivo no puede inferir que la sentencia sea falsa.

Por tanto, en OWA, la falta de información sobre un hecho no implica que sea falso. Por ejemplo, se plantea que una persona, Cécile, tiene la nacionalidad española. A partir de esa información, no se puede concluir que Cécile tenga la nacionalidad francesa, o que no la tenga. Todo ello admitiendo que el conocimiento sobre el mundo es incompleto. Eso implica que la OWA está relacionada con la naturaleza monotónica [u]de la lógica de primer orden: la adición de nueva información nunca niega una conclusión previa. Es decir, en el ejemplo indicado, si se descubre a posteriori que Cécile tiene también la nacionalidad francesa, no cambia la conclusión realizada con anterioridad.

Este hecho queda representado en el significado de las paráfrasis que se puede extraer a partir de las sentencias en OWL (Rector, 2004) (Tabla 8).

Tabla 8: Paráfrasis de OWL

Definición de OWL	**Paráfrasis**
Class (Thing partial ...	Todas las cosas ...
Class (Thing complete parent...	Una cosa es cualquier *padre*[v]que ...
(añadir en todas las descripciones y definiciones[w])	...entre otras cosas...
allValuesFrom	sólo
someValuesFrom	algunos
and	ambos... y también
not(... and ...)	No todos de/ no ambos…y también
not (... or ...)	Ni... ni...
someValuesFrom not	Tiene algunos... que no son...

[u] La naturaleza monotónica en lógica implica que agregar una información a una teoría nunca produce una reducción de su conjunto de consecuencias. Así la lógica de primer orden es de naturaleza monotónica.
[v]Padre en el sentido de una clasificación jerárquica.
[w] Definición que se deriva a partir de la hipótesis de mundo abierto.

Definición de OWL	**Paráfrasis**
not (someValuesFrom ...)	no tiene ... ningún
AllValuesFrom not	Tiene...no... / Sólo tiene...que no son...
not (allValuesFrom ...)	no tiene ... sólo
subclassOf(A , B)	A implica B

2.4.8 Lenguaje SWRL

Mientras la expresividad de OWL promueve la actividad de la Web Semántica, también tiene sus limitaciones. Para resolverlo, se creó el lenguaje SWRL para extender la lógica descriptiva y los axiomas de OWL con reglas de Horn[x]. A partir de este lenguaje se constituyen una serie de reglas que pueden ser utilizadas para inferir nuevo conocimiento desde bases de conocimiento OWL ya existentes (O'Connor, 2005). SWRL se basa en una combinación de los sublenguajes de OWL: OWL DL y OWL Lite con sublenguajes del lenguaje de reglas de marcas (W3C-SWRL, 2004). Muchas de las limitaciones de OWL conciernen al desarrollo de las propiedades OWL.

Las reglas SWRL tienen la forma de antecedente (cuerpo) y consecuente (cabeza). El significado básico de la regla puede plantearse como "si las condiciones que el antecedente adquiera, entonces son las condiciones que el consecuente debe también cumplir·". La cabeza y el cuerpo consisten en una conjunción de uno o más segmentos. Las reglas SWRL razonan sobre individuos (instancias de clases) de OWL, a partir de clases OWL y propiedades. Por ejemplo, una regla SWRL expresa la siguiente sentencia: *Una persona que tiene un sobrino implica que un sobrino tiene una tía*. Esa frase requiere capturar los conceptos de "persona", "sobrino" y "tía" en OWL (O'Connor, 2005).

Golbreich et al. (2005) indican que las reglas SWRL permiten obtener:

- Dependencias entre las propiedades de ontologías.
- Dependencias entre ontologías y otros predicados de dominios.
- Búsquedas.

La necesidad de reglas es por tanto dependiente del uso de la aplicación final utilizando la base de conocimiento OWL (Cuenca-Grau, 2012). Para poder aplicar SWRL en Protégé se utiliza una variación de dicho lenguaje denominado SQWRL (Lenguaje de reglas mejorado para la búsqueda semántica de la web del inglés *Semantic Query-Enhanced Web Rule Language*) que se puede ejecutar en base a Jess, un motor de reglas para la plataforma Java, y a un razonador semántico como por ejemplo Racer Pro (O'Connor, 2005; Protégé, 2012).

2.5 Ontologías en el campo de la Ingeniería del Diseño

Es de especial relevancia mostrar en esta revisión del estado del arte las ontologías existentes en el campo de Ingeniería del Diseño, pues el aporte de la ontología OntoFaBES (Capítulo 3.2) se basa en la necesidad de cubrir un ámbito poco cubierto anteriormente. Para ello se va a realizar una descripción de las ontologías existentes en este ámbito haciendo especial mención a la metaontología DOLCE y la taxonomía OntoRFB, pues en base a ellas se ha constituido OntoFaBES.

2.5.1 Descripción

En esta sección se introducen aquellas ontologías que se encuentran en el dominio de la ingeniería de diseño. De manera general, las ontologías se pueden organizar por su concepto (Gómez-Pérez, 2004):

[x]Las reglas de Horn son reglas basadas en la cláusula de Horn, que un tipo de regla con una serie de premisas, y un único consecuente. Ej.: "A es hija de B si A es mujer y B es padre de A".

- **Metaontologías:** Describen conceptos muy generales (p.ej. sustancia, tangible, intangible) y proveen de nociones generales sobre las cuales se enlazan todos los términos básicos en las ontologías existentes. El objetivo es disponer de un gran número de ontologías disponibles sobre una metaontología.
- **Ontologías de dominio:** Son reutilizables en un dominio específico dado (ingeniería, fabricación, diseño, etc.). Estas ontologías proveen vocabularios sobre conceptos bajo un dominio y sus relaciones, donde determinadas actividades se reproducen bajo el citado dominio, mientras se rigen por teorías y principios elementales.
- **Ontologías de tarea:** Describen el vocabulario relacionado con una tarea específica o actividad (p.ej. diagnóstico, planificación). Proveen un vocabulario sistemático de los términos utilizados para resolver problemas asociados con tareas que pueden o no pertenecer al mismo dominio.
- **Ontologías de aplicaciones** o dependientes de las aplicaciones: Contienen todas las definiciones necesarias para modelar el conocimiento requerido para una aplicación particular. Las ontologías de aplicaciones a menudo extienden y especializan el vocabulario de las ontologías de dominio y de tarea para unas aplicaciones dadas.

En el campo de la ingeniería de diseño, los tipos de ontologías más predominantes son las ontologías de tarea y ontologías de dominio (Cuenca-Grau, 2008). La Tabla 9 contiene una revisión del estado del arte en el ámbito de la ingeniería de diseño relacionado con el campo de las ontologías, basado parcialmente en el trabajo de Kitamura (2004), donde se realiza una descripción de las ontologías más destacadas, indicando el tipo de ontología y el lenguaje utilizado.

Una de las principales ontologías de la Web Semántica, DOLCE, provee un marco de referencia común para ontologías con el objetivo de facilitar el intercambio de información entre ellos. Así, DOLCE se dirige hacia la captura de categorías ontológicas subrayando el lenguaje natural y el sentido común humano (Masolo, 2003).

Gero, quien impulsó uno de los primeros estudios sobre el marco FBS, establece las bases para el modelado computacional de procesos como apoyo al proceso de diseño basado en el marco FBS (Gero 2002, 2004; Christopher, 2010; Pourmohamadi, 2011). Así desarrolló el término "*FBS Ontology*" para referirse a su modelo.

Tabla 9: Ontologías en el dominio de la ingeniería del diseño

Abreviatura	Origen	Denominación	Tipo de ontología	Lenguaje de la ontología
AsD	Kim et al. (2006)	Assembly Design Ontology	Ontología de tarea	OWL/SWRL
DiDeas II	Liu et Bason (2007)	Distributed Design Assistant	Ontología de dominio	OWL
DO	Storga et al. (2005)	Design Ontology	Ontología de dominio	OWL
DOLCE	Masolo et al. (2003)	Descriptive Ontology for Linguistic and Cognitive Engineering	Metaontología	OWL
EDIT	Ahmed et al. (2006)	Engineering Design Integrated Taxonomy	Ontología de dominio	OWL
FB Ontology	Bryant Arnold et al. (2007)	A Function-Based Component Ontology For Systems Design	Ontología de dominio	OWL
Gellish English	Van Rennsen et al. (2007)	A taxonomy of functions on Gellish English	Ontología de tarea	OWL
OntoCAPE	Morbach (2009); Marquatdt et al. (2010)	A Reusable Ontology for Computer-Aided Process Engineering	Metaontología/ Ontología de dominio / Ontología de tarea	OWL

Abreviatura	Origen	Denominación	Tipo de ontología	Lenguaje de la ontología
Ontology-Based Collaborative Design System	Su et al. (2009)	An ontology-Based Collaborative Design System	Ontología de tarea	OWL/SWRL
Ontology-based approach for PLM	Matsokis et al. (2010)	An ontology model of a Product Data and Knowledge Management Semantic Object Model for PLM	Ontología de dominio	OWL
Ontology for CommonKADS	Gobin & Subramanian (2009)	An Owl Ontology for CommonKADS Template Knowledge Models	Ontología de dominio	OWL/SWRL
OntoRFB	Garbacz (2006)	Taxonomy of artefact functions	Ontología de dominio	OWL
OntoSTEP	Krima et al. (2009)	OWL-DL ontology for STEP (*Standard for the Exchange of Product model data*)	Ontología de dominio	OWL
Product configuration knowledge model	Yang et al. (2008)	An ontology-based approach to modeling product configuration	Ontología de dominio	OWL/ SWRL
PDO	Catalano et al. (2008)	Product Design Ontology	Ontología de tarea	OWL
SOFAST®	Kitamura et al. (2003, 2004, 2006)	Non-formal ontology of objects in the FBRL language	Ontología de dominio	Hozo

EDIT (Taxonomía integrada de la ingeniería del diseño, en inglés *Engineering Design Integrated Taxonomy*) (Ahmed, 2006) se constituye a partir de un conjunto de taxonomías integradas y las relaciones entre ellas. Dicha taxonomías se rellenan con instancias, y las relaciones se capturan entre los múltiples conceptos creados. Así, EDIT se establece con el propósito de indexar, buscar y recuperar el conocimiento del diseño (Ahmed, 2006).

La DO (Design Ontology – Ontología de diseño) está relacionada con el trabajo anterior (Ahmed, 2007). Storga (2005) crearon la DO como una potencial descripción formal del conocimiento ingenieril compartido en el dominio de diseño. A partir de ambos trabajos, Ahmed and Storga (Ahmed, 2007) han creado una comparación entre ambos, donde la DO es descrita como una ontología concebida para describir el diseño como un producto. En contraposición, la EDIT establecía el diseño como una actividad, incorporando el proceso tanto como el producto.

Por otro lado, en la evolución de las ontologías en la ingeniería de diseño, Garbacz (2006) expone OntoRFB, una taxonomía de funciones de artefactos, basada en DOLCE y en RFB y analizada desde una perspectiva de la lógica filosófica; la ontología AsD (Kim, 2006) es la primera que hizo uso tanto del lenguaje OWL como del SWRL para el área de fabricación; y la ontología desarrollada por Kitamura et al. (2006) aplicada al apartado industrial, logra desarrollar una ontología de dominio y aplicarla a un ejemplo práctico por medio del programa informático SOFAST®.

Posteriormente, un trabajo muy complejo en el ámbito científico sobre la aplicación de las ontologías al dominio del CAPE (Ingeniería de Procesos Asistidos por Ordenador, del inglés *Computer-Aided Process Engineering*), es OntoCAPE (Morbach, 2009; Marquartdt, 2010): una metaontología genérica, una ontología

de dominio y varias ontologías de tarea para generar patrones para mejorar las prácticas en diseño en varios problemas de modelado.

En otra línea Krima (2009) presenta OntoSTEP, una ontología basada en STEP traduciendo del lenguaje EXPRESS para permitir el razonamiento lógico y la interoperabilidad semántica. Sin embargo, sólo se centran en la geometría propia del producto. Y Matsokis (2010) presenta un método genérico y sistemático para desarrollar ontologías desde modelos existentes basados en PLM (Gestión del Ciclo de Vida del Producto, del inglés *Product Lifecycle Management*) para generar un modelo consistente y soportado lógicamente.

En los últimos años la utilización de OWL y SWRL también se ha ido generalizando: Yang (2008) adopta estos dos lenguajes para modelar el conocimiento relativo a la configuración de productos; Su (2009) y Dutra (2010) lo aplican para generar sendas ontologías basadas en diseño colaborativo; Gobin (2009) emplea esta combinación para convertir un modelo de conocimiento, como es CommonKADS y representar plantillas originarias en ese lenguaje.

2.5.2 DOLCE

Una de las ontologías originarias en la Web Semántica, la metaontología DOLCE, trata de englobar las diferentes categorías ontológicas subrayando el lenguaje natural y el sentido común humano (Figura 11). Una de sus características es proporcionar información explícita para que algunas de sus divisiones puedan ser completada a partir de otras ontologías (Masolo, 2003).

DOLCE se basa en cuatro clases de entidades: *endurant*, *perdurant*, cualidad/atributo (Q*uality*) y abstracto (A*bstract*)[y].

- ***Endurant**:* Son entidades que están presentes en todo momento. Ej. Un trozo de papel.
  Se dividen en *endurant*s físicos (*Physical Endurant*), no-físicos (*Non-physical Endurant*) y en conjuntos donde están unidos ambos: suma arbitraria (*Arbitrary Sum*). Un *endurant* físico es un *endurant* que tiene calidades espaciales. En cambio, un *endurant* no-físico no tiene ninguna calidad espacial.
  La clasificación de los *endurant*s físicos se subdivide en cantidad de materia (*Amount of matter*), características (*feature*), y objetos físicos (*Physical-Object*). Una cantidad de materia se puede definir como *endurant* sin unidad que es mereológicamente invariante[z]. P. ej. Oro, plata, madera o carne.
  Un objeto físico es un *endurant* con unidad, p. ej. Una persona, un coche o un reloj. Asimismo los objetos físicos se pueden dividir en APO (objetos físicos agente, del inglés *Agentive Physical Object*) y NAPO (objetos físicos no-agente, del inglés *Non-Agentive Physical Object*). Los APO son aquellos objetos físicos para los cuales se pueden adscribir intenciones, creencias o deseos. P. ej. Un ser humano. Y los NAPO, los restantes objetos físicos P. ej. La estructura física de un objeto.

[y] Debido a su significado particular y que dichos cuatro términos, especialmente *Endurant* y *Perdurant* no disponen de una traducción literal al castellano se va a mantener en su definición en inglés.

[z] Un elemento es mereológicamente invariante cuando todos los componentes de dicho elemento se mantienen constantes de manera indefinida.

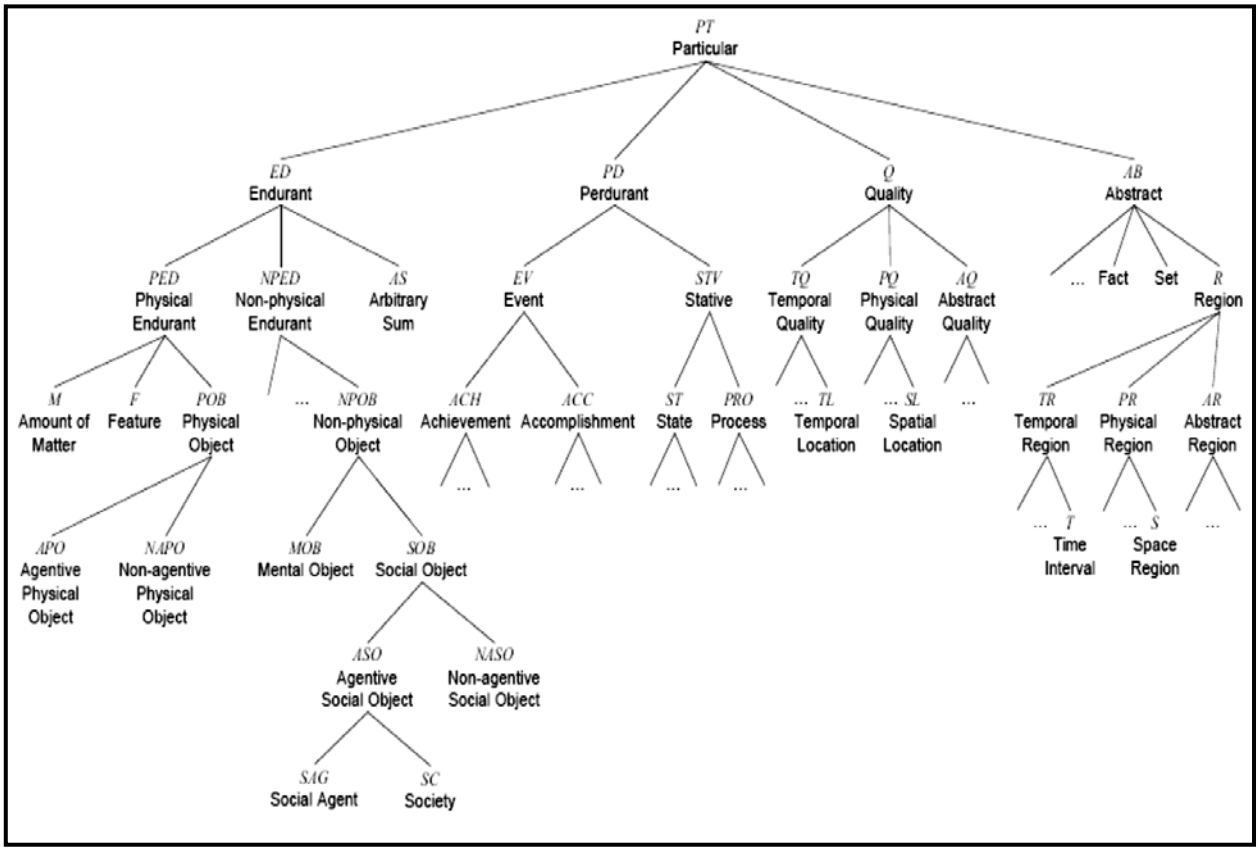


Figura 11: Taxonomía de las categorías básicas de DOLCE (Masolo, 2003)

- ***Perdurant**:* Son entidades que están presentes durante un tiempo determinado. Ej. Una acción.
  Se dividen en eventuales (*event*), entidades dependientes del tiempo, y estacionarios (*stative*), independientes del tiempo. Por ejemplo, el proceso de diseño de un objeto es un *perdurant* eventual y en cambio, estar sentado en una silla, es un *perdurant* estacionario.
  Los *perdurant*s eventuales se subdividen a su vez en cumplimientos (*accomplishment*) y en logros (*achievement*). Así, mientras un logro es instantáneo, un cumplimiento ocupa una fracción de tiempo. Por ejemplo, un cumplimiento sería el proceso de diseño de un objeto y un logro sería la sonrisa del diseñador al ver terminado dicho objeto.
  De la misma forma, los *perdurant*s estacionarios se subdividen en estados (*state*) y en procesos (*process*). Los estados son *endurant*s que se mantienen durante un tiempo indefinido, por ejemplo, la posición de un tornillo en un objeto. Y los procesos no tienen un punto de finalización natural, por ejemplo, el giro de la rueda de un automóvil.

- ***Quality**:* Son entidades que se pueden percibir o medir. Ej. Una medida, un color.
  En DOLCE se indica que una calidad es inherente a una entidad durante todo el intervalo de tiempo en el cual dicha entidad existe. En otras palabras, una entidad tiene una calidad siempre o nunca. Bajo ese razonamiento, una calidad puede considerarse temporal si es inherente a un *perdurant*. Un ejemplo básico en referencia a una calidad temporal es la localización temporal de un *perdurant* (Figura 11).
- ***Abstracto**:* Son entidades que no tienen cualidades/atributos. Ej. El valor del dinero.
  Dentro de la división realizada, cabe comentar el término referido a una región física (*Physical Region*). Según DOLCE, es una región en la cual sólo pueden adherirse las cualidades físicas directamente. P. ej. Peso, módulo de Young o conductividad térmica, entre otras.

A partir de la explicación realizada de DOLCE, se describe a continuación la taxonomía OntoRFB.

2.5.3 OntoRFB

Basado en un estudio exhaustivo de las diferentes teorías de diseño Garbacz (2005) determina que el modelo RFB (Szykman, 1999; Hirtz, 2002) pese a una serie de deficiencias de ambigüedad y precisión como relativas a simplificaciones en su estructura (que posteriormente también comparte Sun (2011)), es el modelo que con mayor rigurosidad se podría estructurar lógicamente como una taxonomía para el dominio del diseño funcional.

Entonces Garbacz (2006) expone OntoRFB, una taxonomía de funciones de artefactos, basada en el modelo RFB y la metaontología DOLCE (Masolo, 2003) y analizada desde una perspectiva de la lógica filosófica con el objetivo de depurar las deficiencias encontradas.

Con el fin de resolver dichas deficiencias, Garbacz hace uso de la DOLCE determinando que la función de un objeto sea un *SoA* (S*tate-of-Affairs*[aa]), es decir, entidad objetiva o no-mental representada por una frase. Término proveniente de la filosofía, se puede traducir como *situación actual* y se define como la combinación de circunstancias aplicadas en una sociedad o grupo en un tiempo particular. Además es compatible con varias teorías filosóficas al respecto de esta noción (Armstrong, 1997; Ingarden, 1967).

Se pueden categorizar tres tipos básicos de SoAs a partir de la taxonomía de DOLCE:

- **SoA i:** Si una entidad tiene una cualidad, entonces el respectivo SoA es del tipo inherencia: *tipo i.* P .ej. Un martillo tiene una masa (como una de sus propiedades).
- **SoA p:** Si un *endurant* participa en un *perdurant*, entonces el respectivo SoA es del tipo participación: *tipo p*. P. ej. Una persona participa en una acción.
- **SoA v:** Si una cualidad tiene un valor que es un *quale*[bb], entonces el respectivo SoA es del tipo valor: *tipo v*. P. ej. La tonalidad particular de un color.

Si el término SoA se lleva al ámbito de las funciones de un objeto, un SoA i no se considera función de un objeto debido a la noción del DOLCE de inherencia atemporal, ya que se establece como premisa que la cualidad de un objeto sea permanente, como el ejemplo mostrado anteriormente del martillo.

Respecto a los dos tipos de SoAs restantes, sí que se consideran que puedan ser funciones, estableciendo la siguiente división:

- **P-funciones** (*Participation-functions*)**:** Basadas en el SoA del tipo p. Se corresponden con las actividades temporales. P. ej. Un interruptor cierra un circuito eléctrico.

  Basándose en la clasificación de *perdurant* realizada en DOLCE, se plantean los cuatro tipos de p-funciones: lograr (*achievement*), cumplir (*accomplishment*), estar (*state*) y procesar (*process*).

- **V-funciones** (*Value-functions*)**:** Basadas en el SoA del tipo v. Se corresponde con las actividades atemporales. P. ej. Un cable conecta el enchufe con el televisor.

  Dos posibles tipos de SoA-v son el SoA Inicial que se corresponde con los valores iniciales de una cualidad y el SoA Terminal con sus valores finales.

De forma similar que en el caso de las p-funciones, se plantean tres tipos de cualidades en base a los términos del DOLCE: localización temporal (*temporal location*), localización espacial (*spatial location*) y energía (*energy*) (Figura 11).

La categoría *conexión topológica* (*Topological Connectedness)* no aplica la cualidad de localización espacial, sino que es un conjunto de distintas cualidades provenientes de DOLCE. Dicha cualidad tiene dos valores opuestos: Sí/No. Cuando adquiere dicha cualidad el valor Sí, el SoA v resultante es una función del tipo *conectar* (*connect*). Y si la cualidad de conexión topológica tiene el valor No, entonces el SoA v es una función del tipo *bifurcar* (*branch*). En los restantes casos al relacionar las SoAs v con la cualidad *localización espacial* y la cualidad energía, se obtiene el término ubicar (*Locate)* y el término "*Energate*" (*energizar*),

[aa] http://plato.stanford.edu/entries/states-of-affairs/

[bb] Un *quale* describe la posición de un atributo individual dentro de un espacio conceptual. Para su comprensión cabe distinguirse entre un *atributo* (p.ej. el color de un tablero), y su *valor* (p.ej. una tonalidad particular de marrón).

respectivamente. De esta forma quedan definidos los cuatro tipos de v-funciones: *Locate, connect, branch, energate* (Tabla 10).

Tabla 10: Taxonomía OntoRFB: p-funciones y v-funciones

	Spatial location	**Topological connectedness**		**Energy**
Achievement	Reach	Touch	Split	Switch
Accomplishment	Channel	Attach	Disjoin	Load
State	***Moor***	Join	Cleave	Conserve
Process	Move	Bind	Carve	Energise
Quale region	Locate	Connect	Branch	Energate

Con el fin de relacionar los cuatro tipos de p-funciones con respecto a las cualidades definidas con anterioridad, se enlaza cada SoA p con un par de SoAs v: el SoA Inicial (*Initial SoA)* y el SoA Terminal (*Terminal SoA*). Para el caso de las cualidades de localización temporal y energía, ambos SoAs v corresponden a un mismo *perdurant*. Y para el caso de la localización temporal, ocurre un caso análogo a la descripción de las v-funciones, creando un término para el SoA Inicial y otro para el SoA Terminal.

De esa forma un SoA del tipo-p puede clasificarse con respecto a una cualidad asociada a él. La taxonomía resultante se puede organizar en forma de tabla (Tabla 10) donde la columna inicial contiene todos los tipos de SoAs p, mientras que la fila inicial especifica los tipos de cualidades asociables con tales SoAs.

Se considera el siguiente ejemplo referido al funcionamiento de una batidora. Concretamente, para que una batidora pueda picar un alimento es necesario que pueda mantenerlo alojado dentro, a partir de lo cual permite formular la siguiente sentencia: "Una batidora conserva una porción de comida". En este caso, según OntoRFB el *endurant*, (la batidora), logra una cierta localización espacial (*spatial location*), manteniendo en un lugar el elemento (la porción de comida). Al ser la actividad temporal y no tiene un punto de finalización natural, el SoA respectivo es del tipo p, y el *perdurant* es mantener (*State*). Ello da como resultado en la taxonomía OntoRFB la función *Moor* (Tabla 10*).*

Sin embargo, cabe indicar una serie de aspectos a mejorar en esta taxonomía:

- No llega a profundizar en las relaciones existentes entre las funciones de una manera exhaustiva para poder generar una ontología de dominio debido a que DOLCE al ser una metaontología, no puede ser específica para un conocimiento concreto.
- Al ser una taxonomía de funciones no trata las relaciones existentes entre funciones, comportamientos y estructura, propios del marco de diseño funcional FBS.
- Al ser una taxonomía no define propiedades ni describe los valores permitidos para dichas propiedades y por tanto, tampoco llena los valores de las propiedades para las instancias que pudieran generarse a partir de las clases de la taxonomía, es decir, no tiene las características de una ontología.

En resumen, OntoRFB propone una taxonomía formalizada a través de conceptos lógicos, lo cual permite eliminar la ambigüedad existente en la taxonomía derivada del modelo RFB para lo cual utiliza la DOLCE. Sin embargo, no llega a profundizar en las características de una ontología ni establece las relaciones relativas al marco FBS al estar basado en flujos y funciones.

2.6 Discusión y conclusiones

En este capítulo se ha realizado una revisión del estado del arte en 4 dominio de conocimiento claves para la posterior comprensión de los aportes de la tesis: el ámbito del diseño funcional, la arquitectura cliente/servidor, la KBE y las ontologías incidiendo concretamente en aquellas que se aplican en el campo de la ingeniería del diseño.

En la revisión realizada respecto a los términos función, comportamiento y estructura, se ha mostrado que es un tema que a nivel de investigación sigue siendo de actualidad (Li, 2012; Crilly, 2013; Kitamura, 2013).

Luego, en el ámbito del marco funcional FBS, se ha indicado que pese a la gran cantidad de trabajos desarrollados en dicho ámbito se percibe una multiplicidad de enfoques sin llegar a un consenso ni una rigurosidad al no estar basado en un razonamiento lógico (Vermaas, 2012). Como una posible solución en el ámbito de los comportamientos se ha mostrado el modelo B-Cube sin embargo indica una serie de deficiencias que no permiten su formalización lógica.

Posteriormente se ha querido explicar todos los términos informáticos de relevancia relativos a la arquitectura cliente/servidor para la posterior comprensión de la aplicación web que aporta esta tesis doctoral y que se presenta en el Capítulo 4. Se ha mostrado razonadamente por qué las RIA y concretamente Adobe Flex permite una funcionalidad muy interesante como aplicaciones web en la interconexión de información entre cliente y servidor en los entornos web. Además se han introducido otros conocimientos informáticos relativos al lenguaje de programación VBA y Java además del lenguaje de programación de base de datos MySQL.

A continuación se introducido aquellos conocimientos relevantes de la KBE para poder comprender su evolución incidiendo en la metodología MOKA de gestión de conocimiento. Se ha indicado que pese a ofrecer un modelado del conocimiento muy estructurado, separando el proceso de adquisición del conocimiento de la integración del conocimiento (Stokes, 2001), el modelado de dicho conocimiento requería un coste demasiado elevado para su aplicación sobretodo en PYMEs que hasta que no se ha desarrollado la ingeniería ontológica no ha mostrado posibles soluciones (Skarka, 2007; Ammar-Khodja, 2008, Montiel-Posoda, 2011; Torres, 2013). Una de ellas, el modelo KSS plantear un enfoque aplicado al diseño funcional y concretamente utilizando el marco FBS para ello. Sin embargo, tal como se ha indicado, no llega a enlazarse con ningún software externo ni posibilita la conexión con Internet y la ontología que utiliza es trivial.

Finalmente, como evolución de la KBE a la ingeniería ontológica, se ha explicado con detalle en qué consiste del concepto de la web semántica o web 3.0. A continuación, la evolución de los lenguajes semánticos remarcando la definición y el rol de las ontologías así como su lenguaje de marcas: OWL y el lenguaje de inferencia de conocimiento SWRL.

Por último se muestran las ontologías que se han desarrollado en el campo de la ingeniería de diseño incidiendo en DOLCE, como una de las metaontologías con mayor rigurosidad en su estructura lógica (Oberle, 2007) y aplicabilidad en este campo además de su relación con la taxonomía OntoRFB. Dicha taxonomía se formaliza a través de conceptos lógicos, lo cual permite eliminar la ambigüedad existente en la taxonomía derivada del modelo RFB siendo una de las taxonomías en el dominio de diseño funcional con una mayor rigurosidad y posibilidades de aplicación en comparación con otros trabajos de mayor complejidad como el de Kitamura (2006). Sin embargo, no llega a profundizar en las relaciones existentes entre las funciones de una manera exhaustiva para poder generar una ontología de dominio ni llega a aplicar el marco FBS incidiendo únicamente en el apartado de las funciones.

Capítulo 3. OntoFaBES

En este capítulo se introduce OntoFaBES, ontología que formaliza el conocimiento sobre un producto con el objetivo de inferir diferentes estructuras de dicho producto a partir de los requerimientos funcionales propuestos por el usuario, con el objetivo de que el conocimiento creado se pueda reutilizar y compartir entre diferentes aplicaciones.

Primero se realiza una descripción en profundidad del marco de diseño funcional FaBES. Para ello se plantean las definiciones de cada una de las capas del citado marco, acompañadas de varios ejemplos que aclaran su organización y labor.

A continuación, se describe la estructura de la ontología OntoFaBES, adaptación del marco FBS como una especificación formal y explícita de una conceptualización compartida del diseño conceptual ante la ausencia de modelos similares en la literatura. El objetivo es disponer de una ontología que pueda formalizar el conocimiento adquirido mediante los formularios ICARE para ser utilizados en el sistema KSS 2.0, aportación mostrada en el Capítulo 4.

OntoFaBES se conforma a partir del marco formal FaBES, teniendo en consideración las características del lenguaje OWL y de Protégé, herramienta informática utilizada para su modelado. Así se indica la definición de cada una de las capas para luego mostrar las clases y propiedades utilizadas. A partir de esta información, se detallan las reglas utilizadas para la inferencia de conocimiento.

3.1 FaBES

El marco FBS estructura el conocimiento sobre un diseño en función, comportamiento y estructura, sin embargo deja pie a una serie de ambigüedades (Garbacz, 2005) que pueden ser resueltas a través de la formalización del conocimiento y la rigurosidad planteada por la ingeniería ontológica que basa sus estructuras en la lógica de la hipótesis abierta. En este capítulo se presenta el modelo FaBES como alternativa a los modelos disponibles en la literatura combinando las ventajas del modelo FBS y el rigor lógico de las ontologías.

3.1.1 Introducción

Según Visser (2009), las características de un diseño están basadas en tres factores: el proceso de diseño, los diseñadores y el artefacto. El proceso de diseño puede ser definido como un proceso de toma de decisiones jerárquicas y los modelos teóricos en este ámbito suelen tener en cuenta uno o dos factores de los citados, pero en raras ocasiones los tres a la hora de plantear un marco teórico que defina la situación de un diseño de un producto.

El esquema FBS tal como se ha indicado en la introducción aborda tanto el proceso de diseño, a través de la definición de los apartados de función y comportamiento como el referido al artefacto, que se puede equiparar al apartado de estructura. Sin embargo, no toma en cuenta la figura de los diseñadores. Y éstos diseñan para otras personas: los “usuarios” del producto realizado pues el comportamiento de un artefacto, al cambiar en el tiempo, puede relacionarse a su impacto en la sociedad a través de la interacción que pueda realizar y para su uso en personas que no necesariamente transforman dicho uso (Andreasen, 2011). Además, los productos actualmente se están conectando cada vez más a servicios, en los cuales los usuarios de éstos cobran más importancia si cabe.

Por otro lado, existe la necesidad de generar un marco basado en el esquema FBS que se pueda trasladar al ámbito ontológico, para lo cual es necesaria una consistencia lógica. Eso proporcionaría una rigurosidad que, tal como se ha comentado en el Aptdo. 2.1.1 no disponen esquemas como el NIST (Garbacz, 2006). Igualmente, el marco FBS propuesto por Gero (2002), denominado *FBS Ontology* es un enfoque que no cumple las condiciones planteadas como ontología pues dicho modelo necesita una revisión y clarificación en algunos aspectos para asegurar su claridad y coherencia lógica (Galle, 2009).

Dicho nuevo enfoque debería representar de una forma explícita, las funciones de un producto o del sistema (el problema), la estructura del producto (solución) y el comportamiento (proceso) interno del

producto. Los objetos se considerarían a través de tres visiones (funcional, comportamental y estructural) y se supone que podrían ser definidas mientras pasan sucesivamente desde una visión a otra. Además, al ser un modelo centrado en el usuario también debería tener en cuenta aquellos aspectos referidos al entorno de una función que el Análisis de Valor o funcional no tratan a la hora de realizar el rediseño de un producto al ser éste un problema abierto (Visser, 2009).

FaBES se estructura como una posible solución ante las necesidades indicadas para este ámbito teórico del diseño funcional. En este apartado del capítulo se introducirá el concepto de A-QB, como elemento necesario para comprender el significado de acción en FaBES y su posterior relación con las capas de función y comportamiento. Posteriormente, se incluyen tres ejemplos de aplicación práctica para observar su aplicación práctica.

3.1.2 Marco teórico de FaBES

FaBES surge como una extensión y refinamiento del modelo B-FES desarrollado por Tor (2002). Este marco teórico une al modelo B-FES una mayor profundización con el fin de poder relacionar el apartado funcional con el estructural de manera racional. Para ello se utiliza como fundamento el marco formal proporcionado por la metaontología DOLCE y las investigaciones realizadas a partir del A-QB (Capítulo 3.1.4).

La Figura 12 muestra las características de FaBES consistente en 3 fases: Acción, que contempla dos capas: función (F_x) y comportamiento (B_x); Entorno de la acción (E_x) y la estructura en la que se considera tanto el objeto de diseño como el agente que realiza la acción del citado diseño (A_x, y P_x). Para facilitar su implementación, la función se relaciona con el comportamiento, el comportamiento con la estructura y el entorno de la acción se asocia de manera implícita entre el comportamiento y la estructura tal como se puede observar en la Figura 13.

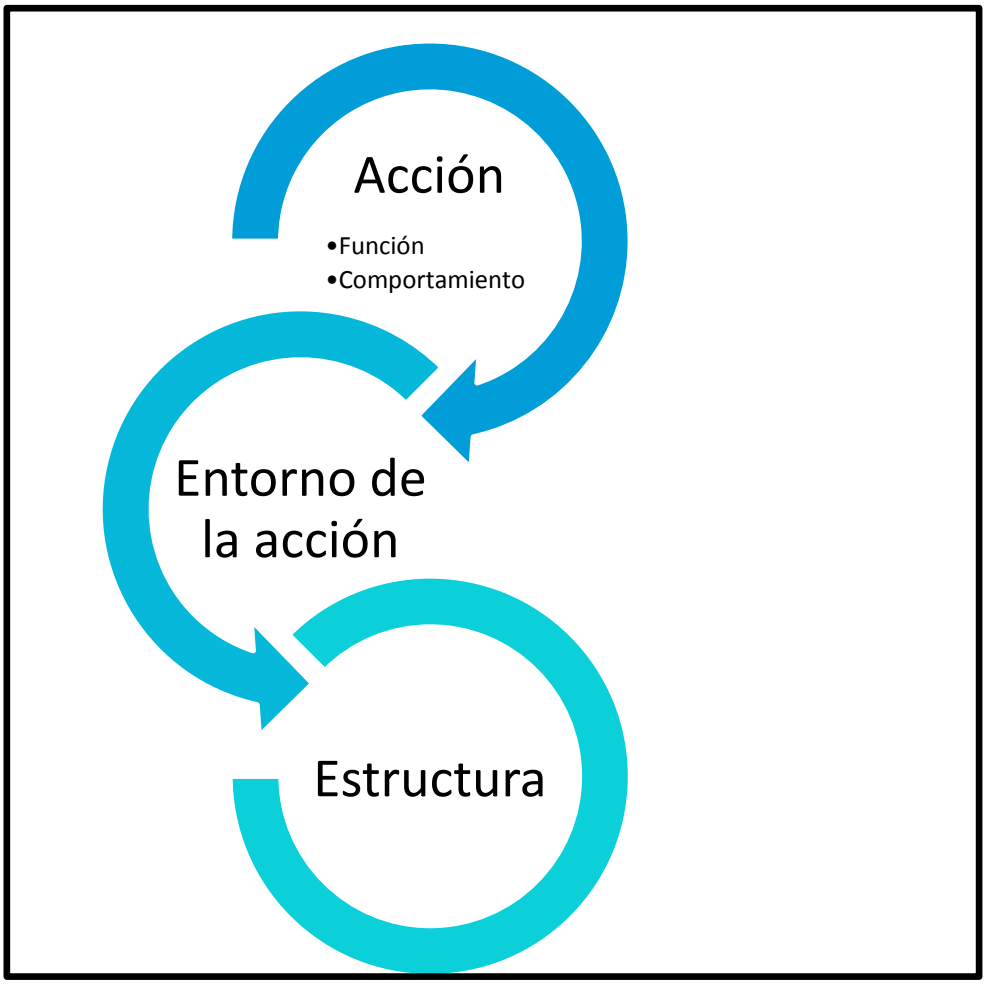


Figura 12: Marco de FaBES

FaBES adquiere sus fundamentos a partir del conocimiento proveniente de DOLCE. Por tanto, la definición de cada una de las capas está relacionada con la citada ontología con el objetivo de mantener una consistencia lógica para poder permitir la posterior construcción de la ontología. Así, se utilizan las clases definidas en profundidad en el Apdo. 2.5.2: *Perdurant, Endurant, NAPO, APO, Qualities, Physical-region, amount-of-matter, Accomplishment, Achievement, Process, State*. En la posterior explicación de cada una de las capas se indicarán las diversas relaciones entre las clases definidas en A-QB y FaBES. A continuación se explica con profundidad la relevancia de los conceptos de acción y entorno en FaBES, como introducción al esquema A-QB.

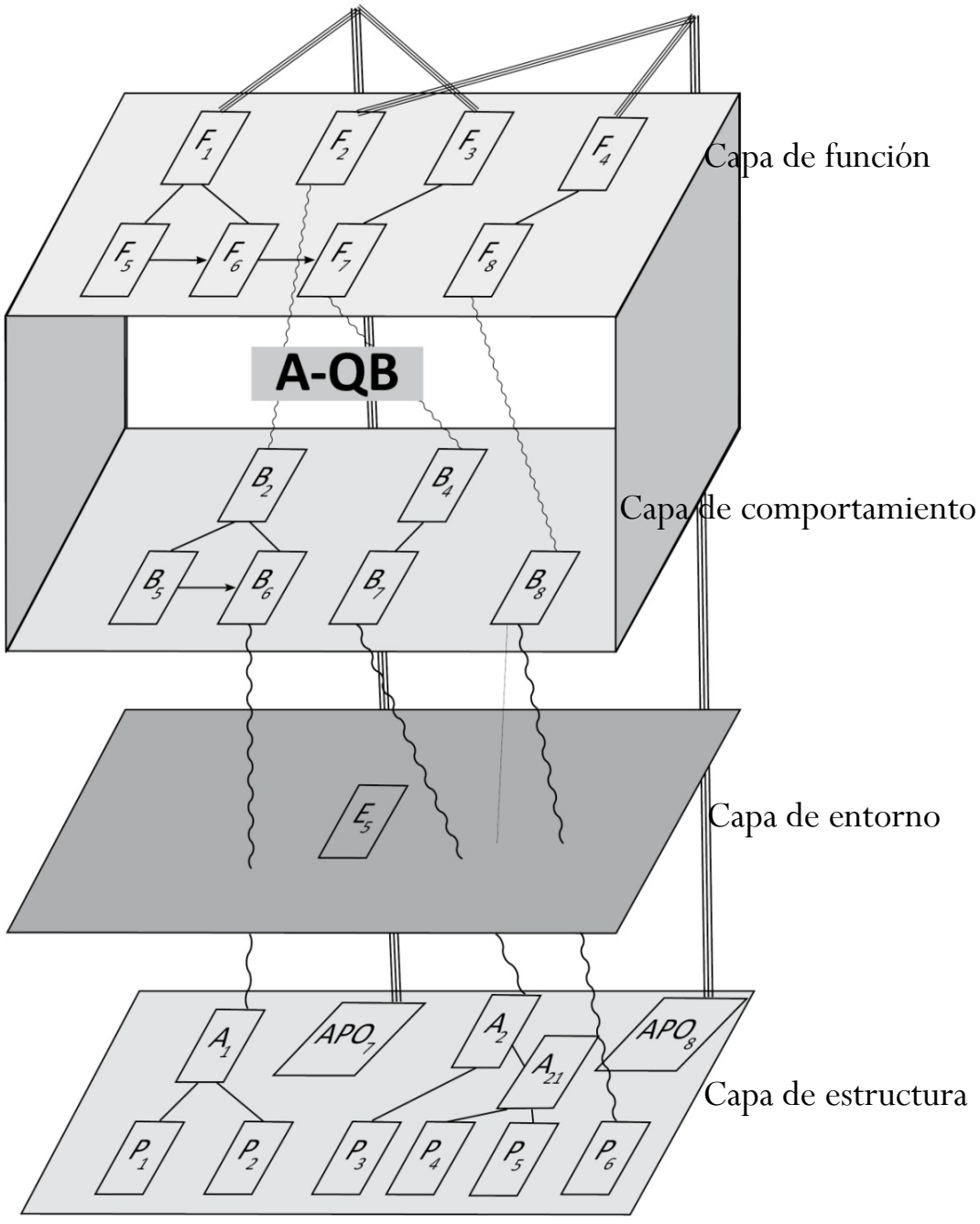


Figura 13: Esquema simplificado del marco FaBES

3.1.3 La importancia de la acción y el entorno en FaBES

El diseño de un objeto dispone de unas características propias análogas a las de un proceso de comunicación, pues en ambos se genera un diálogo entre dos sujetos. En el caso del diseño, entre el sujeto y el objeto, es decir, el diseñador del citado objeto propone una serie de acciones con éste, es decir, plantea una acción (Crilly, 2008). Una acción[cc] que se puede trasladar, por un lado, en una serie de funciones del objeto, pudiendo ser principales o secundarias, o en una serie de comportamientos, los cuales se derivarán a partir de las funciones asignadas al artefacto en cuestión y cuyo agente que promoverá la acción será un componente de dicho artefacto. Y dichas acciones están enlazadas, con diferente prioridad y con un orden, para poderse cumplir.

Por otro lado, el lenguaje natural a través del cual se comunica cualquier información sigue una estructura de frases, la base para construir cualquier tipo de conocimiento, como por ejemplo puede ser un diseño. Y eso deriva en que para formalizar esa información se deba entender cada uno de los elementos del mensaje conformado.

Por tanto, planteando esta hipótesis de trabajo se puede plantear este esquema análogo al proceso de diseño de un objeto (Figura 14) según las teorías de la lingüística en que una frase está conformada por los siguientes elementos: sujeto, verbo y predicado. Dentro de esta estructura tan simple se puede observar las características de la analogía.

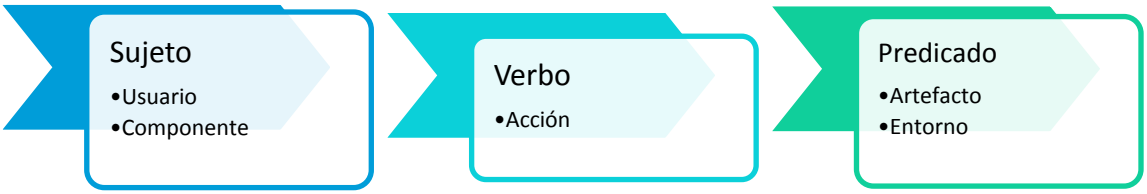


Figura 14: Analogía entre el proceso de comunicación y el diseño

Según la Figura 14 el sujeto es quien ejecuta la acción principal del diseño. Eso permite establecer la posibilidad que dependiendo del usuario de un diseño, si es un agente inteligente o no, éste pueda disponer por tanto de funciones o comportamientos distintos, con una misma estructura. Introduciendo al usuario dentro de las características del diseño se puede aceptar, por ejemplo, que un martillo sirva para clavar clavos para unos usuarios y para otros sirva como pisapapeles.

O también se puede indicar el caso de un tornillo. A priori se podría considerar que la función que puede desempeñar éste es fijar dos piezas para posibilitar la unión entre ellas. Sin embargo, cabe dos interpretaciones según si el tornillo es el elemento de diseño en sí mismo o si pertenece a un diseño compuesto por un conjunto de elementos. En el primer caso, se considera que cumple una función, porque es el elemento óptimo para la fijación de dos elementos, es decir, asegurar un cuerpo en otro. Y según las necesidades, se puede acondicionar a ellas. Así, el usuario (agente inteligente) para esta función sería aquella persona que utiliza el tornillo para realizar una fijación determinada. En el segundo caso, en el que no se utiliza el tornillo para concebir su propio diseño sino como elemento estructural de un conjunto, este elemento ya no estaría desarrollando una función en el diseño sino un comportamiento, pues la acción de fijar no está dirigida por ningún agente inteligente. De hecho, entonces dicha acción se desarrolla indefinidamente hasta que el usuario determine realizar la función opuesta que sería eliminar la fijación creada.

Manteniendo la analogía, el verbo constituye una acción de un diseño de un objeto, que tal como se ha indicado, puede constituirse como una función o un comportamiento. Y finalmente, el predicado conformado por un lado, por el artefacto, que en un análisis sintáctico se consideraría como complemento directo. Se debe a que se requiere por el verbo para que aporte el significado que le falta, o en términos de diseño, la acción necesita un diseño para poderse llevar a cabo (Figura 15). Y por el otro, el entorno conformado por todos aquellos elementos que modifican la acción.

[cc] En esta tesis se considera acción según la definición de la R.A.E. "*Efecto que causa un agente sobre algo*".

Figura 15: Ejemplos de la analogía de lenguaje y diseño

Según la hipótesis anterior, se puede plantear la ecuación [1], que plantea la equivalencia siguiente en referencia al (re)diseño de un objeto. A continuación se describen los factores que la componen:

$$\sum_{i=1}^{n} \text{Estructura}_i + \text{Acción}_i + \text{Entorno}_i = \sum_{i=1}^{m} \text{Estructura}_j + \text{Acción}_j + \text{Entorno}_j \qquad [1]$$

Ecuación [1]: Esta igualdad muestra que el diseño de un objeto se constituye por un sumatorio de un conjunto de 3 elementos (*Structure*, *Action* y *Environment*). Eso establece que si se sustituye uno de los elementos, para mantener la igualdad se han de modificar algunos de los elementos restantes (estructura, acción o entorno) en el otro lado de la igualdad. Eso significaría que si, por ejemplo, el *Environment* cambia para mantener las mismas propiedades del diseño, deben modificarse la *Structure* o las *Action* que lo conforman.

Estructura: Este apartado se refiere a la estructura original de un diseño (subíndice i) y a la posterior modificación en el rediseño posterior de dicho elemento (subíndice j). Esta estructura según lo planteado en el apartado X tiene dos concepciones distintas: por una parte está relacionado con el Objeto Físico (definido por DOLCE, Masolo 03) que se divide entre el NAPO y el APO, por lo que tanto se refiere al usuario del diseño, como al NAPO, la propia estructura del diseño, que al modificarse tanto sus acciones como su entorno, puede adquirir unas propiedades distintas.

Acción: Se refiere a la acción que acomete un diseño, dividiéndose en función o comportamiento. Es función si es un *perdurant* eventual, dependiente del tiempo y dirigida por un APO, y comportamiento en el caso contrario, si es un *perdurant* estático, independiente del tiempo, y regido por un NAPO. De forma análoga al apartado de *Structure*, las acciones tienen una función/comportamiento distinto dependiendo de los distintos elementos dentro de un diseño.[dd].

Entorno: En el rediseño de un objeto, este entorno se modifica debido a que las acciones que se requiere que haga la estructura primaria ya no son cubiertas por el planteamiento original. Este entorno es el que va a marcar las distintas restricciones en el diseño de un objeto y su conjunción va a complementar el resultado final.

Eso establece que en el proceso de rediseño de un objeto sea tan importante la información proveniente de la formalización del objeto como la del usuario que esté haciendo uso de dicho objeto o la del entorno donde se ubica éste.

Un ejemplo se podría constituir con el caso de una cajonera de madera (*Estructura*) utilizada para guardar objetos (*Acción*) en una zona con un clima seco a temperatura ambiente (*Entorno*). Si considera un cambio en la estructura del mueble (*Estructura*), como son sus dimensiones, éste puede servir para otras acciones (colgar ropa) u otros entornos. Si se establece la modificación en el *Entorno,* como puede ser el caso de que la cajonera se encuentre en un entorno inflamable, se deberá sustituir el material si se quiere mantener las propiedades iniciales. Si no es así, la cajonera puede servir como material de combustión (cambio de la *Acción*).

[dd] En los siguientes apartados se hablará más en profundidad al respecto de esta apreciación.

Por otro lado, en el ámbito referente a la ingeniería ontológica, al establecer esta estructura de tríada, se facilita la correlación con la información disponible en el lenguaje RDF u OWL, en el que el mensaje se distribuye en tripletes de información tal como se ha indicado en el apartado 2.4.4.

Por tanto, traduciendo la ecuación en lenguaje natural, un cliente quiere algo que cumpla una función y le indica una serie de restricciones al respecto para que las ejecute. Se plantea entonces que el usuario pide que ejecute una función más una serie de restricciones. Dicha función si es muy compleja se puede subdividir en varias subfunciones para poder cumplirse.

Respecto al enlazamiento de diferentes acciones, tal como se ha podido ver en la Figura 13 se plantea un esquema basado en dos ejes: pertenencia y consecución (Figura 16). Con respecto a la pertenencia, se plantea que una acción compleja esté formada por dos acciones más simples. Y en referencia al apartado de consecución, se concibe que para que una acción se deba realizar, se establezcan un orden determinado de acciones previas.

Para poder entender de manera práctica este planteamiento, se puede imaginar uno la acción de lavar los platos (*Action_C*). Simplificando, dicha acción está ***formada por*** dos acciones secundarias: enjabonar (*Action_A*) y enjuagar (*Action_B*). Así la *Action_B* (enjuagar) está ***precedida por*** la *Action_A* (enjabonar) y ambas ***son parte de*** la Action_C (lavar los platos).

3.1.4 A-QB

El A-QB (Action Cube de "QB" pronunciado en inglés), es una evolución del B-Cube (Chulvi, 13) que plantea que "*un comportamiento venga definido por una palabra o taxón, hecho que puede dar pie a ambigüedades e interpretaciones erróneas, sino que el comportamiento viene definido como un vector tridimensional (X, Y, Z), determinado por sus características y cualidades*" (Aptdo. 2.1.3). El A-QB concibe una evolución del citado sistema, debido a una serie de inconsistencias en el desarrollo del B-Cube y se adapta al ámbito de las acciones, sean éstas funciones o comportamientos, para poder ser utilizado en una ontología basada en el marco FaBES. Para ello se profundiza en una mayor y detallada integración de la ontología OntoRFB.

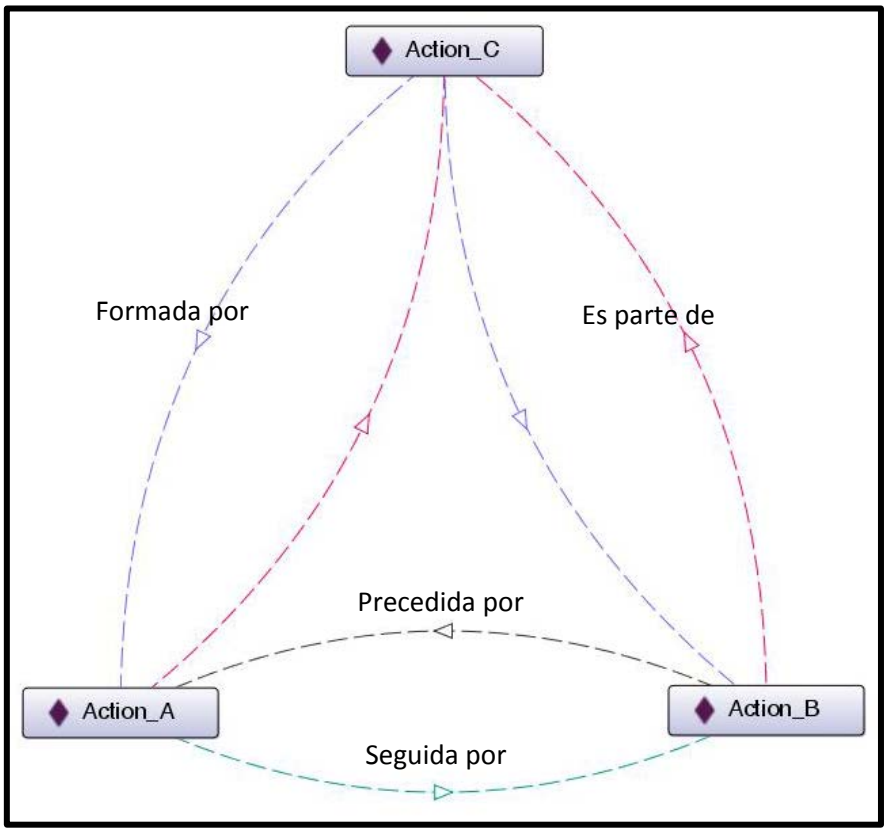


Figura 16: Enlazamiento de las acciones

La elaboración del modelo A-QB recoge las tres definiciones básicas de DOLCE (Masolo 03) que también se utilizan en el B-Cube (Aptdo. 2.1.3) referidas a *Endurant*, *Perdurant* y *Cualidad*. La primera diferencia se establece debido a que dentro del apartado de *Endurant*, el A-QB se centra sólo en los *endurant*s físicos al considerar el objeto que se (re)diseña como tal; y respecto a la cualidad, se toma como referencia únicamente, la temporal, al ser inherente a un *perdurant*.

Eje X-Y

La segunda apreciación con respecto al modelo del B-Cube es en referencia al eje X-Y. Respecto al eje Y, referido al *Perdurant*, se hereda la organización original de la ontología DOLCE respecto a las cuatro clases heredadas: *State (0), Process (1), Accomplishment (2)* y *Achievement (3)*. Se colocan en el citado orden en relación al tiempo que pueden estar presentes y a la dependencia del tiempo que muestran (Figura 17):

- *State*: Acción cuyo comportamiento es acumulativo, no tiene una finalización natural y no se puede descomponer en acciones diferentes. Por ejemplo, *state* es estar sentado, donde si sumas dos acciones de estar sentado sigues estando sentado, y cualquier momento de la acción se describe con el mismo nombre. Dicha acción finaliza cuando la persona decide cambiar el estado.
- *Process*: Acción cuyo comportamiento es acumulativo, no tiene una finalización natural y se puede descomponer en acciones diferentes. Así, correr es un *process*. Del mismo modo que en el ejemplo anterior, si sumas dos acciones de correr, el resultado sigue siendo correr, pero en este caso si descompones la acción en partes tienes estados distintos: pie derecho o pie izquierdo.
- *Accomplishment*: Acción cuya función no es un acto instantáneo y no es acumulativo. Por ejemplo, dar una conferencia: con otra acción igual has dado dos conferencias, pero en este caso no es un acto instantáneo pues tiene una duración determinada.
- *Achievement*: Acción cuya función no es un acto instantáneo y no acumulativo. Como ejemplo la acción de caer un rayo es un *achievement*, pues no es acumulativo, si sumas otra acción igual da como resultado la caída de dos rayos, y es instantáneo.

En referencia al eje X, se mantienen las definiciones que el B-Cube heredaba de la taxonomía OntoRFB más la incluida al apartado de Magnitud, es decir*, Spatial Location* (0), *Topological Connectedness* (1), *Energy* (2), *Magnitude* (3), *Signal* (4).

Eje Z

Respecto al eje Z, se toman los valores indicados en el B-Cube heredados de la taxonomía OntoRFB en relación a la cualidad temporal: *Initial SoA*, *Immutable SoA* y *Terminal SoA (*indicado en el B-Cube como *Final SoA)*. Se restructura de manera que *Initial SoA* se considere con un efecto negativo correspondiéndose con el valor -1. Posteriormente, el valor referido a *Immutable SoA* se considera como un valor 0, lo que implica resituarlo en el en el eje central. Y finalmente, el valor referido a *Terminal SoA* que está contemplado como el valor 1, que implica que considera un efecto positivo al final de la acción.

Esta colocación representa que en la organización de acciones, se configuren dos sentidos de trabajo: El primero, cuando un *endurant* está actuando en un sentido sobre otro, es decir, está provocando una acción; y el segundo, a la vez, en el otro sentido se provoca con el valor negativo. Igualmente, este hecho permite representar en el marco de FaBES la selección de sólo un comportamiento al interaccionar dos elementos de la estructura.

En consistencia con lo planteado en el apartado anterior sobre la relevancia de la acción en FaBES, permite la traslación de la información disponible en lenguaje natural. Como ejemplo, una frase tal como: "*El usuario atornilla un clavo*" se puede descomponer en:

Endurant: Usuario, clavo.

Cualidad física (X): *Topological Connectedness* (1).

***Perdurant* (Y):** *Accomplishment* (2)

Cualidad temporal (Z): *Terminal SoA* (1).

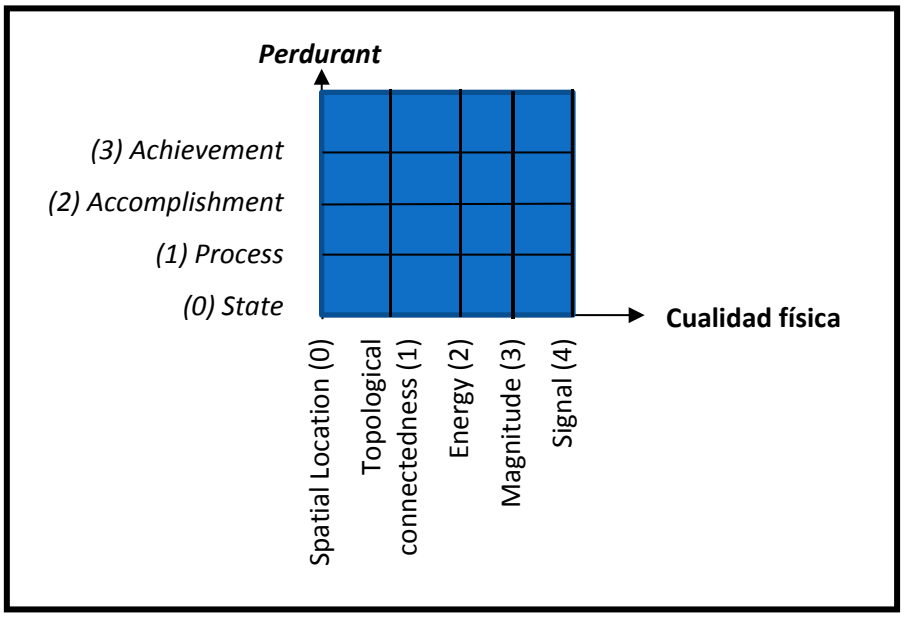


Figura 17: Eje XY del A-QB

Si dicha frase se plantea en el sentido reflexivo: "*Un clavo es atornillado por el usuario*", el valor es el mismo. Sin embargo, si se provoca la acción de desatornillar, la descomposición cambia en el apartado referido de la cualidad temporal, tomando como valor -1. De esa forma, se observa cómo se pueden evitar contrasentidos cuando se utilizan verbos reflexivos para definir una acción determinada de un objeto (*endurant*).

Esto permite que el eje (0, 0, 0) se considere como localización espacial entre dos estructuras, es decir, estado neutro o de reposo entre ellas, pues el sentido de la acción es indiferente y la acción no provoca ningún movimiento. Se estaría hablando entonces de dos estructuras que no tienen interacción entre ellas, es decir, ningún comportamiento en común (Aptdo. 3.1.6).

Y luego, para evitar confusiones con funciones que aparentemente pueden considerarse como comportamientos, como puede ser el caso de una caja, cuya función es guardar objetos y que según el modelo A-QB se puede descomponer en:

Endurant: Usuario, caja.

Cualidad física (X): *Spatial Location* (0).

***Perdurant* (Y):** *Accomplishment* (2)

Cualidad temporal (Z): *Immutable SoA* (0).

El punto (0, 2, 0), que se ha obtenido, muestra que no ha variado su cualidad física. Además, es un *perdurant* eventual, dependiente del tiempo, ya que el objeto estará guardado en la caja un tiempo determinado fijado por el usuario de la caja, pudiendo ser éste, un segundo, o mil años; siempre con un final consciente de la acción.

A partir de ahí se establece, como resultado el modelo A-QB (Figura 18) aunando las ventajas propias del modelo B-Cube adaptándolo a las características de FaBES. Tal como se observa en la Figura 13, el A-QB engloba las capas de función y comportamiento.

A continuación, en los siguientes apartados, se describe en profundidad el significado de las diferentes capas del marco FaBES: función, comportamiento, estructura y entorno.

3.1.5 Capa de función

La definición del concepto función (indicado en la Figura 13 como F_i) se establece como la acción inherente al diseño del cual está siendo procesado, y que está intencionada o deseada por un agente inteligente. Esta definición proviene de la unificación de las posturas de Borgo (2006), que plantea una función como el

conjunto de usos al cual un objeto se emplea y se diseña, y de Chandrasekaran (2005) que considera que una función está destinada o deseada por un agente.

Con el fin de abarcar un número finito de acciones en esta capa, y evitar futuros problemas de cálculo en su posterior adaptación para una ontología, tal como se ha comentado en el apartado anterior, esta capa se engloba dentro del modelo A-QB.

Según el modelo del A-QB, esta capa está representada generalmente por un *perdurant* eventual, dependiente del tiempo, es decir, *accomplishment* o *achievement*, pues dicha acción está intencionada o deseada por un agente inteligente, por tanto va a tener un fin concreto que se concreta en un tiempo determinado, por lo que es dependiente.

En concordancia, las funciones están realizadas por un APO. Se puede observar en el ejemplo del apartado anterior referido a la frase: "*El usuario atornilla un clavo*" que se corresponde con el punto (1, 2, 1).

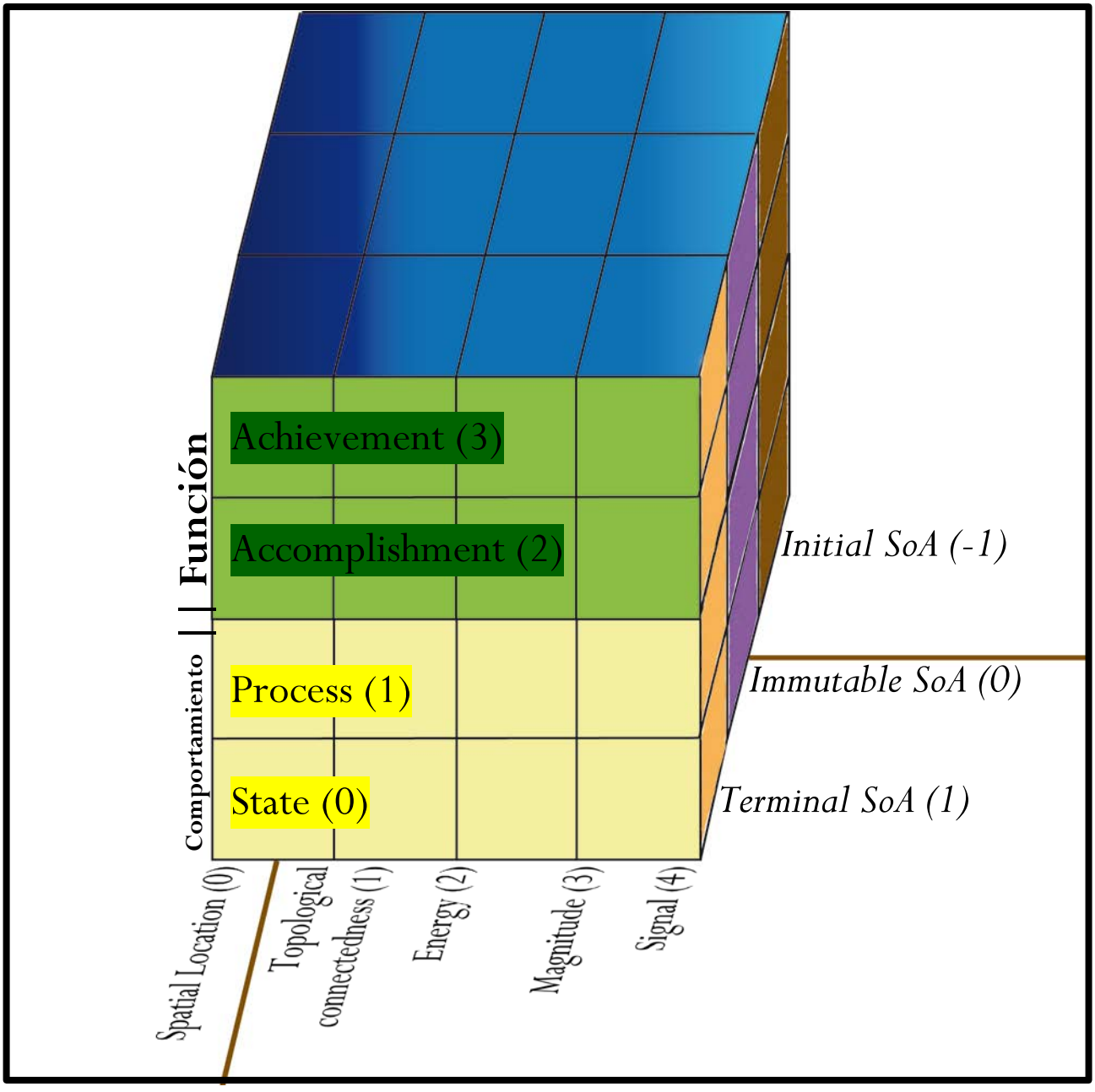


Figura 18: A-QB

3.1.6 Capa de comportamiento

La definición del comportamiento (indicado en la Figura 13 como B_i) se establece como la acción que relaciona la estructura de un diseño, su función y es realizada por un NAPO.

Esta idea parte de la conjunción de las diversas definiciones sobre el comportamiento (Tor, 2002; Masolo, 2003; Chandrasekaran, 2005; Vermaas, 2007; Borgo, 2006; Kitamura, 2003). Así a partir del trabajo de Borgo (2006) se considera el comportamiento como "una relación entre un objeto y el proceso en el cual participa". Esta definición, permite establecer la analogía con el esquema FBS y facilita la posterior formalización de FaBES al relacionar de forma explícita la función de un objeto con su estructura.

Por los mismos motivos indicados en el apartado anterior, esta capa se engloba también dentro del modelo A-QB. Según dicho modelo, esta capa está representada generalmente por un *perdurant* estacionario, independiente del tiempo, es decir, *state* and *process*, ya que por propia definición de este tipo de acciones (relación entre un objeto y el proceso en el cual participa), están presentes durante un tiempo determinado pero son independiente de éste.

Por tanto, los comportamientos se realizan por un NAPO. Por ejemplo, la acción *conectar* en el ejemplo de un cable que conecta un monitor se definiría según el A-QB como (1, 0, 0), es decir,

***Endurant*s**: cable y monitor.

Cualidad física (X): *Topological Connectedness* (1)*;*

***Perdurant* (Y):** State (0);

Cualidad temporal (Z): *Inmutable SoA* (0).

En este ejemplo se puede observar como el *perdurant* es estacionario, tal como se había indicado en la definición. Igualmente se puede remarcar que la cualidad temporal es *Immutable SoA*, por lo que es lo mismo decir, que el cable conecta el monitor como que el monitor es conectado por el cable, al no haber cambio de sentido de la energía en la acción.

En un diseño como puede ser el de una aspiradora, puede ser necesario que para la unión de diferentes piezas sea precisa la utilización del tornillo, pero no por ello, una de las funciones de la aspiradora consiste en fijar dos materiales. En este caso, la acción que realiza el tornillo es un comportamiento de la aspiradora y no una función ya que el tornillo realiza una tarea *para la cual se ha diseñado* (unir diferentes piezas) y no una función propia de la aspiradora que es aspirar partículas.

3.1.7 Capa de estructura

La definición del concepto estructura se establece como la representación física de un diseño. Se indica en la Figura 13 como APO, A_i y P_i referido a los elementos *Agentive Physical Object* (*Objetos físicos agente*), *Assembly* (*Ensamblaje*) y *Part* (*Parte*).

Por tanto, se recogen en esta capa los objetos físicos. Tomando como referencia la taxonomía DOLCE, se plantean dos tipo de estructuras: APO y NAPO. Los APO son agentes que tienen una intención o desean una acción, en coherencia con la capa de función (Aptdo. 3.1.8) y respectivamente, los NAPO son las representaciones físicas de un diseño, que se dividen en partes y ensamblajes.

Debido a que el modelo propuesto pudiera componerse de un número excesivo de elementos, se han incorporado las bases del diseño modular (Gerhenson, 1999) para poder optimizar la fase del ensamblaje. Así, la estructura de un objeto se divide entre partes y ensamblajes (indicado en la Figura 13 como P_k y A_l, respectivamente).

Una parte es una pieza que puede formar un ensamblaje y que está formada de un material determinado, lo cual es dependiente de la capa del entorno. Un ensamblaje se puede considerar como un conjunto mínimo de dos partes conectadas que tienen una serie de restricciones físicas restringidas por reglas de diseño modular (capítulo 2.1.1) y por condiciones de la capa de entorno.

3.1.8 Capa de entorno

La definición del concepto de entorno (indicado en la Figura 13 como E_i) se establece como una región dónde sólo cualidades físicas pueden ubicarse. Según el ambiente donde se reproduzca la acción, es decir, función y/o comportamiento, van a surgir una serie de restricciones acotando el ámbito de desarrollo de éstas. Dichas restricciones se basan en una serie de cualidades físicas dependientes al entorno donde se desarrolla la acción en cualquiera de sus fases (sólido, líquido y gaseoso): temperatura, presión y volumen específico[ee]. Por tanto, están relacionadas con conceptos de resistencia de materiales: rigidez, equilibrio mecánico, flexión y torsión.

Por tanto, esta capa define el tipo de materiales que la estructura puede tener dependiendo del entorno de la acción donde se halle. En el caso de APOs, se podría considerar también a los factores externos, como el ámbito de una empresa ya que proporciona información clave para el desarrollo del trabajo.

Por ejemplo, para un caso posible de corrosión atmosférica se tendrían en cuenta los diferentes factores ambientales como la humedad relativa, la temperatura y los contaminantes existentes en el entorno (Vázquez, 1991).

3.1.9 Aplicación práctica

Para una correcta comprensión del significado de cada una de las capas, se van a aplicar a cuatro ejemplos simplificados de estructuras mostrando la aplicación de FaBES a diferentes escalas:

- un horno monoestrato de rodillos dedicado a la cocción de azulejos que muestra el ámbito de los sistemas automatizados (Figura 23);
- una aspiradora (Figura 26), que muestra el ámbito de los sistemas dependientes de un usuario;
- un tornillo (Figura 28), elemento estructural.
- un portaminas (Figura 30), ejemplo detallado de un sistema simple.

3.1.9.1 HORNO MONOESTRATO DE RODILLOS

Los pavimentos y revestimientos cerámicos, denominados también baldosas cerámicas, son piezas que están constituidas normalmente por un soporte de naturaleza arcillosa y porosidad variable, recubiertas o no por una capa de esmalte. Este recubrimiento vítreo se realiza básicamente para impermeabilizar el soporte arcilloso, más o menos poroso, evitando así la retención de suciedad y absorción de posibles manchas, y dotar a la pieza de mayor riqueza estética y decorativa. Existe una gran variedad de baldosas cerámicas, en cuanto a la forma y en cuanto a sus características técnicas (absorción de agua, resistencia a la abrasión, resistencia mecánica, etc.), condicionada por las múltiples utilidades de estos productos y sus variados lugares de colocación (Escardino, 2001).

La cocción de los productos cerámicos es una de las etapas más importantes del proceso de fabricación, ya que de ella dependen gran parte de las características del producto cerámico. Las variables fundamentales a considerar en la etapa de cocción son el ciclo térmico y la atmósfera del horno, que deben adaptarse a cada composición y tecnología de fabricación, dependiendo del producto cerámico que se desee obtener.

Esta cocción se realiza en hornos monoestrato de rodillos, con ciclos de cocción entre 40 y 80 minutos, dependiendo del tipo de producto a cocer. En los hornos monoestrato, las piezas se mueven por encima de los rodillos y el calor necesario para su cocción es aportado por quemadores gas natural-aire, situados en las paredes del horno.

Podemos diferenciar tres etapas básicas en el proceso de cocción: precalentamiento, cocción y enfriamiento.

- **Precalentamiento**: Es el intervalo desde el inicio del proceso hasta temperaturas de 800-900 ºC. Es una etapa fundamental en la definición del ciclo de cocción.

[ee] El volumen específico es el volumen ocupado por unidad de masa de un material. Es la inversa de la densidad, por lo cual no depende de la cantidad de materia.

- **Cocción**: Aunque en realidad la cocción es el proceso completo, se preserva este término para la fase en que se producen las transformaciones deseadas. Se trata del intervalo que va desde los 800- 900 ºC, hasta la temperatura máxima de cocción (1050 – 1200 ºC).

- **Enfriamiento**: Cuando finaliza el aporte calorífico se inicia el período de enfriamiento. El sistema evoluciona durante el enfriamiento por lo que las características finales del producto dependerán en gran medida de cómo se ha realizado éste.

Con el fin de simplificar el proceso, se va a considerar que en la fase de cocción se mantiene la temperatura constante y la etapa de precalentamiento dura hasta alcanzar la temperatura máxima de cocción. De esa forma se establece la Figura 19. Igualmente se va a considerar que un horno monoestrato de rodillos está conformado por las siguientes piezas: quemadores de gas metano, ventiladores, estructuras metálicas y rodillos de alúmina refractaria.

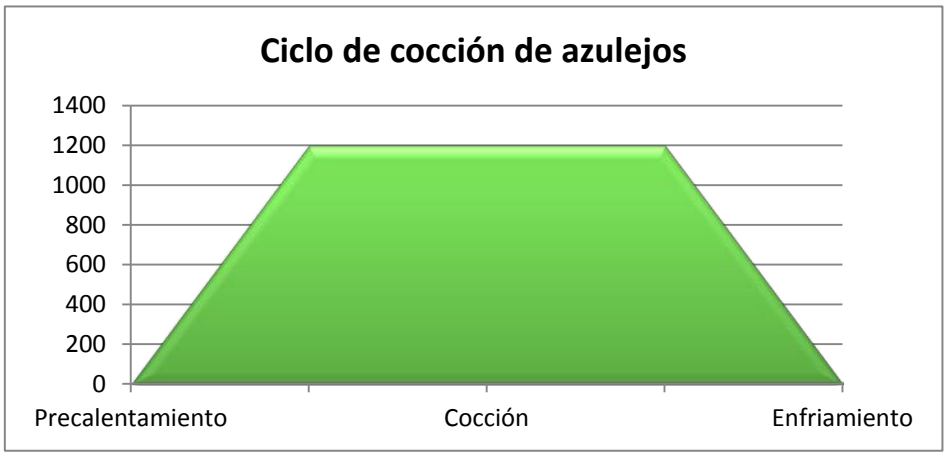


Figura 19: Simplificación del ciclo de cocción de baldosas cerámicas

Acción

La acción fundamental es la cocción de baldosas cerámicas. Según el modelo A-QB aunado al esquema FaBES, las acciones del diseño simplificado de un horno monoestrato de rodillos se muestran en la Figura 20. En dicho diagrama se puede observar todos los elementos del marco FaBES, es decir, tanto funciones (etiqueta FaBES: FUNCIÓN y A-QB: SUBFUNCIÓN) como comportamientos (etiqueta FaBES/A-QB: COMPORTAMIENTO y A-QB: SUBCOMPORTAMIENTO) y las relaciones que se establecen entre ellos con el usuario (una persona) y la estructura para acometerse (etiqueta FaBES: ESTRUCTURA). Igualmente se muestran las características del entorno (etiqueta FaBES: ENTORNO). A continuación se irán tratando en profundidad cada uno de los apartados.

Función

La función objetivo de un horno de estas características es cocer baldosas cerámicas. Un horno es una máquina parcialmente automatizada de un proceso complejo donde intervienen muchas variables tales como la temperatura, el contenido en oxígeno o la humedad de la pieza entre otras.

Es un diseño complejo que adecúa su funcionamiento a las necesidades del producto que se está produciendo en ese momento. Si se analiza esa automatización parcial, se puede observar que la acción de cocer proviene del calor producido por los diversos quemadores de gas natural colocados a lo largo del horno más las distintas corrientes de aire que se inducen en el interior de éste.

No obstante, el control de las variables tiene que realizarse por una persona, si no, es imposible cumplir la función del horno, pues las piezas pueden no alcanzar las temperaturas máximas adecuadas en la pieza, o por otro lado, podrían carbonizarse si las temperaturas máximas fueran demasiado elevadas.

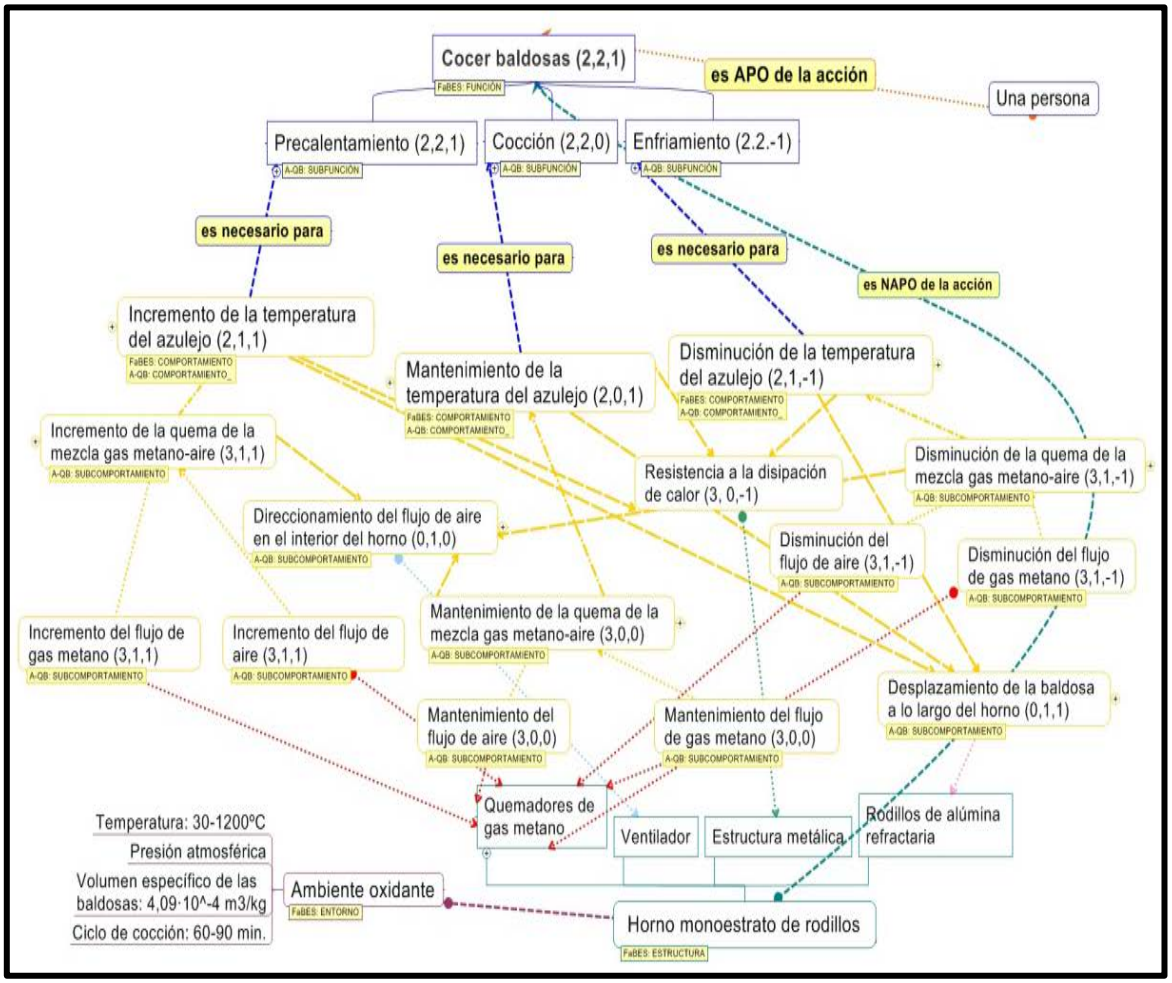


Figura 20: Esquema FaBES simplificado de un horno monoestrato de rodillos.

Por otro lado, según el esquema FaBES y la simplificación indicada anteriormente, la función objetivo (cocer baldosas) se puede dividir en 3 subfunciones: precalentamiento, cocción y enfriamiento, tal como se muestra en la Figura 21. En dicha figura se puede observar la explicación de los puntos elegidos según el modelo A-QB. En el caso de la función objetivo el detalle según el A-QB es (2, 2, 1), es decir, una persona cuece baldosas gracias a un horno monoestrato de rodillos:

***Endurant*s**: Una persona y horno monoestrato de rodillos. En el primer caso, la persona es el APO de la función y el horno monoestrato de rodillos es el NAPO (Figura 20).

Cualidad física (X): *Energy* (2)*;* ya que el hecho de cocer baldosas corresponde a un cambio de estado químico en la baldosa, modificando sus propiedades físicas*.*

***Perdurant* (Y):** Accomplishment (2); al ser una acción dependiente del tiempo cuyo cumplimiento ocupa una fracción del tiempo.

Cualidad temporal (Z): *Terminal SoA* (1); ya que las propiedades deseadas de la baldosa se obtienen al final del proceso.

Con ello se muestra que en este caso, la función del horno está dirigida por un agente inteligente que acota el desarrollo de la acción a unas variables concretas en su desarrollo cumpliendo las hipótesis de partida de FaBES y el A-QB.

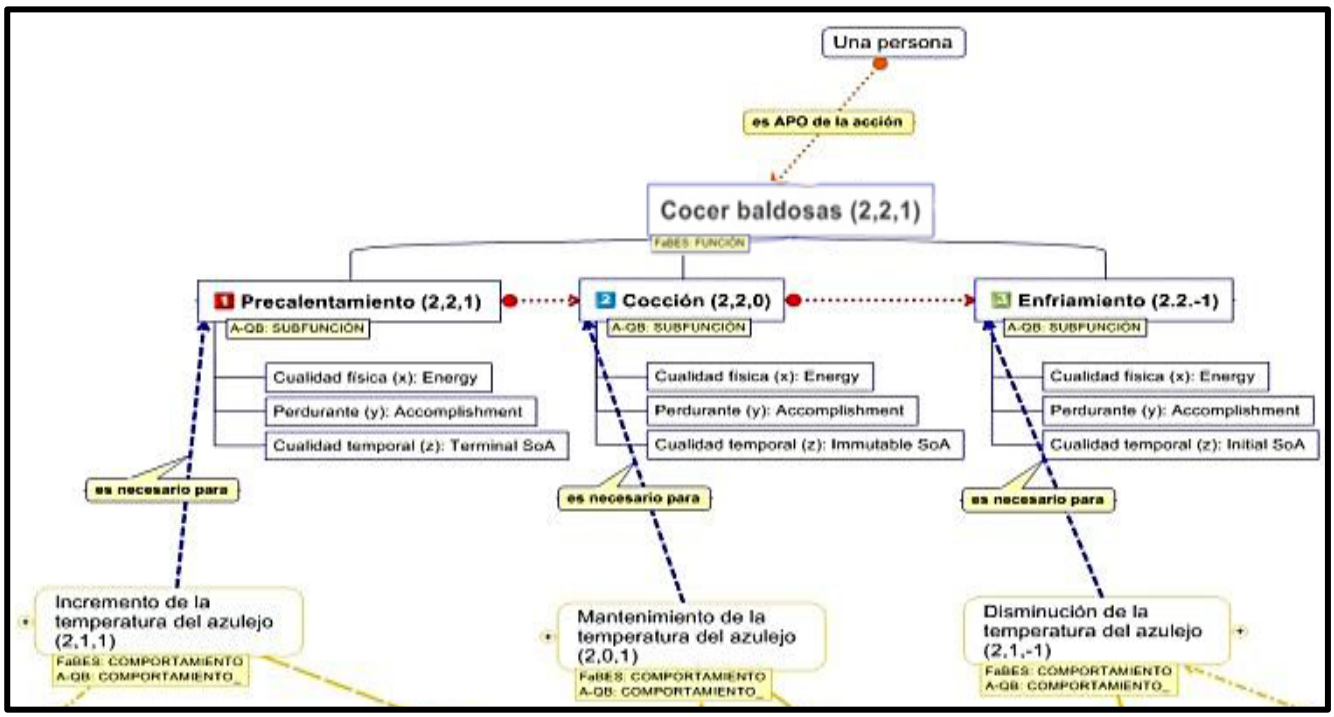


Figura 21: Esquema FaBES de las funciones del horno monoestrato.

Comportamiento

En el caso del horno monoestrato, los comportamientos derivados de la función principal son incremento, mantenimiento y disminución de la temperatura del azulejo (Figura 22). El incremento de temperatura provoca otros comportamientos ligados como son las reacciones químicas, que permiten a su vez modificar la estructura interna de la pieza que permite mejorar sus propiedades físicas caracterizando la utilización de un azulejo, que no se muestran en la figura para simplificar el proceso.

En el caso del comportamiento referido al incremento de temperatura del azulejo según el modelo A-QB el valor es (2, 1, 1), es decir, el quemador de gas metano incrementa la temperatura del azulejo:

Endurants: Quemador de gas metano y azulejo. Tanto el quemador como la baldosa son NAPO (Figura 20).

Cualidad física (X): *Energy* (2)*;* ya que el hecho de cocer el azulejo corresponde a un cambio de estado químico en la baldosa, modificando sus propiedades físicas*.*

***Perdurant* (Y):** *Process* (1); al ser una acción independiente del tiempo cuya acción no tiene un punto de finalización natural, pues depende del usuario del horno o de un dispositivo automático, que indique en qué momento la pieza va a pasar al estado de mantenimiento de la temperatura.

Cualidad temporal (Z): *Terminal SoA* (1); ya que las propiedades deseadas de la baldosa se obtienen al final del proceso.

Estructura

En este primer caso, la estructura que conforma un horno, está condicionada con el objetivo de optimizar la función principal de éste: cocer azulejos (Figura 23). El esquema FaBES se muestra en la Figura 24 referido a la estructura simplificada que se ha tenido en cuenta.

Entorno

En este primer caso, el ambiente oxidante y con altas temperaturas necesarias para una cocción adecuada del azulejo, condicionan la elección de los materiales conformantes de la estructura así como su vida útil. Las variables físicas que se han tenido en cuenta son las siguientes:

Temperatura: *30-1200 ºC.*

Volumen específico de un azulejo tipo*: 4,09·10^{-4} m^3/kg.*

Presión*: Atmosférica.*

Ciclo de cocción*: 60-90 min.*

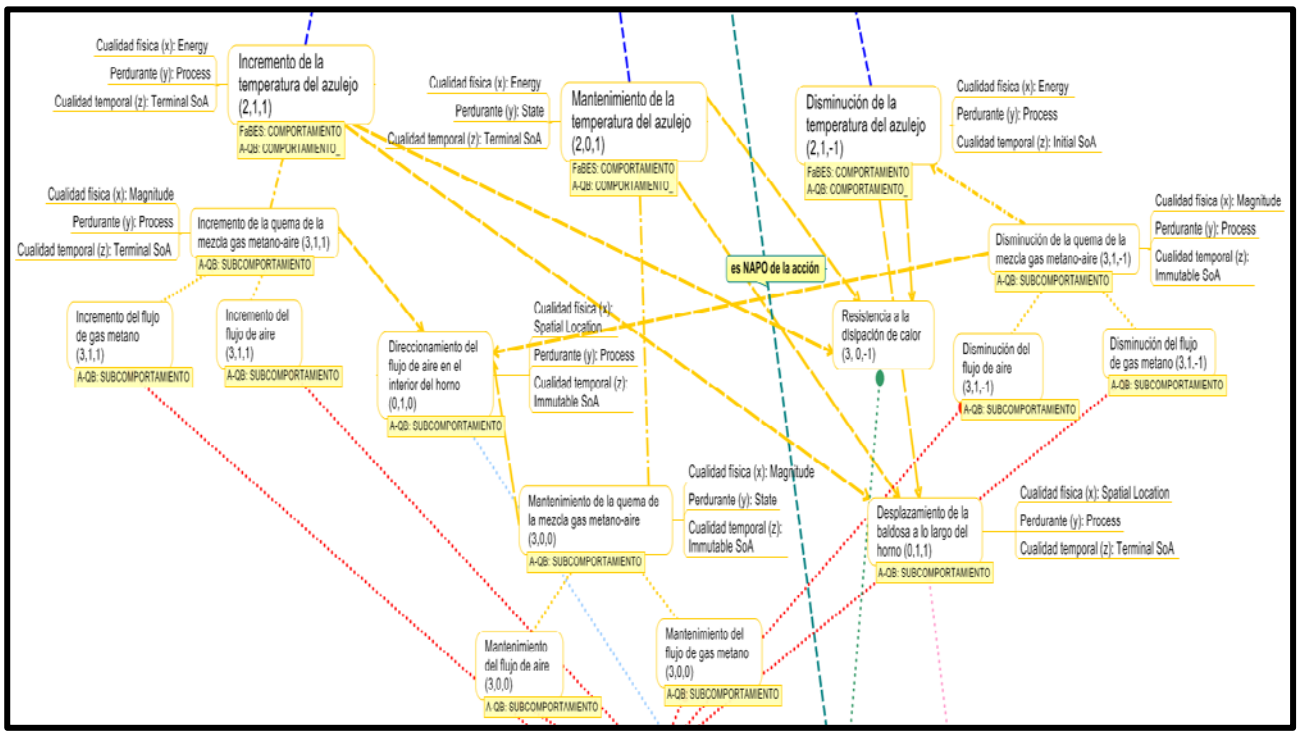


Figura 22: Esquema FaBES de los comportamientos del horno monoestrato

Figura 23: Horno monoestrato de rodillos.

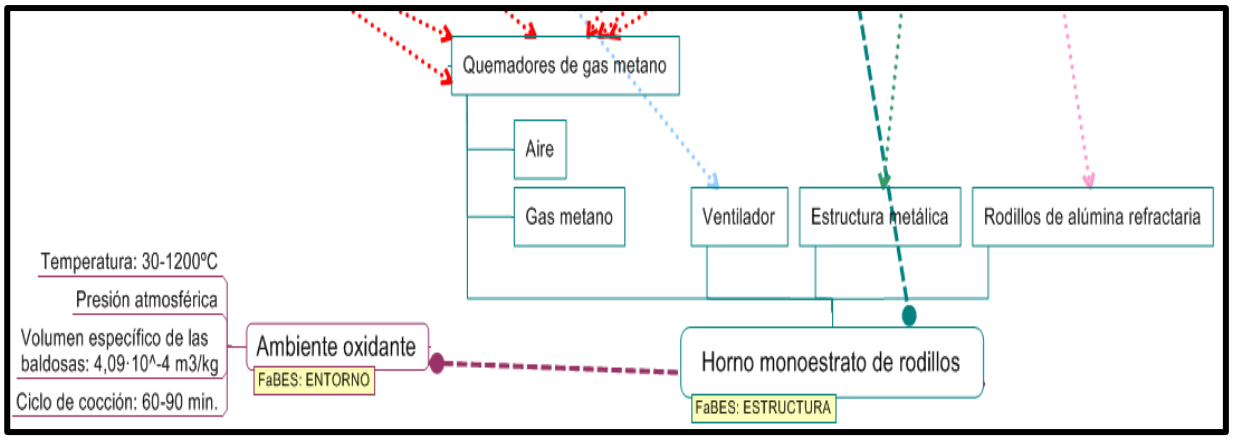


Figura 24: Esquema FaBES de la estructura y entorno del horno monoestrato

3.1.9.2 ASPIRADORA

Acción

La acción fundamental es la aspiración de polvo. Según el modelo A-QB aunado al esquema FaBES, análogamente al caso anterior del apartado 3.1.9.1, las acciones del diseño simplificado de una aspiradora se muestran en la Figura 25.

Función

En este caso, se pueden hallar dos funciones según si se considera el punto de vista de la persona que utiliza como medio la aspiradora para limpiar, o siendo más estrictos, para quitar el polvo, y otra si se considera el punto de vista desde el diseño, en el cual la función principal de la aspiradora, tal como indica su propio nombre es aspirar. Si se considera este segundo punto de vista que podría plantearse como más problemático, se puede llegar a estimar que a priori no interviene ningún agente que intervenga en la acción de aspirar. No obstante, la pregunta que se podría establecer a continuación es qué elementos la aspiradora quiere aspirar. Los elementos que aspira la aspiradora no pueden ser seleccionados por la propia aspiradora, pues no es un agente inteligente, entonces tiene que aparecer en la acción un agente inteligente para poder realizar la citada acción en el ámbito en el que el agente desea.

Por otro lado, según el esquema FaBES y la simplificación indicada anteriormente, la función objetivo (aspirar polvo) se muestra en la Figura 25. Ahí se puede observar la explicación de los puntos elegidos según el modelo A-QB. En el caso de la función objetivo el detalle según el A-QB es (1, 3, -1), es decir, una persona dirige una aspiradora que aspira el polvo:

Endurants: Una persona, aspiradora y polvo. En el primer caso, la persona es el APO de la función y la aspiradora y el polvo es el NAPO (Figura 25).

Cualidad física (X): *Topological Connectedness* (1)*;* ya que el hecho de aspirar consiste en una separación del contacto de las motas de polvo con el suelo.

***Perdurant* (Y):** Achievement (3); al ser una acción dependiente del tiempo cuyo cumplimiento es instantáneo.

Cualidad temporal (Z): *Initial SoA* (-1); ya que el *perdurant* actúa eliminando el *endurant*, es decir, el polvo.

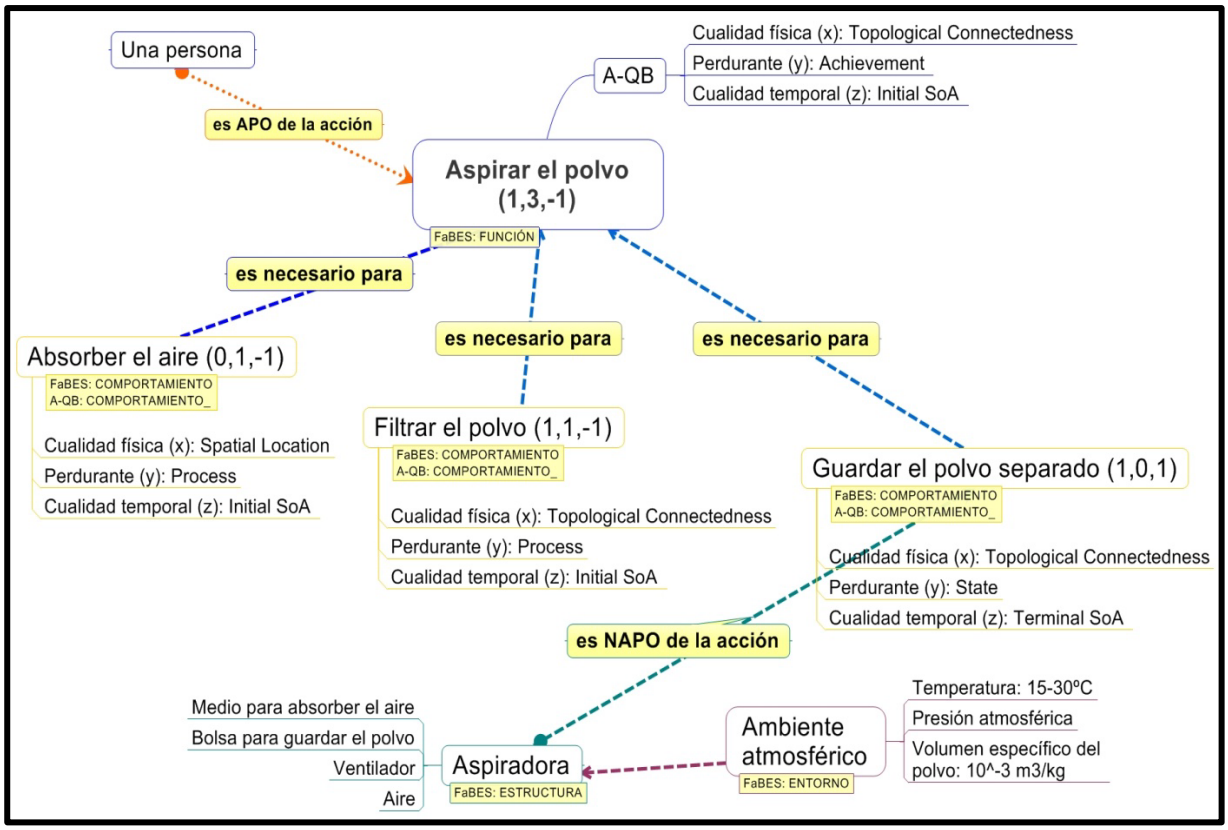


Figura 25: Esquema FaBES de una aspiradora

Comportamiento

En el caso de la aspiradora, los comportamientos ligados a su función objetivo están relacionados con la absorción de aire, filtrar el polvo y guardar el polvo separado. Según el modelo A-QB el valor del primer comportamiento citado es (0, 1, -1), es decir, la aspiradora absorbe aire (con polvo):

***Endurant*s**: Aspiradora y aire (con polvo). Tanto la aspiradora como el aire son NAPO (Figura 25).

Cualidad física (X): *Spatial Location* (0)*;* ya que el aire se desplaza de su localización hacia el interior de la aspiradora*.*

***Perdurant* (Y):** Process (1); al ser una acción independiente del tiempo cuya acción no tiene un punto de finalización natural, pues depende del usuario de la aspiradora que indique en qué momento qué zona va a estar el robot para aspirar aire y por tanto cuánto aire.

Cualidad temporal (Z): *Initial SoA* (-1); ya que el *perdurant* actúa sustrayendo el *endurant*, es decir, el aire.

Estructura

En este caso, la evolución en la estructura de la aspiradora ha permitido la automatización parcial de parte de sus funciones. Lo cual se puede observar en el modelo robot SR8855 de la empresa Samsung (Figura 26).

Entorno

En este segundo caso, las condiciones del entorno no condicionan la elección de los materiales conformantes de la estructura (Figura 25). Las variables físicas que se han tenido en cuenta son las siguientes:

Temperatura: *15-30 ºC.*

Presión*: Atmosférica.*

Volumen específico del polvo*: 10^{-3} m^3/kg.*

Figura 26: Modelo de aspiradora robot Navibot SR8855 de Samsung (Navibot, 2012)

3.1.9.3 TORNILLO

Acción

La acción fundamental es la fijación de dos elementos. Según el modelo A-QB aunado al esquema FaBES, análogamente al caso anterior del apartado 3.1.9.1, las acciones del diseño de un tornillo se muestran en la Figura 27.

Función

A priori se podría considerar que la función que puede desempeñar un tornillo es fijar dos piezas para posibilitar la unión entre ellas. Sin embargo, caben dos interpretaciones según si el tornillo es el elemento de diseño en sí mismo, caso que sí estaría realizando una función, o si pertenece a un diseño compuesto por un conjunto de elementos, situación que conformaría un comportamiento[ff].

La explicación de este aspecto se trata a continuación en la definición de comportamiento. Así tampoco no se deben confundir un tornillo con elementos similares que pueden desarrollar acciones distintas, como podría ser un clavo o una alcayata, que pueden desempeñar la función de colgar.

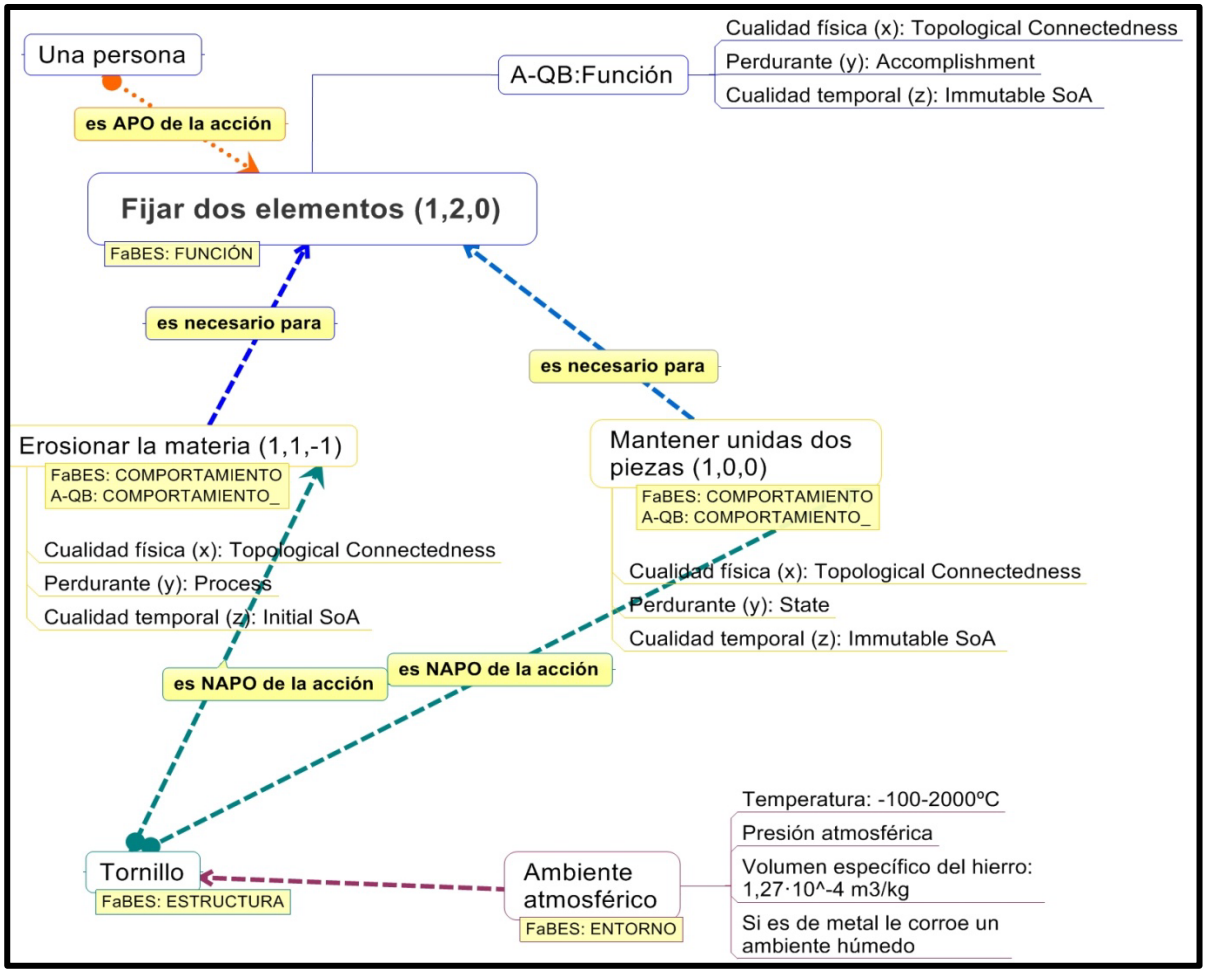


Figura 27: Esquema FaBES de un tornillo

Por otro lado, según el esquema FaBES, la función objetivo (fijar dos elementos) se muestra en la Figura 27. Ahí se puede observar la explicación de los puntos elegidos según el modelo A-QB. En el caso de la función objetivo el detalle según el A-QB es (1, 2, 0), es decir, una persona con un tornillo fija dos elementos:

Endurants: Una persona, tornillo y dos elementos. En el primer caso, la persona es el APO de la función y los restantes *endurant*s son NAPO (Figura 27).

Cualidad física (X): *Topological Connectedness* (1)*;* ya que el hecho de fijar consiste en un contacto de con el suelo.

***Perdurant* (Y):** Accomplishment (2); al fijar ser una acción dependiente del tiempo que le cueste a la persona, es decir, dicho cumplimiento ocupa una fracción del tiempo.

Cualidad temporal (Z): *Immutable SoA* (0); ya que el *perdurant* mantiene su estado.

[ff] Para una mayor aclaración al respecto se puede revisar el capítulo 3.1.3.

Comportamiento

En este caso, se deben tener en cuenta los dos casos nombrados en el apartado anterior. A la hora de considerar el tornillo como un diseño en sí mismo, los comportamientos que están ligado a la función de fijar son la erosión del material que según el modelo A-QB el valor del comportamiento citado es (1, 1,-1), es decir, el tornillo erosiona el material:

***Endurant*s**: Tornillo, material. Tanto el tornillo como el material son NAPO (Figura 25).

Cualidad física (X): *Topological Connectedness* (1)*;* ya que el hecho de erosionar consiste en un contacto del tornillo con el material*.*

***Perdurant* (Y):** Process (1); al ser una acción independiente del tiempo cuya acción no tiene un punto de finalización natural, pues depende del usuario del tornillo que indique cuánto va a erosionar el material.

Cualidad temporal (Z): *Initial SoA* (-1); ya que el *perdurant* actúa sustrayendo el *endurant*, es decir, la materia del susodicho material.

Si se considera el tornillo en un diseño conformado por varios elementos, el comportamiento que tiene entonces es el de mantener unidas dos piezas, que según el modelo A-QB el valor del comportamiento citado es (1, 0, 0), es decir, el tornillo mantiene unidas dos piezas:

***Endurant*s**: Tornillo, dos piezas. Tanto el tornillo como las dos piezas son NAPO (Figura 25).

Cualidad física (X): *Topological Connectedness* (1)*;* ya que el hecho de unir consiste en un contacto del tornillo con las dos piezas*.*

***Perdurant* (Y):** State (0); al ser una acción independiente del tiempo cuya acción es mantenida durante un tiempo indefinido, pues depende del usuario del tornillo que indique cuánto tiempo va a mantener las dos piezas en contacto.

Cualidad temporal (Z): *Immutable SoA* (0); ya que el *perdurant* mantiene su estado.

Estructura

La estructura del tornillo, con la punta en forma de torno, permite tanto erosionar la superficie de un material como facilitar la adhesión a una superficie determinada (Figura 27). Así pueden surgir diferentes opciones variando el tipo de rosca, diámetro y longitud de roscado, así como diferente cabeza dependiendo de la herramienta a la cual esté unida (Figura 28).

Figura 28: Diferentes modelos de tornillo

Entorno

En este tercer caso, las condiciones del entorno condicionan la elección de los materiales conformantes de la estructura (Figura 25). Las variables físicas que se han tenido en cuenta son las siguientes:

Temperatura: (*-100)-2000 ºC.*

Presión*: Atmosférica.*

Volumen específico de un tornillo de metal*: 1,27·10^{-4} m^3/kg.*

Los tornillos más utilizados son metálicos, con lo cual un ambiente húmedo, implica que su vida útil se reduzca debido a la corrosión del aire, y que sea necesaria en muchos casos su protección por diferentes medios, o incluso la imposibilidad de su uso, en los casos que se produzca corrosión galvánica.

3.1.9.4 PORTAMINAS

Acción

La acción fundamental es la experimentación de la escritura con minas. Según el modelo A-QB aunado al esquema FaBES, análogamente al caso anterior del apartado 3.1.9.1, las acciones del diseño de un portaminas se muestran en la Figura 29.

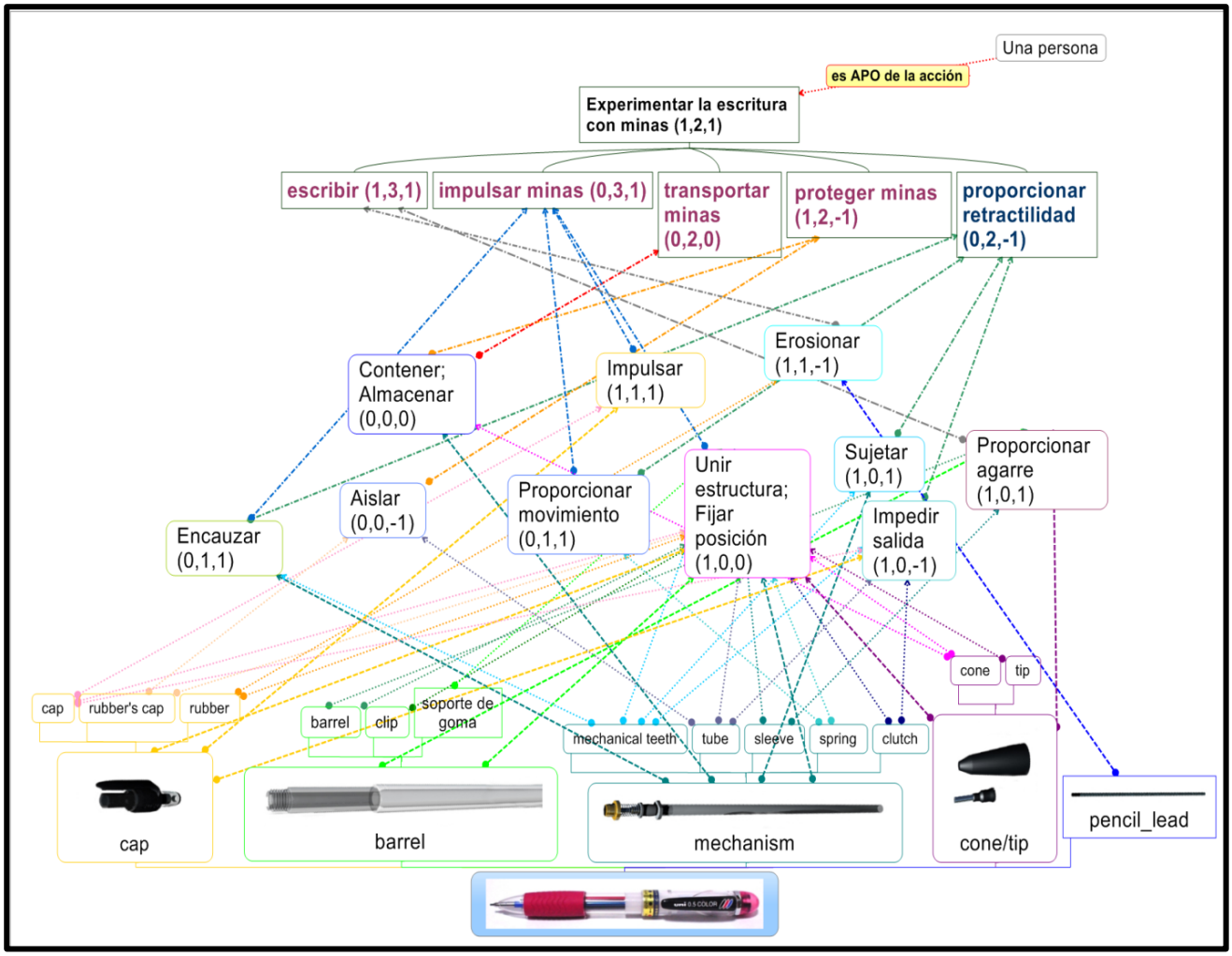


Figura 29: Esquema FaBES de un portaminas

Función

Según el esquema FaBES la función objetivo (experimentar la escritura con minas) se muestra en la Figura 29. Ahí se puede observar la explicación de los puntos elegidos según el modelo A-QB. En el caso de la

función objetivo el detalle según el A-QB es (1, 2, 1), es decir, una persona con minas experimenta la escritura:

Endurants: Una persona, un portaminas. En el primer caso, la persona es el APO de la función y el portaminas es el NAPO (Figura 29).

Cualidad física (X): *Topological Connectedness* (1)*;* ya que el hecho de experimentar la escritura consiste en un contacto de la mina del portaminas con una superficie.

***Perdurant* (Y):** Accomplishment (2); al fijar ser una acción dependiente del tiempo que le cueste a la persona, es decir, dicho cumplimiento ocupa una fracción del tiempo.

Cualidad temporal (Z): *Terminal SoA* (1); ya que las propiedades deseadas se obtienen al final del proceso.

Comportamiento

En el caso del portaminas, los comportamientos ligados a sus diversas funciones están relacionados con variados comportamientos. Cabe reseñar en este caso que todos los comportamientos en este caso detallado cumplen las premisas del modelo FaBES, es decir que los *perdurant*s son procesos o estados cuyas acciones están realizadas por NAPOs. Igualmente se puede observar cómo dentro del rango fijado se pueden hallar ejemplos de gran parte de las posibilidades existentes entre elementos del modelo A-QB.

Estructura

El portaminas está constituido por los ensamblajes siguientes: *cone/tip* (cono/punta), *mechanism* (mecanismo), *barrel* (cañón) y *cap* (tapa) y formado por las siguientes partes: *barrel* (caña), *cap* (tapa), *clutch* (agarre), *sleeve* (manga), *spring* (muelle), *mechanism teeth* (diente mecánico), *tip* (punta) y *tube* (tubo) (Figura 30).

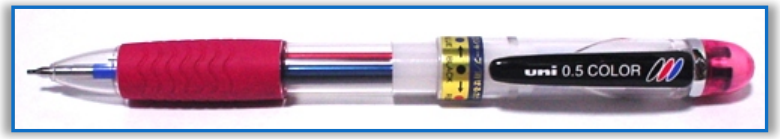


Figura 30: Ejemplo de la estructura de un portaminas

Entorno

En este segundo caso, las condiciones del entorno no condicionan la elección de los materiales conformantes de la estructura. Las variables físicas que se han tenido en cuenta son las siguientes:

Temperatura: *15-30 ºC.*

Presión*: Atmosférica.*

Volumen específico de una mina*: 1,40·10^{-4} m^3/kg.*

3.1.10 Discusión

En este apartado ante la necesidad de crear un nuevo marco de diseño funcional basado en el marco FBS que pudiese ser formalizado posteriormente en una ontología manteniendo sus características propias y rigurosidad formal, se ha generado FaBES. Dicho modelo añade 2 capas nuevas a la estructura de función-comportamiento-estructura: acción y entorno.

El motivo de añadir el concepto de acción viene como consecuencia lógica de considerar el diseño como un proceso de comunicación (Crilly, 2006). Si se plantea dicha hipótesis, las acciones de un diseño estarán conformadas por premisas generadas ante el (re)diseño de un producto que resuelva las necesidades de un público objetivo. Y dichas premisas se habrán comunicado mediante un lenguaje natural que posteriormente se transformará en funciones y comportamientos ligados a unas posibles estructuras que vendrán restringidas por el entorno donde se ubique el producto creado. Estas acciones mostradas en forma de verbo serán así funciones o comportamientos dependiendo de si el sujeto de dichas acciones es

un agente inteligente o es un objeto no inteligente, generalmente considerado como componente de la estructura.

En dicho marco se incluye una evolución del trabajo del B-Cube creando el modelo de acciones A-QB que precisamente desestructura el modelo B-Cube en acciones que dependiendo de si su finalización es natural o no, serán funciones o comportamientos, respectivamente. Cabe explicar que este razonamiento es coherente con la definición previa de las capas de función y comportamiento, pues solo un ente inteligente es capaz de establecer una finalización no natural de una acción. Para clarificar este aspecto, se puede uno remitir al ejemplo presentado de la aspiradora o el ejemplo de una tostadora, la cual hay un dispositivo automático que determina la acción de calentar una tostada. Sin embargo, en ambos casos, hay un agente inteligente que va a determinar la duración de la acción. Sin él, la tostadora no funcionaría. Y considerando el caso de que lo hiciera, entonces se convertiría en un agente inteligente (pues no necesariamente debe ser un ser humano).

La capa de entorno se establece en FaBES, tal como se ha indicado como necesidad de considerar las condiciones del ambiente donde se llevan a cabo las acciones de un producto como factores relevantes a la hora de diseñarlo (Visser, 2009). Para explicitar dicha relevancia en el apartado 3.1.3 se introduce la Ecuación [1] donde se muestra cómo la variación de las condiciones del entorno afecta al diseño del producto.

Finalmente, se han planteado una serie de ejemplos para poder mostrar el funcionamiento de FaBES.

3.2 OntoFaBES

La ontología OntoFaBES surge como la adaptación del marco FBS para facilitar su modelado como una ontología ante la ausencia de modelos similares en la literatura (capítulo 2.5). Adquiere su marco formal de FaBES, tomando como referencia las nociones definidas en la metaontología DOLCE, desarrollada usando la herramienta Protégé, herramienta informática utilizada para su modelado, escrita en el lenguaje OWL y utilizando el lenguaje SWRL para inferir las reglas de conocimiento (Aptdo. 2.4.7).

Esta ontología se ha diseñado a través de un profundo análisis de la naturaleza de los conceptos, apoyada con una base de razonamiento lógico gracias a la utilización de DOLCE. De esa manera se fundamenta como una base estable para formalizar la información de modelos de diseño evitando incoherencias lógicas y permitiendo explicitar todo el conocimiento posible.

Cabe indicar además que hubo colaboración de diferentes expertos en el ámbito: diseñadores, ingenieros informáticos, un centro tecnológico especializado en diseño de producto y personal de una empresa de fabricación de productos.

En este capítulo, primero se muestra el esquema inicial de la ontología, para a continuación indicar para cada capa del esquema formal de OntoFaBES, las clases y propiedades para definirlo de forma adecuada. Y por último, se indican las reglas utilizadas para la inferencia de conocimiento. Para ello se utiliza el lenguaje SQWRL[gg].

3.2.1 Esquema general

El árbol de clases general (Figura 31) se estructura según la organización formal de FaBES: *Structure*[hh], *Action, Function, Behavior y Environment*. De la misma manera se añaden dos clases más necesarias para la correcta representación del conocimiento en el esquema creado: *Material* y *Constraint*.

[gg] SQWRL es un lenguaje de búsqueda basado en SWRL que puede ser usado para hacer consultas en ontologías OWL.

[hh] Se debe indicar al lector de la tesis que tanto en la ontología OntoFaBES como en su aplicación a los casos prácticos y en el capítulo 5, la gran mayoría de las referencias utilizadas van a ser en inglés debido a que OntoFaBES sea reusable por el mayor número de personas posibles.

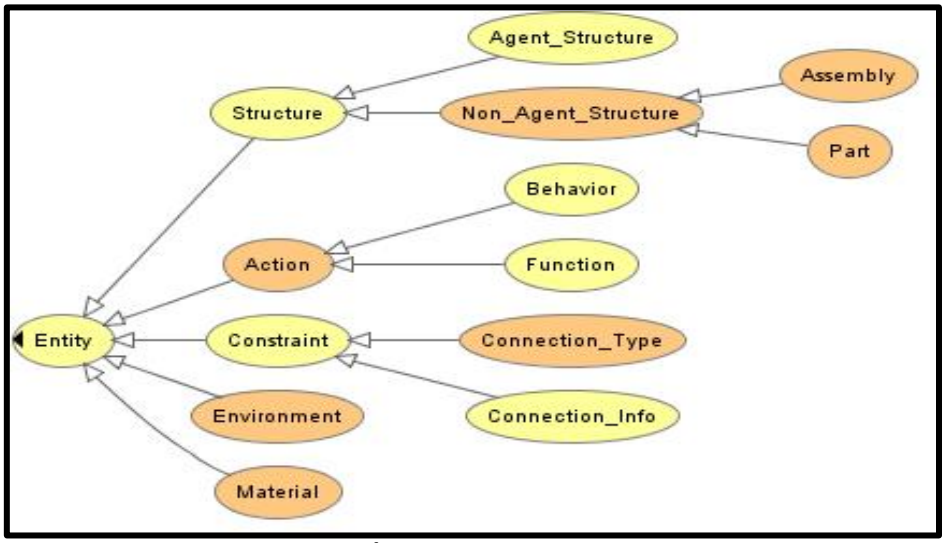


Figura 31: Árbol de clases de OntoFaBES

El motivo de esta adición es que para la correcta definición de la estructura de un producto es necesario conocer tanto su material como las restricciones entre las piezas que lo conforman. Por tanto, por consistencia con DOLCE (Figura 11) que dentro de la metaontología, considera el *Amount of Matter, (*y *Material* como subclase de ella), una clase diferente a la de un APO, en OntoFaBES se mantiene esa coherencia manteniendo *Material* como una clase disjunta a la de *Estructura*. De la misma forma, la clase *Constraint* es disjunta de la clase *Structure* ya que está conformada por instancias que son características (clase *Feature* según *DOLCE*), disjunta de los NAPOs y APOs, es decir, de la clase de *Physical Object* que los agrupa.

A continuación se explican en profundidad cada una de las clases creadas en OntoFaBES. Para su comprensión, cada apartado indicado en el esquema formal se indica primero parafraseado en lenguaje natural su significado, a continuación se muestran las subclases y propiedades utilizadas para con su descripción, respectivamente.

Cabe indicar que en la descripción se utiliza la Tabla 8 con el objetivo de comprender la analogía entre el lenguaje OWL y su traducción a lenguaje natural.

Del mismo modo, para las capturas de pantalla referidas a la definición de cada una de las clases de la ontología a través de la herramienta Protégé se utiliza la visión lógica proporcionada por dicha herramienta (Figura 32) donde se observa en la sección izquierda la **jerarquía de clases**; en la sección superior derecha, los **comentarios** donde se indican los diferentes comentarios de la clase; en la sección media derecha, se indican las **restricciones** que tiene la clase para su definición; y en la sección inferior derecha, las **clases disjuntas**.

Para el apartado de propiedades, se debe indicar que se han considerado las propiedades objeto y las propiedades de tipo de datos. Dichas propiedades se heredan de la formalización del conocimiento realizada en la herramienta KSS 2.0 tal como se explica en el apartado 4.3.3.

Las relaciones de las propiedades objeto de la ontología OntoFaBES se indican en la Figura 33, Figura 34 y Figura 35. En la Figura 33 se muestra que para cada clase principal de OntoFaBES se indica un cuadro descriptivo donde se indica su **URI**, **clase** a la que pertenece (*Superclasses*), **clases disjuntas** (*Disjoint classes*), y **anotaciones** (*Annotations*) además de las relaciones de las propiedades objeto indicadas por flechas de diferente color que las relaciona. Así mismo, en la Figura 34 se muestran dichas propiedades objeto tal como se visualizan en el editor Protégé. Finalmente, en la Figura 35 se muestran las propiedades objeto acompañadas de la información relativa a su **funcionalidad** (*Func*), **simetría** (*Sym*), **funcionalidad inversa** (*Inv Func*) además de su **dominio** (*Domain*), **rango** (*Range*) y **propiedad inversa**.

Mientras, las propiedades de tipo de datos se muestran en la Figura 36. En dicha figura se puede observar su organización acompañada de su **funcionalidad** (*Func*), **dominio** (*Domain*) y **rango** (*Range*).

A continuación se exponen los distintos apartados que conforman la descripción ontológica de la capa de acción.

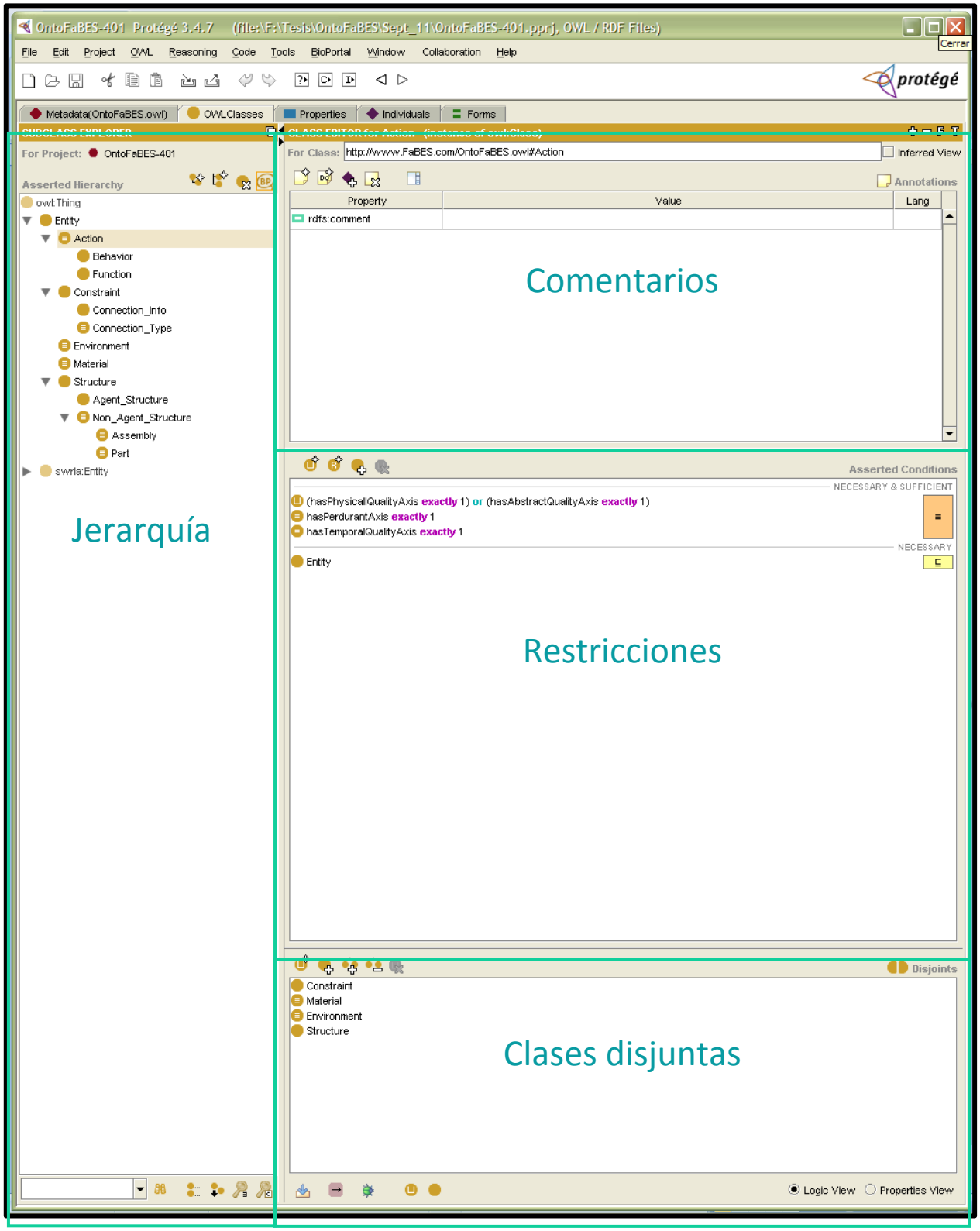


Figura 32: Captura de pantalla de OntoFaBES en Protégé del apartado de Clases

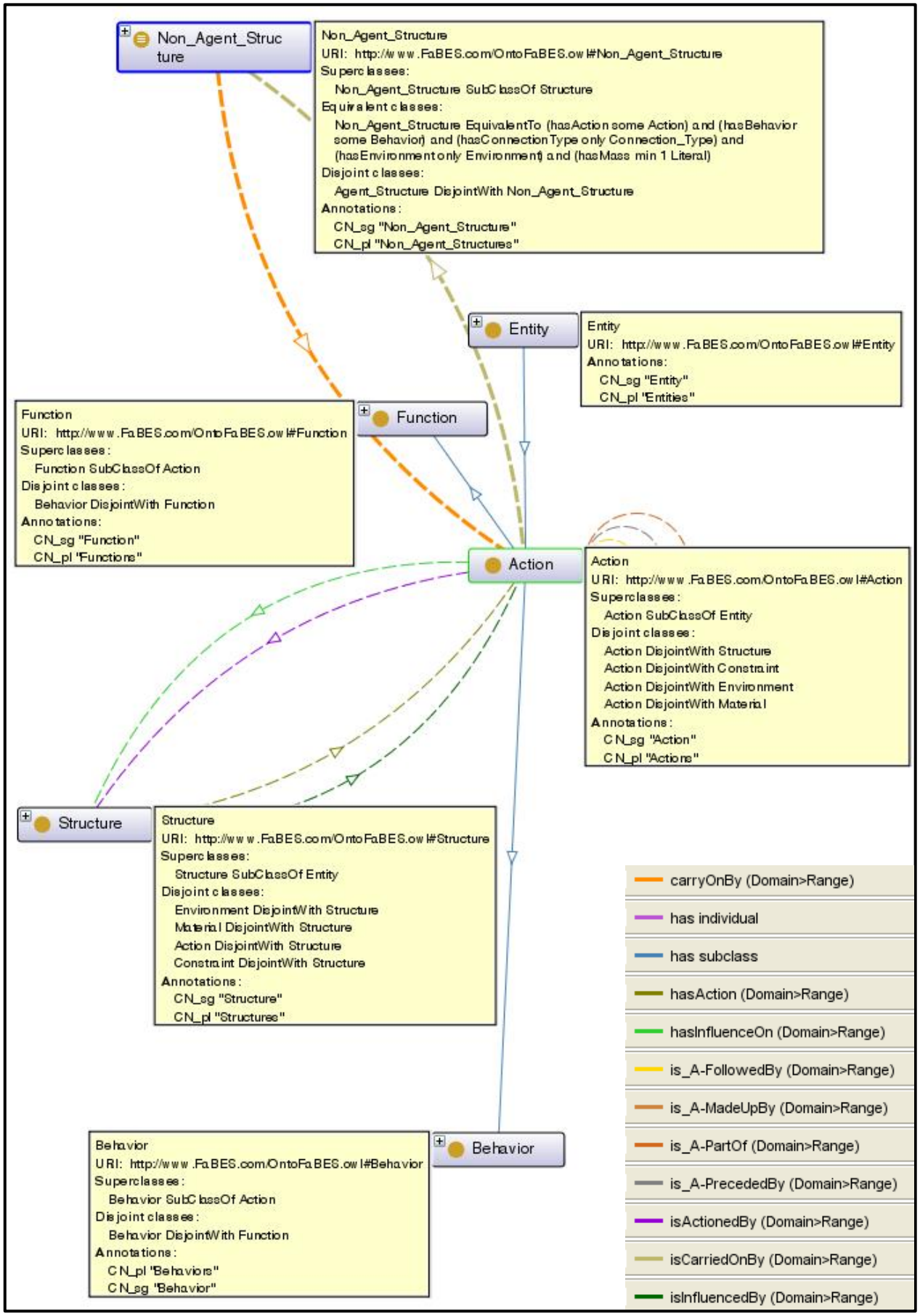


Figura 33: Propiedades relativas a las relaciones entre clases de OntoFaBES

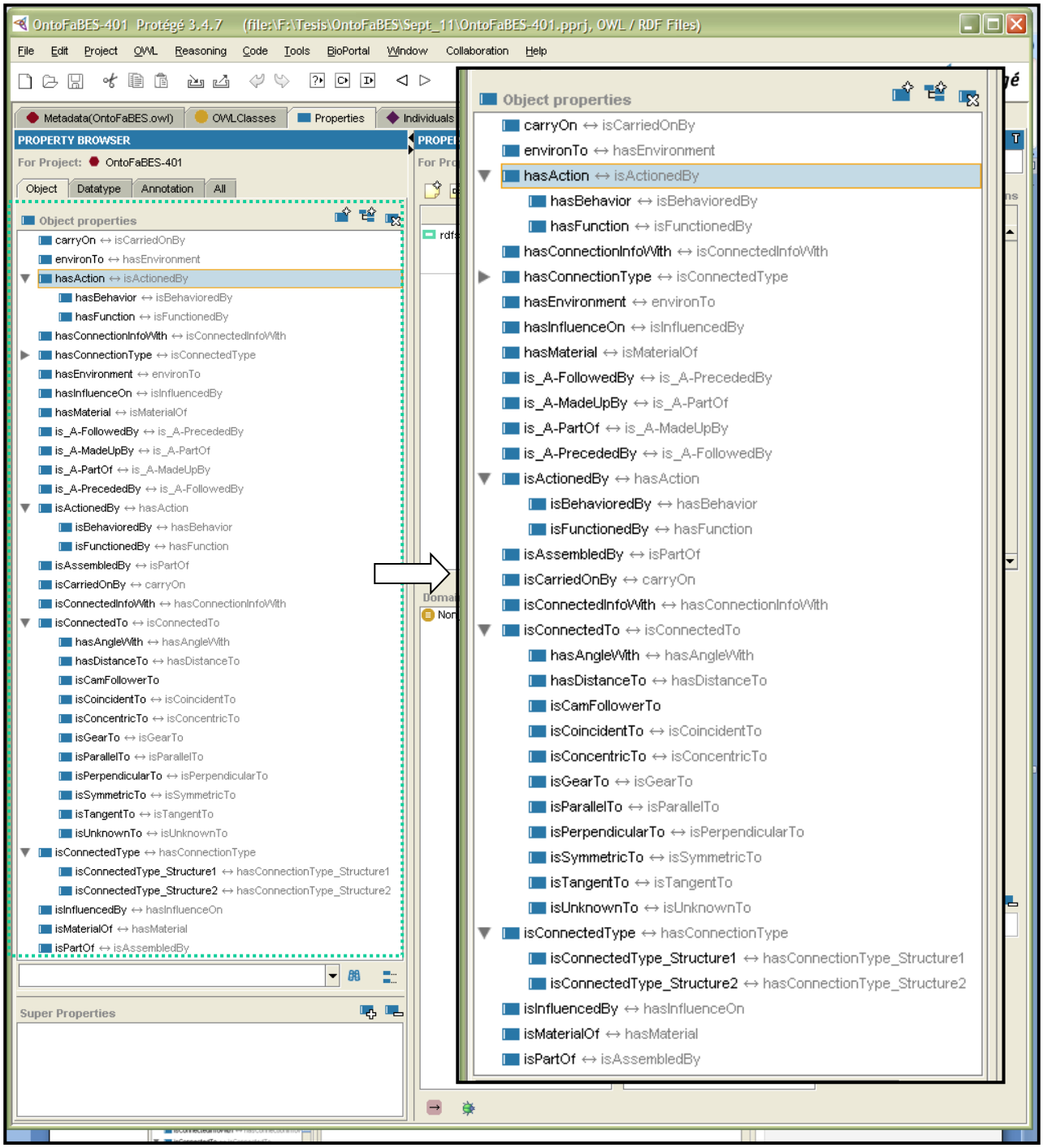


Figura 34: Captura de pantalla de OntoFaBES en Protégé del apartado de Propiedades

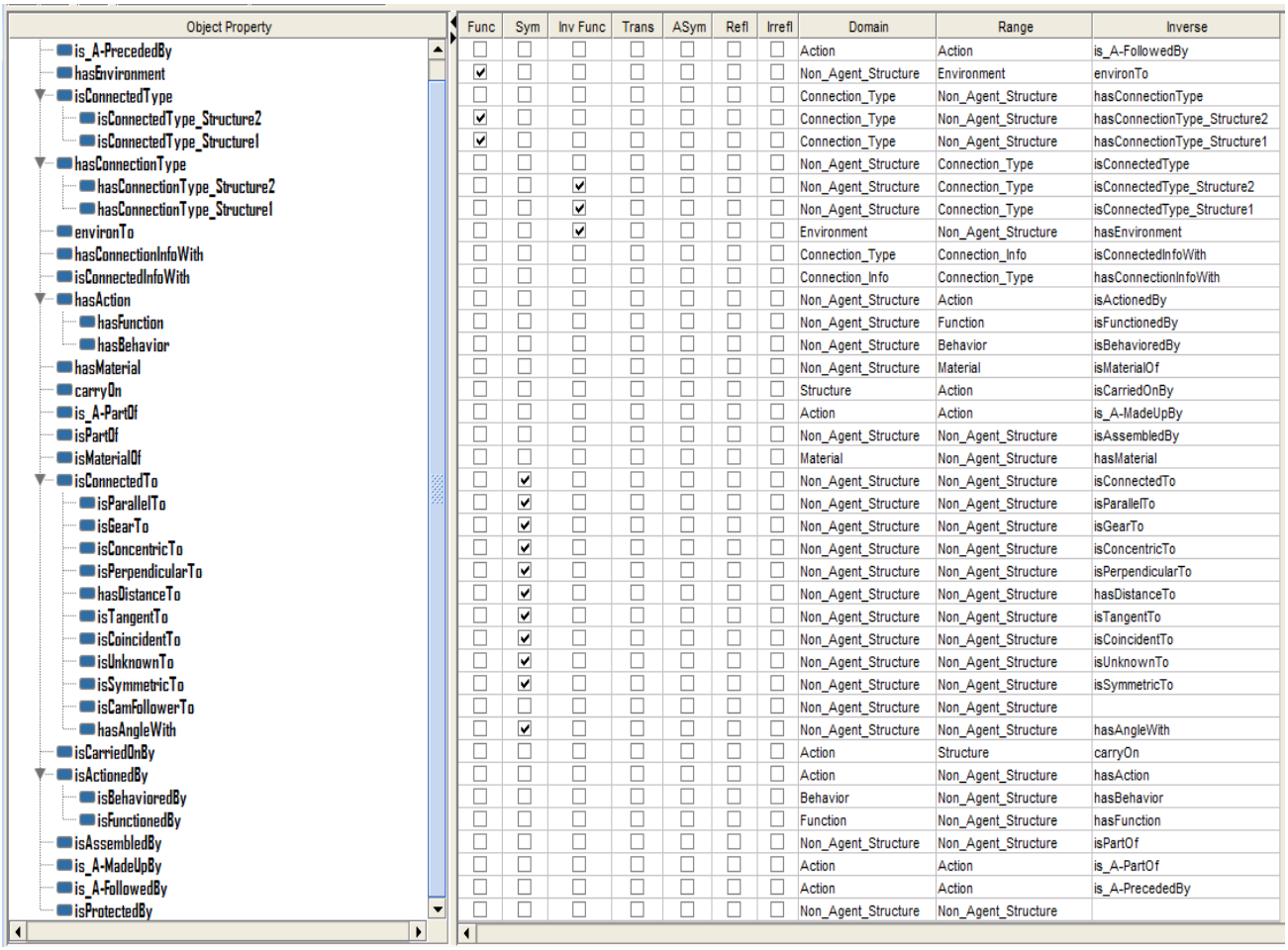


Figura 35: Captura de pantalla de OntoFaBES en Protégé del apartado de Propiedades objeto

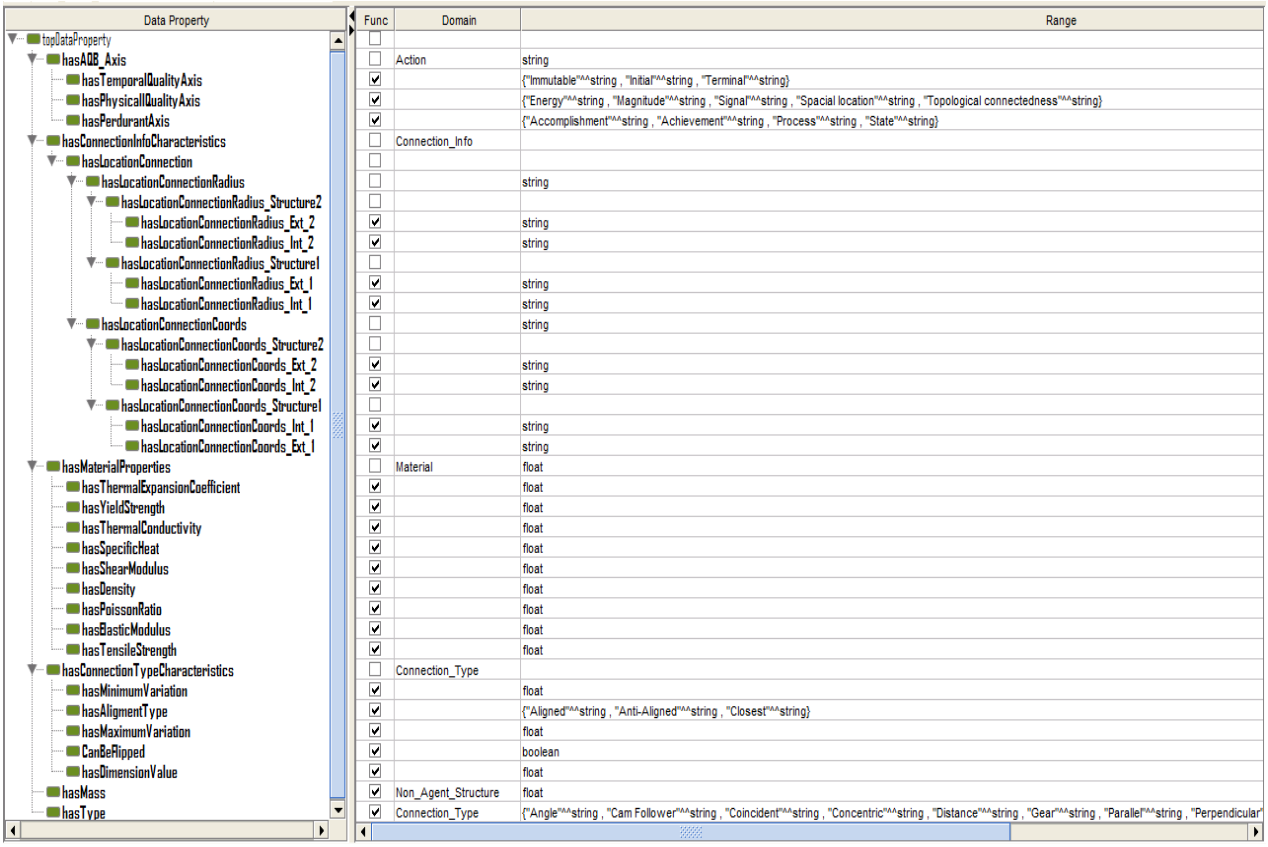


Figura 36: Captura de pantalla de OntoFaBES en Protégé del apartado de Propiedades de datos

3.2.2 Capa de acción

Se representa por la clase *Action*.

La definición proporcionada en la ontología OntoFaBES, se hereda de la mostrada en FaBES en los apartados 3.1.3 y 3.1.4. Por tanto, una acción es considerada un *perdurant* que, entre otras cosas, se ejecuta por un objeto físico, agente o no y está adscrito como mínimo a una estructura. Además tiene un valor en el eje de cualidad física, tiene un valor en el eje de *perdurant* y un valor en el eje de cualidad temporal.

Según el modelo del A-QB, las acciones dependen de la situación en el A-QB dependiendo del valor en los ejes *Cualidad física*, *Perdurant* y *Cualidad Temporal* según las descripciones indicadas en el apartado 3.1.4.

La Figura 37 da una descripción de las restricciones aplicadas a la clase ***Action*** mediante pseudolenguaje OWL complementada con la correspondiente paráfrasis. En dicha figura, como en las siguientes donde se muestra el pseudolenguaje OWL, para la correspondiente paráfrasis se utiliza la información recogida en la Tabla 8.

En la Figura 38 se muestra la presentación de información de la clase *Action* donde se observa el árbol de clases indicado en la Figura 32 y las restricciones indicadas en la definición. Esta información se muestra en lenguaje OWL en la Figura 39. En ambas figuras se observa en la parte de comentarios su definición; en el apartado de clases disjuntas se indican las clases que cumplen dicha condición y en el apartado de restricciones se puede ver que la acción tiene 3 restricciones de tipo *necesario y suficiente* y una de tipo *suficiente* de forma coherente a los mostrado en la Figura 37.

OWL:

clase (Action complete

restricción ((hasPhysicallQualityAxis exactly 1) or (hasAbstractQualityAxis exactly 1))

restricción (has*Perdurant*Axis exactly 1)

restricción (hasTemporalQualityAxis exactly 1)

clase (Action partial OntoFaBES)

Paráfrasis: Las acciones son *perdurant*s en OntoFaBES, entre otras cosas, se ejecuta por un objeto físico, agente o no, está adscrito como mínimo a una estructura. Además tiene un valor en el eje de cualidad física, tiene un valor en el eje de *perdurant* y un valor en el eje de cualidad temporal.

Figura 37: Restricciones de la clase Action

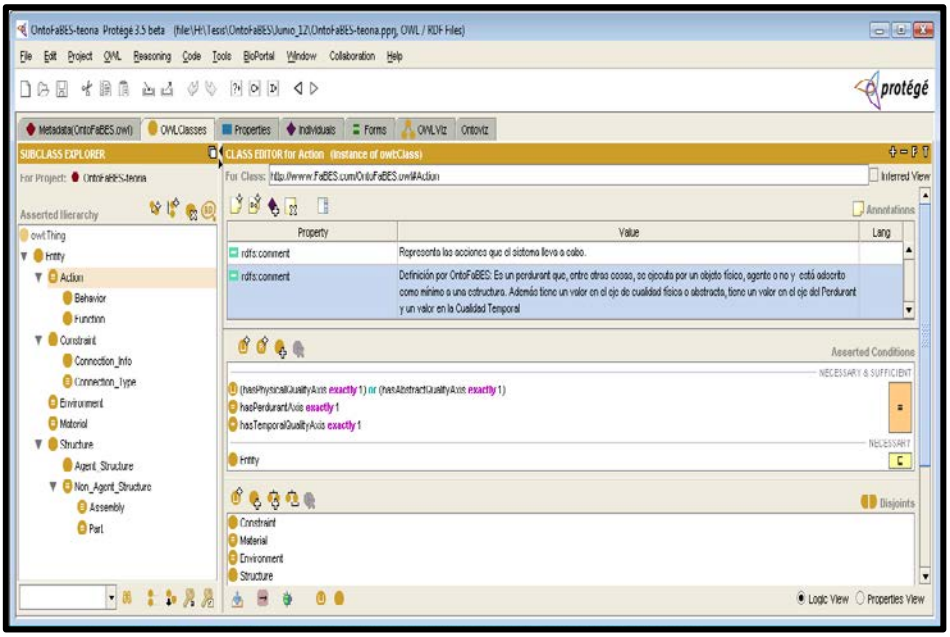


Figura 38: Captura de pantalla de OntoFaBES en Protégé de la clase Action

```
<owl:Class rdf:ID="Action">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
              <owl:Restriction>
                <owl:onProperty rdf:resource="#hasAbstractQualityAxis"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
              </owl:Restriction>
              <owl:Restriction>
                <owl:onProperty rdf:resource="#hasPhysicallQualityAxis"/>
                <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
              </owl:Restriction>
            </owl:unionOf>
          </owl:Class>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#has*Perdurant*Axis"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasTemporalQualityAxis"/>
            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#Entity"/>
    <owl:disjointWith rdf:resource="#Constraint"/>
    <owl:disjointWith rdf:resource="#Environment"/>
    <owl:disjointWith rdf:resource="#Material"/>
    <owl:disjointWith rdf:resource="#Structure"/>
    <rdfs:comment rdf:datatype="&xsd;string"
      >Definición por OntoFaBES: Es un *perdurant* que, entre otras cosas, se ejecuta por un objeto físico agente o no, está adscrito como mínimo a una estructura. Además tiene un valor en el eje de cualidad f&#237;sica, tiene un valor en el eje del *Perdurant* y un valor en la Cualidad Temporal</rdfs:comment>
    <rdfs:comment rdf:datatype="&xsd;string"
      >Representa las acciones que el sistema lleva a cabo.</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="Agent_Structure">
    <rdfs:subClassOf rdf:resource="#Structure"/>
    <owl:disjointWith rdf:resource="#Non_Agent_Structure"/>
  </owl:Class>
```

```
  <owl:Class rdf:ID="Assembly">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#isAssembledBy"/>
            <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">2</owl:minCardinality>
            <owl:valuesFrom rdf:resource="#Non_Agent_Structure"/>
          </owl:Restriction>
          <owl:Class rdf:about="#Non_Agent_Structure"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Part"/>
  </owl:Class>
<owl:Class rdf:ID="Action">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
```

Figura 39: Clase Action en lenguaje OWL

3.2.2.1 Propiedades de la clase Action

En el caso de la clase *Action* hay propiedades objeto y de tipo de datos.

La Figura 40 representa las propiedades de la clase Action:

- **is_A-FollowedBy**: Indica que una acción está seguida por otra acción.
- **is_A-MadeUpBy**: Indica que una acción está compuesta por otra acción.
- **is_A-PartOf**: Indica que una acción es parte de otra acción
- **is_A-PrecededBy:** Indica que una acción está precedida por otra acción.
- **isActionedBy**: Indica que una acción es accionada por un APO.
- **isCarriedOnBy:** Indica que una acción es llevada a cabo por una estructura.
- **hasAQB_Axis:** Indica que una acción tiene un punto en el eje según el modelo A-QB. Eso genera 3 subpropiedades
    - **hasTemporalQualityAxis**: Indica que una acción tiene un punto en el eje de cualidad temporal del modelo A-QB.
    - **hasPhysicalQualityAxis:** Indica que una acción tiene un punto en el eje de cualidad física del modelo A-QB.
    - **has*Perdurant*Axis:** Indica que una acción tiene un punto en el eje de *perdurant* del modelo A-QB.

Las características de las propiedades objeto quedan resumidas en la Tabla 11[ii] y las de tipo de datos en la Tabla 12. Se pueden observar en la Figura 40.

[ii] Para evitar confusiones con respecto al significado de dominio, rango y rol en una ontología se recomienda revisar el capítulo 2.4.1.

Tabla 11: Descripción de las propiedades objeto de la clase Action.

Propiedad de Objeto	Dominio	Rango	Rol	Propiedad Inversa
is_A-FollowedBy	Action	Action	-	is_A-PrecededBy
is_A-MadeUpBy	Action	Action	-	is_A-PartOf
is_A-PartOf	Action	Action	-	is_A-MadeUpBy
is_A-PrecededBy	Action	Action	-	is_A-FollowedBy
isActionedBy	Action	APO	Transitiva	hasAction
isCarriedOnBy	Action	Structure		carryOn

Tabla 12: Descripción de las propiedades de tipo de datos de la clase Action.

Propiedad de tipo de datos	Dominio	Rango	Rol	Valores permitidos
hasAQB_Axis	Action	string	-	-
hasPhysicalQualityAxis	Action	string	Functional	{Energy, Magnitude, Spatial location, Topological connectedness}
hasTemporalQualityAxis	Action	string	Functional	{Immutable, Initial, Terminal}
hasPerdurantAxis	Action	string	Functional	{Accomplishment, Achievement, Process, State}

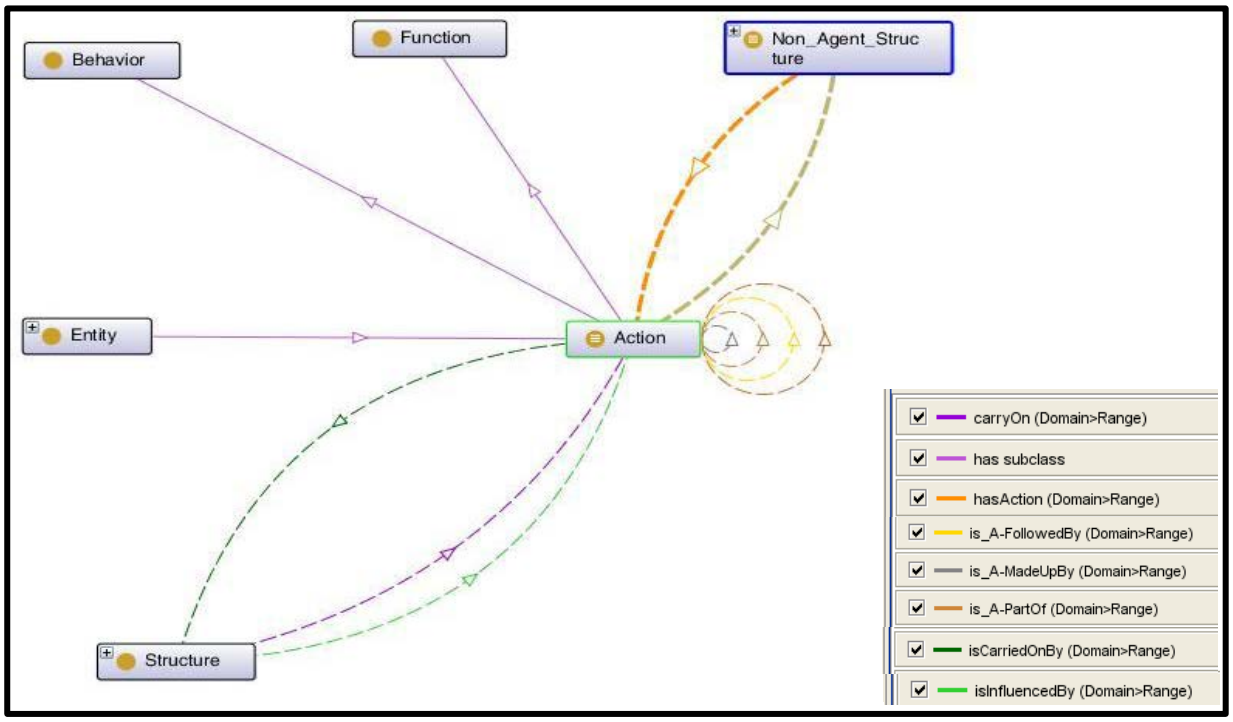


Figura 40: Representación gráfica de las propiedades de la clase Action

3.2.3 Capa de función

Se representa por la clase *Function,* subclase de *Action*.

Según la definición proporcionada en la ontología OntoFaBES, una función se considera un *Perdurant*, entidad que está presente temporalmente, cuya acción, entre otras cosas, es realizada por un APO, objeto

agente físico, que está ligada a un comportamiento y que necesita otra función para ejecutarse o es necesaria para que se ejecute otra función.

Esta clase es disjunta de *Behavior* ya que una instancia que sea un comportamiento nunca es una función.

En el esquema FaBES, las acciones son *perdurants*, y a partir de su definición, se puede deducir que una función es un punto en el eje del modelo A-QB, según las descripciones indicadas en el Apdo. 0.

En la Figura 41 se muestra la presentación de información donde se observa el árbol de clases indicado en la Figura 13 y las restricciones indicadas en la definición. Esta información se muestra en lenguaje OWL en la Figura 42.

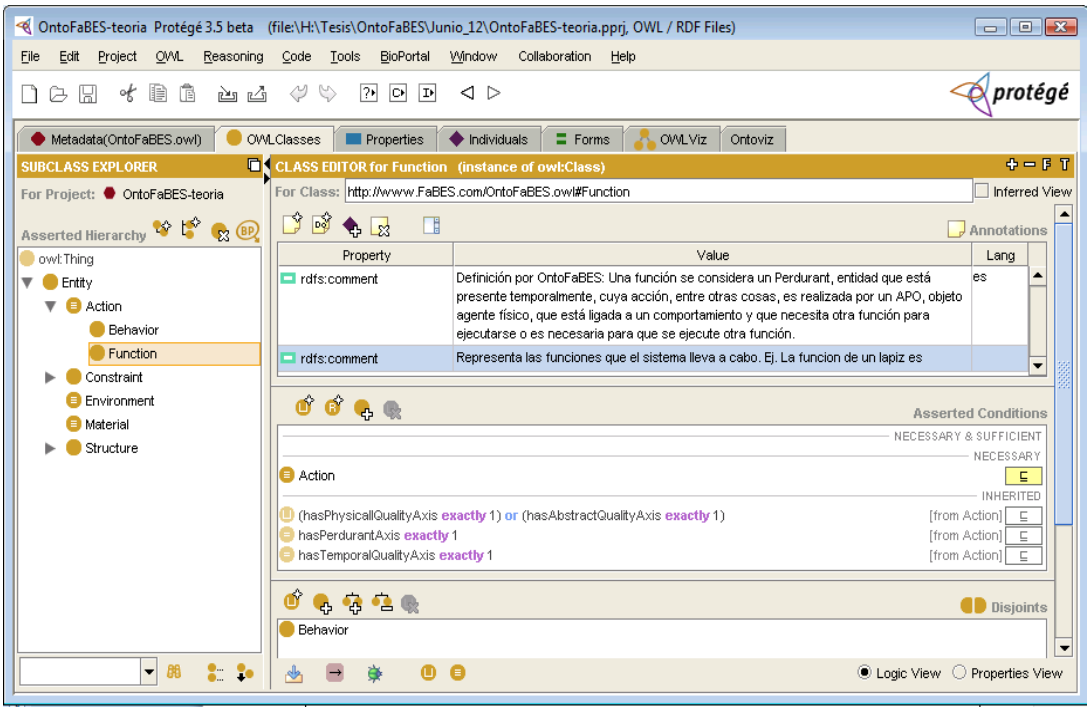


Figura 41: Captura de pantalla de OntoFaBES en Protégé de la clase Function

La Figura 43 da una descripción de las restricciones aplicadas a la clase Function mediante pseudolenguaje OWL con la correspondiente paráfrasis.

```
<owl:Class rdf:ID="Function">
<owl:Class rdf:ID="Function">
    <rdfs:subClassOf rdf:resource="#Action"/>
    <owl:disjointWith rdf:resource="#Behavior"/>
    <rdfs:comment xml:lang="es"
      >Definici&#243;n por OntoFaBES: Una funci&#243;n se considera un Perdurant, entidad que est&#225; presente temporalmente, cuya acci&#243;n, entre otras cosas, es realizada por un APO, objeto agente f&#237;sico, que est&#225; ligada a un comportamiento y que necesita otra funci&#243;n para ejecutarse o es necesaria para que se ejecute otra funci&#243;n.</rdfs:comment>
    <rdfs:comment rdf:datatype="&xsd;string"
      >Representa las funciones que el sistema lleva a cabo. Ej. La función de un lápiz es escribir.</rdfs:comment>
  </owl:Class>
```

Figura 42: Clase Function en lenguaje OWL

OWL:

clase (Function complete

restricción ((hasPhysicallQualityAxis exactly 1) or (hasAbstractQualityAxis exactly 1))

restricción (has*Perdurant*Axis exactly 1)

restricción (hasTemporalQualityAxis exactly 1)

clase (Function partial

OntoFaBES)

Paráfrasis:

Las funciones son acciones en OntoFaBES que, entre otras cosas, todas son realizadas sólo por objetos físicos agentes, funcionan por algunos comportamientos y también o necesitan otra función para ejecutarse o son necesarios para que se ejecute otra función.

Figura 43: Restricciones de la clase Function

A continuación se exponen las propiedades de la clase Function.

3.2.3.1 Propiedades de la clase Function

En el caso de la clase Function hereda las propiedades de la clase Action y añade una propiedad objeto (Figura 44):

- **isFunctionedBy**: Indica que una función está relacionada con un comportamiento.

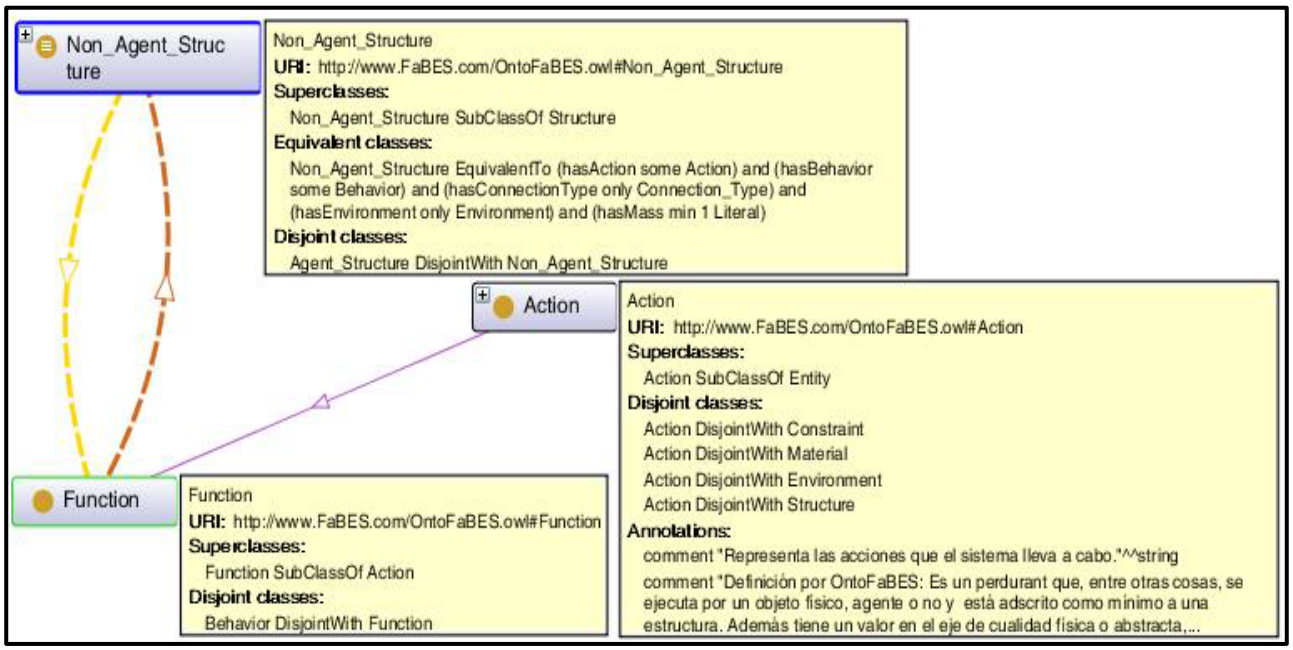


Figura 44: Relaciones de las propiedades objeto de la clase Function.

Las características de las propiedades quedan resumidas en la Tabla 13.

Tabla 13: Descripción de las propiedades de la clase Function.

Propiedad de Objeto	Dominio	Rango	Rol	Propiedad Inversa
isFunctionedBy	Function	Structure	-	hasFunction

3.2.4 Capa de comportamiento

Se representa por la clase *Behavior,* subclase de *Action*.

Según la definición proporcionada en la ontología OntoFaBES, un comportamiento se considera un *Perdurant*, cuya acción es realizada por un NAPO, objeto físico no agente, que, entre otras cosas, está ligado a la capa de funciones y a la capa estructural, y que también o necesitan otro comportamiento para ejecutarse o son necesarios para que se ejecute otro comportamiento.

Esta clase es disjunta de *Function* ya que una instancia que sea una función nunca es un comportamiento.

En el esquema FaBES, las acciones son *perdurant*s, y a partir de su definición, se puede deducir que un comportamiento es un punto en el eje del modelo A-QB, según las descripciones indicadas en el Apdo. 0.

En la Figura 45 se muestra la presentación de información donde se observa el árbol de clases indicado en la Figura 13 y las restricciones indicadas en la definición. Esta información se muestra en lenguaje OWL en la Figura 46.

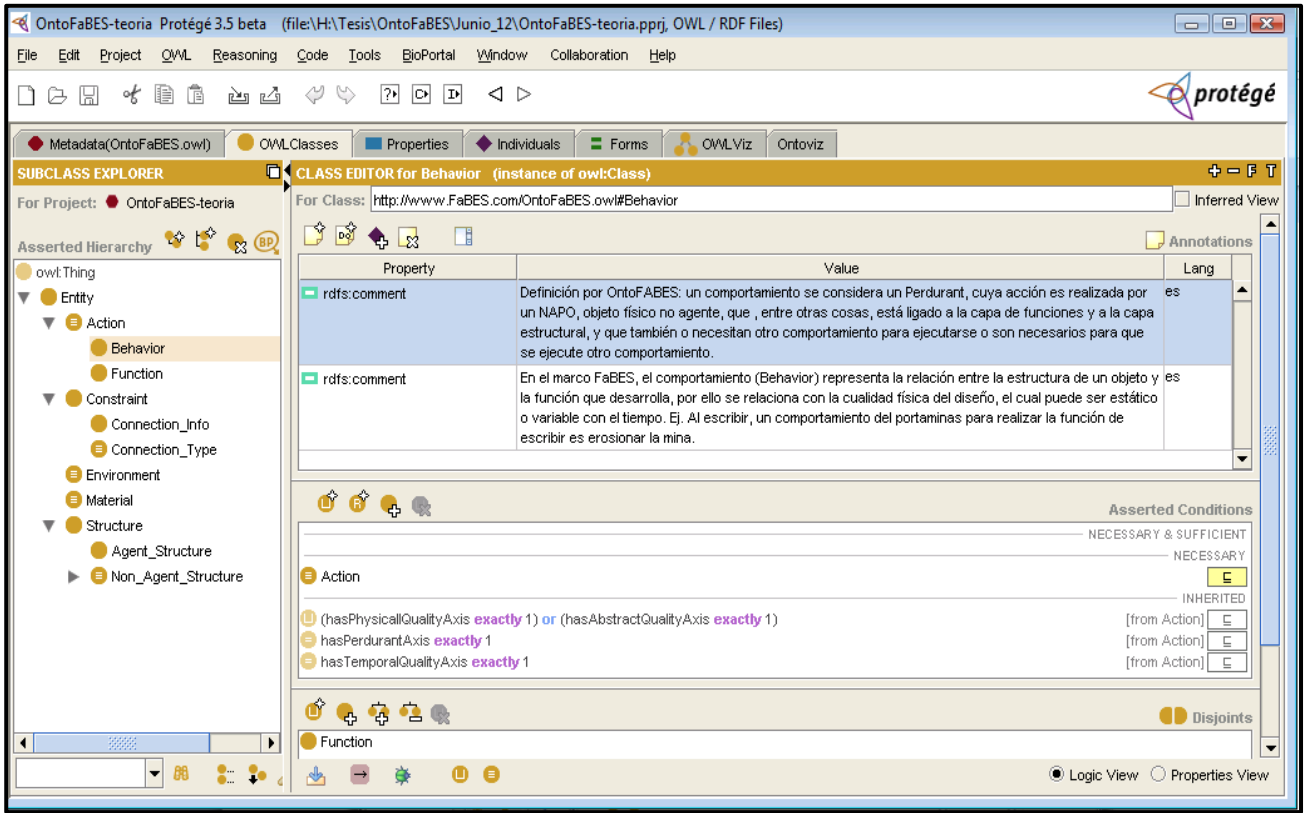


Figura 45: Captura de pantalla de OntoFaBES en Protégé de la clase Behavior

OWL:

clase (Behavior complete

restricción ((hasPhysicallQualityAxis exactly 1) or (hasAbstractQualityAxis exactly 1))

restricción (has*Perdurant*Axis exactly 1)

restricción (hasTemporalQualityAxis exactly 1)

clase (Behavior partial OntoFaBES)

Paráfrasis:

Los comportamientos son acciones en OntoFaBES que, entre otras cosas, todas tienen algunas funciones, también son realizadas sólo por objetos físicos no agentes, establecen su comportamiento por algunas estructuras y también o necesitan otro comportamiento para ejecutarse o son necesarios para que se ejecute otro comportamiento.

Figura 46: Restricciones de la clase Behavior.

A continuación se exponen las propiedades de la clase *Behavior*.

3.2.4.1 Propiedades de la clase Behavior

En el caso de la clase Behavior hereda las propiedades de la clase Action y añade una propiedad objeto (Figura 47):

- **isBehavioredBy:** Indica que un comportamiento está relacionado con una estructura**.**

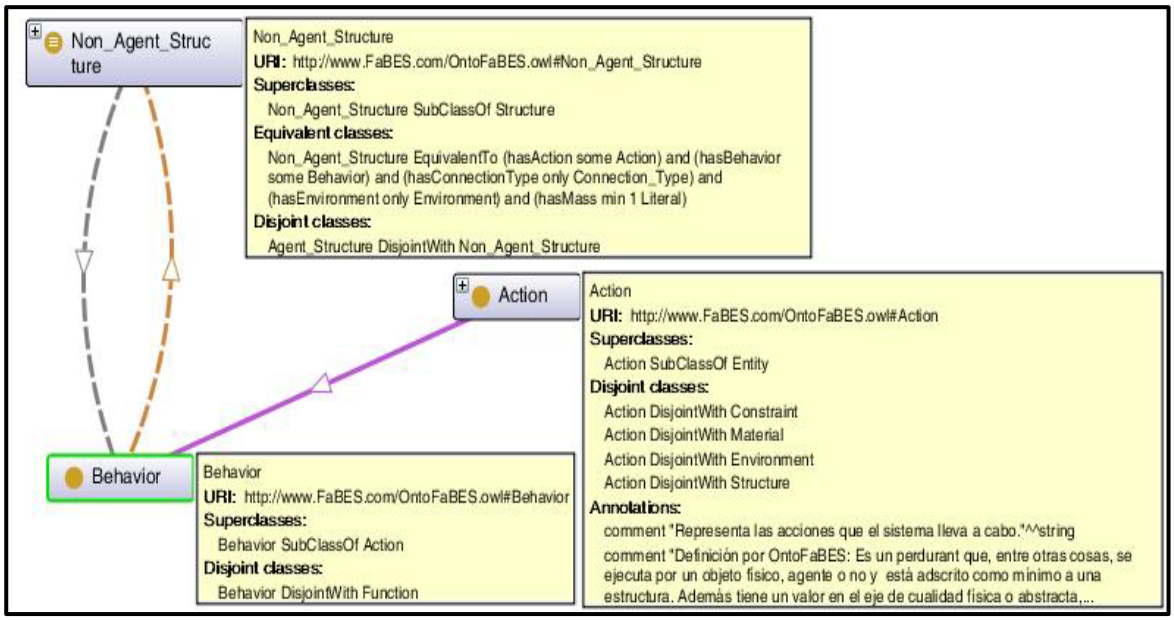


Figura 47: Relaciones de las propiedades objeto de la clase Behavior.

Las características de las propiedades quedan resumidas en la Tabla 14:

Tabla 14: Descripción de las propiedades de la clase Behavior.

Propiedad de Objeto	Dominio	Rango	Rol	Inverso
isBehavioredBy	Behavior	Structure	-	hasBehavior

3.2.5 Capa de estructura

Se representa por la clase *Structure*.

Según la definición proporcionada en la ontología OntoFaBES, una estructura se considera una entidad que tiene como subclases a una estructura agente y una no agente.

En la Figura 48 se muestra la presentación de información donde se observa el árbol de clases indicado en la Figura 13 y las restricciones indicadas en la definición.

La Figura 49 da una descripción de las restricciones aplicadas a la clase de la estructura mediante pseudolenguaje OWL con la correspondiente paráfrasis.

A continuación se exponen los distintos apartados que conforman la descripción ontológica de la capa de la estructura.

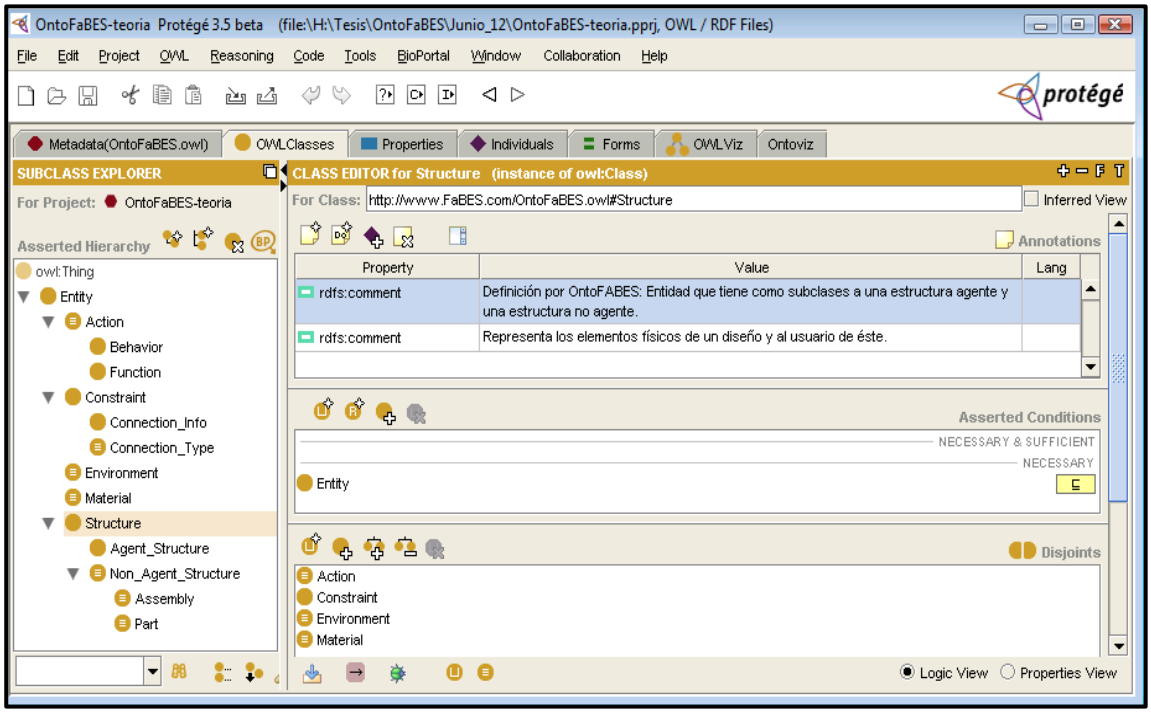


Figura 48: Captura de pantalla de OntoFaBES en Protégé de la clase Structure

OWL:

clase (Structure complete

clase (Structure partial OntoFaBES)

Paráfrasis:

Las estructuras son entidades en OntoFaBES que tienen como subclases a estructuras agentes y no agentes.

Figura 49: Restricciones de la clase Structure

3.2.5.1 Subclases de la clase Structure

En este caso, se definen dos tipos de subclases: Agent_Structure y Non_Agent_Structure, tal como se muestra en el árbol de clases indicado en la Figura 13.

Clase Agent_Structure

Una estructura agente se considera una estructura que, entre otras cosas, lleva a cabo conscientemente una acción.

La Figura 50 da una descripción de las restricciones aplicadas a la clase de la estructura agente mediante pseudolenguaje OWL con la correspondiente paráfrasis.

OWL:

clase (Agent_Structure complete

Structure)

Paráfrasis:

Las estructuras agentes son estructuras que, entre otras cosas, llevan a cabo conscientemente una acción.

Figura 50: Restricciones de la clase Agent_Structure

Clase Non_Agent_Structure

Una estructura no agente son NAPO que tienen, entre otras cosas, todas sólo un entorno de acción y un tipo de conexión, como mínimo un comportamiento, una acción y un valor de masa.

La Figura 51 da una descripción de las restricciones aplicadas a la clase de la estructura no agente mediante pseudolenguaje OWL con la correspondiente paráfrasis.

OWL:

clase (Non_Agent_Structure complete

Structure

restricción (hasAction someValuesFrom Action)

restricción (hasBehavior someValuesFrom Behavior)

restricción (hasConnectionType onlyValuesFrom Connection_Type)

restricción (hasEnvironment onlyValuesFrom Behavior)

restricción (hasMass minimum 1))

Paráfrasis:

Las estructuras no agente son NAPO que, entre otras cosas, entre otras cosas, todas sólo un entorno de acción y un tipo de conexión, como mínimo un comportamiento, una acción y un valor de masa.

Figura 51: Restricciones de la clase Non_Agent_Structure

Subclases de la clase Non_Agent_Structure

En este caso, se definen dos tipos de subclases: Assembly y Part, tal como se muestra en el árbol de clases indicado en la Figura 13.

Clase Assembly

Un ensamblaje se considera una estructura que, entre otras cosas, está ensamblada como mínimo por dos elementos estructurales.

La Figura 52 da una descripción de las restricciones aplicadas a la clase del ensamblaje mediante pseudolenguaje OWL con la correspondiente paráfrasis.

OWL:

clase (Assembly complete

Structure

restricción (isAssembledBy minCardinality 2 Structure))

Paráfrasis:

Los ensamblajes son estructuras que, entre otras cosas, todas están ensamblados por mínimo 2 estructuras.

Figura 52: Restricciones de la clase Assembly

Clase Part

Una parte se considera una estructura, que, entre otras cosas, es parte de algún ensamblaje y que está formado por algún material.

La Figura 53 da una descripción de las restricciones aplicadas a la clase del ensamblaje mediante pseudolenguaje OWL con la correspondiente paráfrasis.

OWL:

clase (Part complete

Structure

restricción (hasMaterial someValuesFrom Material)

restricción (isPartOf someValuesFrom Assembly))

Paráfrasis:

Las partes son estructuras que, entre otras cosas, todas son partes de un ensamblaje y también tienen algún material (en su composición).

Figura 53: Restricciones de la clase Part

A continuación se exponen las propiedades de la clase Structure y de sus subclases (Figura 54).

3.2.5.2 Propiedades de la clase Structure

En el caso de la clase Structure, existe una propiedad objeto

- **carryOn:** Indica que una estructura está llevada a cabo por una acción**.**

Las características de las propiedades quedan resumidas en la Tabla 15:

Tabla 15: Descripción de las propiedades de la clase Structure.

Propiedad de Objeto	Dominio	Rango	Rol	Inverso
carryOn	Structure	Action	-	isCarriedOnBy

Cabe indicar que la subclase *Agent_Structure* hereda la propiedad de la clase *Structure*.

Propiedades de la clase Non_Agent_Structure

En este caso, la clase *Non_Agent_Structure* se define tanto por propiedades objeto como por propiedades de tipo de datos. Al ser subclase de *Structure* hereda sus propiedades tanto objeto como de datos.

En la Figura 54 se representan las propiedades de la clase Non_Agent_Structure:

- **hasAction**: Indica que una NAPO tiene una acción. Tiene como subpropiedades:
  - **hasBehavior**: Indica que una NAPO tiene un comportamiento.
  - **hasFunction**: Indica que una NAPO tiene una función.
- **hasConnectionType**: Indica que una NAPO tiene un tipo de conexión topológica. Tiene dos subpropiedades:
  - **hasConnectionType_Structure1**: Indica que una estructura 1[jj] tiene un tipo de conexión.
  - **hasConnectionType_Structure2**: Indica que una estructura 2 tiene un tipo de conexión.
- **isStructureInferredBy**: Indica que una NAPO es inferida por una función.
- **isConnectedTo**: Indica una NAPO está conectada topológicamente a otra NAPO. Esta propiedad tiene una serie de subpropiedades:
  - **hasAngleWith:** Indica que una NAPO forma un ángulo con otra estructura.
  - **hasDistanceTo:** Indica que una NAPO tiene una distancia con otra estructura.
  - **isCamFollowerTo:** Indica que una NAPO es seguidor de la leva de otra estructura.
  - **isCoincidentTo:** Indica que una NAPO es coincidente a otra estructura.
  - **isConcentricTo:** Indica que una NAPO es concéntrica a otra estructura.
  - **isGearTo:** Indica que una NAPO es un engranaje de otra estructura.
  - **isParallelTo:** Indica que una NAPO es paralela a otra estructura.
  - **isPerpendicularTo:** Indica que una NAPO es perpendicular a otra estructura.
  - **isSymmetricTo:** Indica que una NAPO es simétrica a otra estructura.
  - **isTangentTo:** Indica que una NAPO es tangente a otra estructura.
  - **isUnknownTo**: Indica que una NAPO tiene una conexión topológica desconocida con otra estructura.

[jj] Tanto esta propiedad **hasConnectionType_Structure1** como **hasConnectionType_Structure2** y todas aquellas referidas a conexiones, por convenio se dirigen a dos estructuras genéricas (denominadas estructura1 y estructura2) que establecen la conexión.

- **hasEnvironment**: Indica que una NAPO tiene un ambiente donde se desarrolla.
- **hasMass**: Propiedad de datos que indica que una NAPO tiene una masa expresada en Kg.
- **hasMaterial**: Indica que una NAPO tiene un material del cual está compuesto.
- **isAssembledBy**: Indica que una NAPO está ensamblada a otra.
- **isPartOf**: Indica que una NAPO es parte de otra estructura.

Las características de las propiedades objeto quedan resumidas en la Tabla 16 y las referentes a tipo de datos en la Tabla 17.

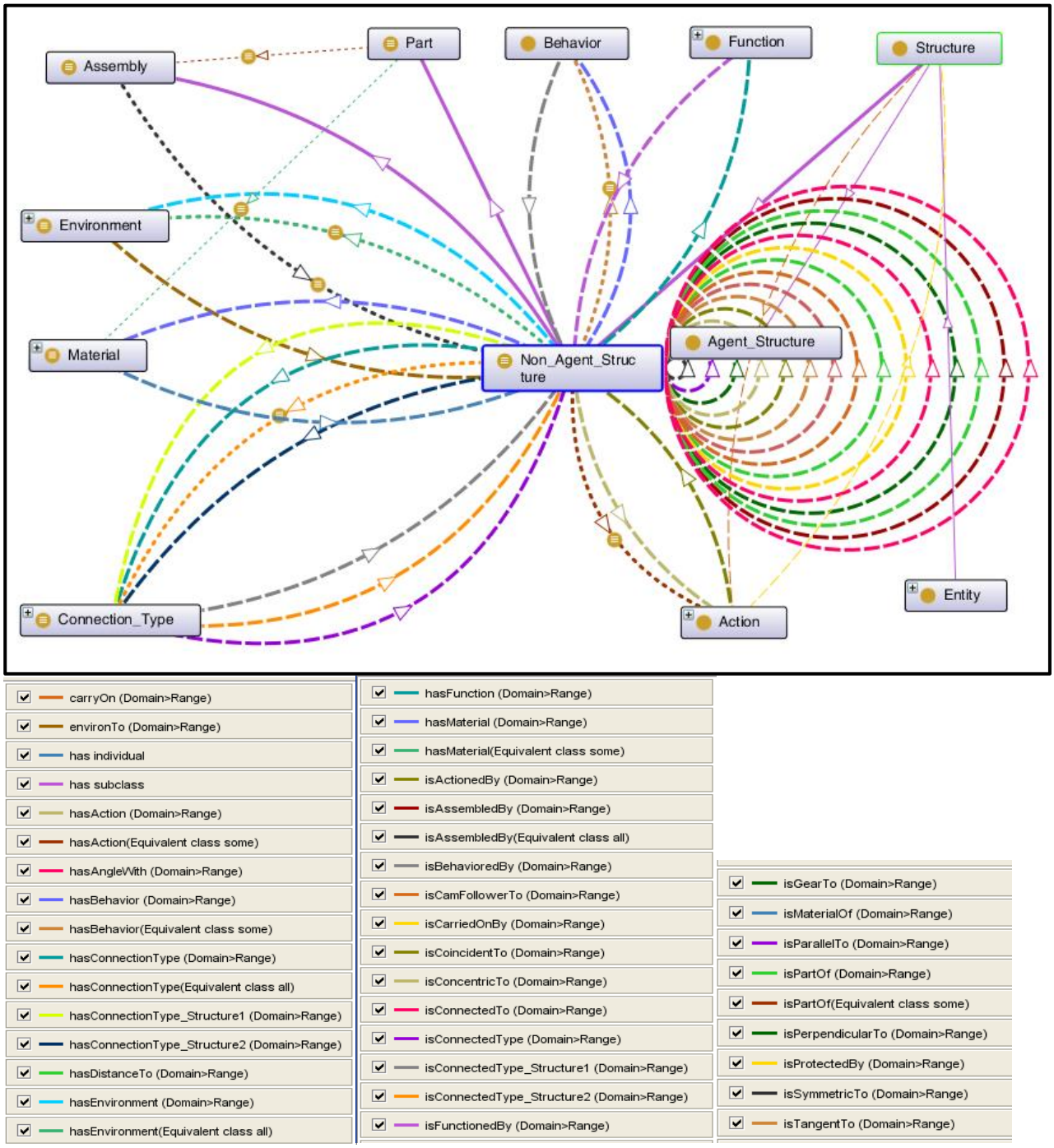


Figura 54: Relaciones de las propiedades de la clase Structure y sus subclases

Tabla 16: Descripción de las propiedades de objeto de la clase Non_Agent_Structure.

Propiedad de Objeto	Dominio	Rango	Rol	Inverso
hasAction	NAPO	Action	-	isActionedBy
hasBehavior	NAPO	Behavior	-	isBehavioredBy
hasFunction	NAPO	Behavior	-	isBehavioredBy
hasConnectionType	NAPO	Connection_Type		isConnectedType
_hasConnectionType_Structure1	NAPO	Connection_Type	Funcional inverso	isConnectedType_Structure1
_hasConnectionType_Structure2	NAPO	Connection_Type	Funcional inverso	isConnectedType_Structure2
isConnectedTo	NAPO	Structure	Simétrico	isConnectedTo
_hasAngleWith	NAPO	Structure	Simétrico	_hasAngleWith
_hasDistanceTo	NAPO	Structure	Simétrico	_hasDistanceTo
_isCamFollowerTo	NAPO	Structure	Simétrico	_isCamFollowerTo
_isCoincidentTo	NAPO	Structure	Simétrico	_isCoincidentTo
_isConcentricTo	NAPO	Structure	Simétrico	_isConcentricTo
_isGearTo	NAPO	Structure	Simétrico	_isGearTo
_isParallelTo	NAPO	Structure	Simétrico	_isParallelTo
_isPerpendicularTo	NAPO	Structure	Simétrico	_isPerpendicularTo
_isSymmetricTo	NAPO	Structure	Simétrico	_isSymmetricTo
_isTangentTo	NAPO	Structure	Simétrico	_isTangentTo
_isUnknownTo	NAPO	Structure	Simétrico	_isUnknownTo
has_Environment	NAPO	Environment	-	EnvironTo
hasMaterial	NAPO	Material	Funcional	isMaterialOf
isAssembledBy	NAPO	Structure	-	isPartOf
isPartOf	NAPO	Structure	-	isAssembledBy

Tabla 17: Descripción de las propiedades de tipo de datos de la clase Non_Agent_Structure.

Propiedad de tipo de datos	Dominio	Rango	Rol	Valores permitidos
hasMass	NAPO	float	Funcional	-

3.2.6 Capa de entorno de la acción

Se representa por la clase *Environment*.

Según la definición proporcionada en la ontología OntoFaBES, se considera una región física que, entre otras cosas, condiciona a la estructura con una serie de restricciones basadas en un rango entre dos temperaturas, presión y volumen específico. Dichas restricciones se crean condicionadas al diseño para el cual se aplique OntoFaBES.

La Figura 55 da una descripción de las restricciones aplicadas a la clase del entorno de la acción mediante pseudolenguaje OWL con la correspondiente paráfrasis.

A continuación se exponen los distintos apartados que conforman la descripción ontológica de la capa del entorno de la acción. Cabe comentar al respecto, que la clase Environment no tiene subclases.

OWL:
clase (Environment complete
restricción (EnvironTo someValuesFrom Structure)
restricción (hasTemperature minCardinality 2)
restricción (hasSpecificVolume minCardinality 1)
restricción (hasPressure minCardinality 1))
clase (Environment partial
OntoFaBES)
Paráfrasis:
Los entornos de la acción son regiones físicas en OntoFaBES que todas, entre otras cosas, ambientan a algunas estructuras a partir de un rango de dos temperaturas, un valor de presión y un valor de volumen específico.

Figura 55: Restricciones de la clase Environment.

3.2.6.1 Propiedades de Environment

En el caso de esta clase, hay una propiedad objeto que se concreta en:

- **EnvironTo**: Esta propiedad indica que un entorno de acción condiciona a una estructura.

Y tres propiedades de datos definidas:

- **hasTemperature**: Propiedad que indica que un entorno tiene un rango de temperatura expresada con dos valores, uno inferior y otro superior.
- **hasPressure**: Propiedad que indica que un entorno tiene una presión expresada en atmósferas.
- **hasSpecificVolume**: Propiedad que indica que un entorno tiene un volumen específico expresado en m^3/kg.

Las características de dicha propiedad queda resumida en la Tabla 18 y Tabla 19.

Tabla 18: Descripción de las propiedades de objeto de la clase Environment

Propiedad de Objeto	Dominio	Rango	Rol	Inverso
EnvironTo	Environment	Structure	-	Has_Environment

Tabla 19: Descripción de las propiedades de tipo de datos de la clase Environment.

Propiedad de tipo de datos	Dominio	Rango	Rol	Valores permitidos
hasTemperature	Environment	float	-	-
hasPressure	Environment	float	Funcional	-
hasSpecificVolume	Environment	float	Funcional	-

Las siguientes dos clases: *Material* y *Constraint*, tratan aspectos referidos a la estructura de un objeto, sin embargo, por consistencia con DOLCE (Figura 11) son clases disjuntas a la clase *Structure*. Se explica en más detalle los apartados siguientes.

3.2.7 Material

Se representa por la clase *Material*.

Es la materia de la que está hecha una parte de la estructura. Se considera una cantidad de materia pues es un *endurant* que no tiene unidad y es mereológicamente invariante[kk], en el sentido que cambia toda su entidad cuando cambia alguna de sus partes.

Según la definición proporcionada en la ontología OntoFaBES, se considera un *Amount of Matter* que, entre otras cosas, es material de alguna estructura, tiene algún valor de Densidad, algún valor de índice de Poisson, algún valor de módulo de elasticidad transversal, algún valor de conductividad térmica, algún valor de módulo de Young, algún valor de calor específico y algún valor de límite elástico.

Por tanto, por consistencia con DOLCE (Figura 11) que dentro de la metaontología, considera el *Amount of Matter, (*y *Material* como subclase de ella), una clase diferente a la de un APO, en OntoFaBES se ha mantenido esa coherencia manteniendo *Material* como una clase disjunta a la de *Estructura*.

En la Figura 57 se muestra la presentación de información donde se observa el árbol de clases indicado en la Figura 13 y las restricciones indicadas en la definición.

La Figura 58 da una descripción de las restricciones aplicadas a la clase *Material* mediante pseudolenguaje OWL con la correspondiente paráfrasis.

A continuación se exponen los distintos apartados que conforman la descripción ontológica de la clase Material. Cabe comentar al respecto, que la clase Material no tiene subclases.

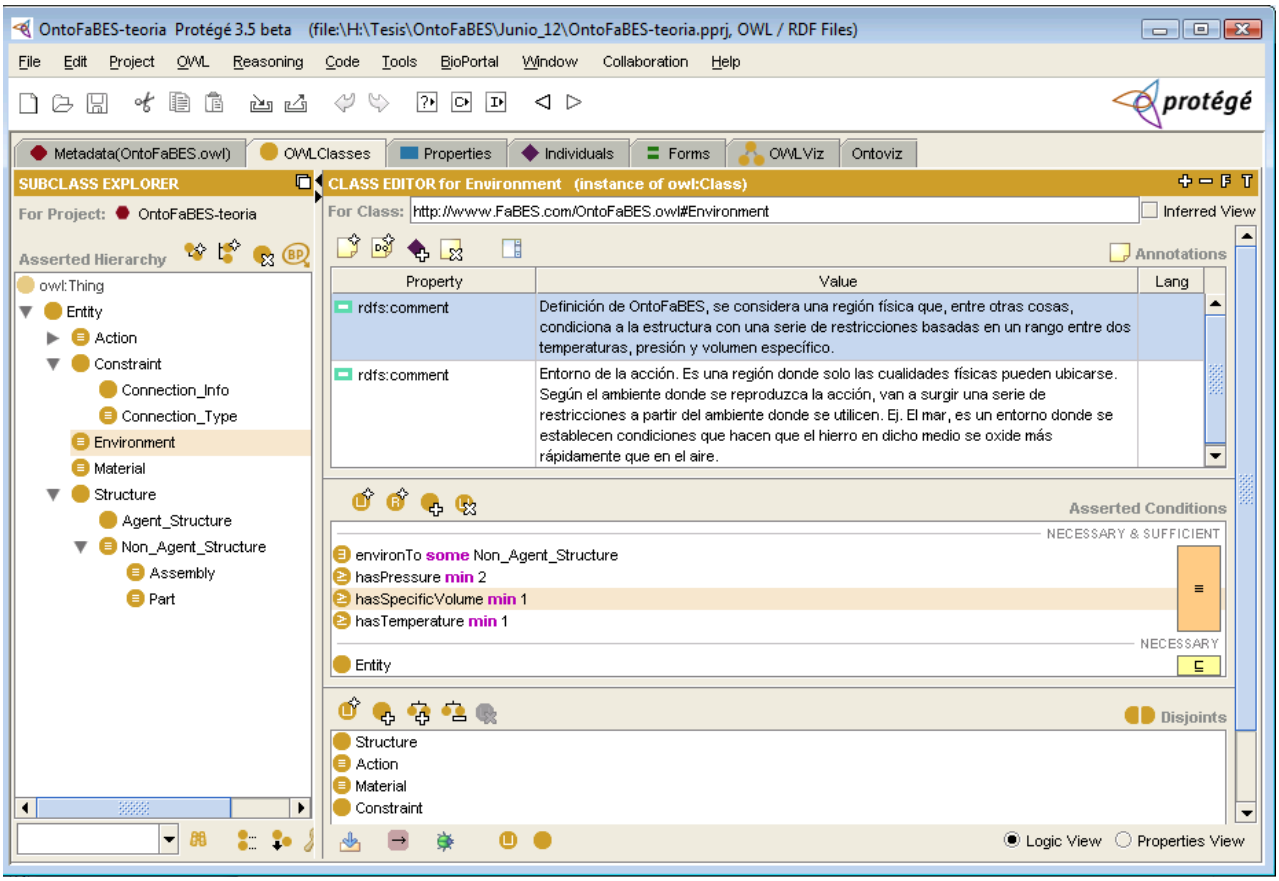


Figura 56: Captura de pantalla de OntoFaBES en Protégé de la clase Environment

[kk]Un elemento es mereológicamente invariante cuando todos los componentes de éste se mantienen constantes de manera indefinida.

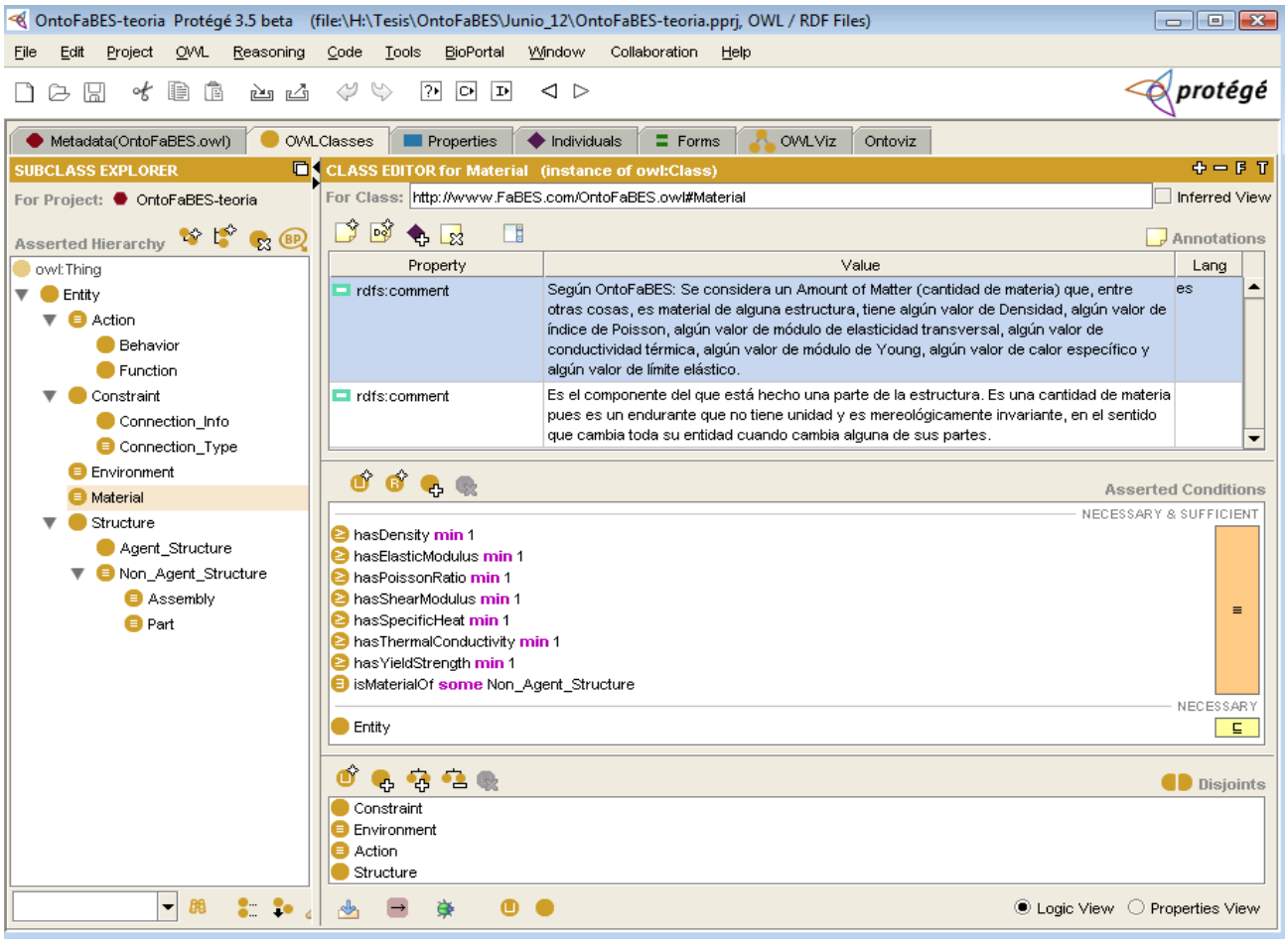


Figura 57: Captura de pantalla de OntoFaBES en Protégé de la clase Material

OWL:

clase (Material complete

restricción (hasDensity minCardinality 1)

restricción (hasPoissonRatio minCardinality 1)

restricción (hasShearModulus minCardinality 1)

restricción (hasSpecificHeat minCardinality 1)

restricción (hasThermalConductivity minCardinality 1)

restricción (hasYieldStrength minCardinality 1)

restricción (hasYoungModulus minCardinality 1)

restricción (isMaterialOf someValuesFrom Non_Agentive_Structure))

clase (Environment partial

OntoFaBES)

Paráfrasis:

Los materiales son cantidades de materia en OntoFaBES que todas tienen, entre otras cosas, tiene algún valor de Densidad, algún valor de Índice de Poisson, algún valor de módulo de elasticidad transversal, algún valor de conductividad térmica, algún valor de módulo de Young, algún valor de calor específico y algún valor de límite elástico y también son material para alguna NAPO.

Figura 58: Restricciones de la clase Material

3.2.7.1 Propiedades de la clase Material

En este caso, la clase *Material* se define tanto por propiedades objeto como por propiedades de tipo de datos

Propiedades Objeto de la clase Material

Las propiedades de objeto de la clase *Material* se concretan en:

- **isMaterialOf**: Esta propiedad indica que sólo un material forma parte de un NAPO. Se considera este aspecto para poder calcular las propiedades físicas de una estructura con la información disponible.

Las características de las propiedades de objeto quedan resumidas en la Tabla 20.

Tabla 20: Descripción de las propiedades de objeto de la clase Material.

Propiedad de Objeto	Dominio	Rango	Rol	Inverso
isMaterialOf	Material	NAPO		hasMaterial

Propiedades de tipo de datos de la clase Material

Las propiedades de tipo de datos de la clase Material se establecen para poder definir las propiedades físico-químicas de los materiales de una estructura.

- **hasDensity**: Indica un material tiene un valor de densidad. Unidades: kg/m^3.
- **hasPoissonRatio**: Indica un material tiene un valor del índice de Poisson.
- **hasShearModulus**: Indica un material tiene un valor de elasticidad transversal o módulo cortante. Unidades: GPa.
- **hasSpecificHeat**: Indica un material tiene un valor de calor específico. Unidades: J/g ºC.
- **hasThermalConductivity**: Indica un material tiene un valor de conductividad térmica. Unidades: W/m K.
- **hasYieldStrength**: Indica un material tiene un valor de límite elástico. Unidades: MPa.
- **hasYoungModulus**: Indica un material tiene un valor de módulo de Young. Unidades: GPa.

Las características de las propiedades de tipo de datos quedan resumidas en la Tabla 21.

Tabla 21: Descripción de las propiedades de tipo de dato de la clase Material.

Propiedad de tipo de datos	Dominio	Rango	Rol	Valores permitidos
hasDensity	Material	float	Funcional	-
hasPoissonRatio	Material	float	Funcional	-
hasShearModulus	Material	float	Funcional	-
hasSpecificHeat	Material	float	Funcional	-
hasThermalConductivity	Material	float	Funcional	-
hasYieldStrength	Material	float	Funcional	-
hasYoungModulus	Material	float	Funcional	-

3.2.8 Constraint

Se representa por la clase *Constraint*.

Esta clase se ha creado con el fin de poder enlazar los formularios ICARE de MOKA con OntoFaBES. Concretamente al apartado referido a la obtención de información de los programas CAD según se explica en el aptdo. 4.1

Según la definición proporcionada en la ontología OntoFaBES, se considera una característica [II] representando de las conexiones topológicas entre NAPOs para describir de forma detallada su uso. Por tanto, por consistencia con DOLCE (Figura 11), una clase que es una característica (*Feature*), es disjunta de los NAPOs y APOs, es decir, de la clase de *Physical Object* que los agrupa.

La Figura 59 da una descripción de las restricciones aplicadas a la clase de mediante pseudolenguaje OWL con la correspondiente paráfrasis.

A continuación se exponen los distintos apartados que conforman la descripción ontológica de la clase Constraint. Cabe comentar al respecto, que la clase Constraint tiene dos subclases: *Connection_Info* y *Connection_Type*, la primera referida a la información de la conexión topológica entre dos estructuras y la segunda referida al tipo de conexión que se establece.

3.2.8.1 Propiedades de la clase Constraint

En este caso, la clase *Constraint* se define tanto por propiedades objeto como por propiedades de tipo de datos. Así, se van a indicar conjuntamente las propiedades de sus subclases *Connection_Info* y *Connection_Type*.

OWL:
clase (Constraint complete
clase (Constraint partial
OntoFaBES)
Paráfrasis:
Las restricciones son características representando las conexiones topológicas entre NAPOS para describir de forma detallada su uso.

Figura 59: Restricciones de la clase Constraint

Propiedades Objeto de la clase Constraint

Las propiedades de objeto de la clase *Constraint* se concretan en:

- **isConnectedInfoWith**: Esta propiedad indica que la información de una restricción está relacionada con un tipo de conexión.
- **hasConnectionInfoWith**: Esta propiedad indica que un tipo de conexión está relacionada con la información de una restricción.
- **isConnectedType**: Indica que un tipo de conexión se dispone por un NAPO. Tiene dos subpropiedades:
    - **hasConnectionType_Structure1**: Indica que un tipo de conexión es dispuesto por una estructura1.
    - **hasConnectionType_Structure2**: Indica que un tipo de conexión es dispuesto por una estructura2.

Las características de las propiedades de objeto quedan resumidas en la Tabla 22.

Tabla 22: Descripción de las propiedades de objeto de la clase Constraint.

Propiedad de Objeto	Dominio	Rango	Rol	Inverso
isConnectedInfoWith	Connection_Info	Connection_Type		hasConnectionInfoWith
hasConnectionInfoWith	Connection_Type	Connection_Info		isConnectedInfoWith
isConnectedType	Connection_Type	NAPO		hasConnectionType
_isConnectedType_Structure1	Connection_Type	NAPO	Funcional	hasConnectionType_Structure1
_isConnectedType_Structure2	Connection_Type	NAPO	Funcional	hasConnectionType_Structure2

[II] Una característica se representa en DOLCE como *Feature*. Se puede observar este aspecto en la Figura 10.

Propiedades de tipo de datos de la clase Constraint

Las propiedades de tipo de datos de la clase Constraint se establecen para poder regular las características de la conexión entre estructuras.

- **hasConnectionInfoCharacteristics**: Indica una restricción tiene unas características de información sobre la conexión. Tiene las siguientes subpropiedades:
    - **hasLocationConnection**: Indica que tiene la localización de la conexión entre dos elementos. Tiene las siguientes subpropiedades:
        - **hasLocationConnectionCoords**: Indica que tiene las coordinadas de la localización de la conexión. Tiene las siguientes subpropiedades:
            - **hasLocationConnectionCoords_Structure1**: Indica que tiene las coordinadas de la localización de la conexión de la estructura 1. Tiene las siguientes subpropiedades:
                - **hasLocationConnectionCoords_Ext_1**: Indica que tiene las coordinadas externas sobre la localización de la conexión de la estructura 1.
                - **hasLocationConnectionCoords_Int_1**: Indica que tiene las coordinadas internas sobre la localización de la conexión de la estructura 1.
            - **hasLocationConnectionCoords_Structure2**: Indica que tiene las coordinadas de la localización de la conexión de la estructura 2. Tiene las siguientes subpropiedades:
                - **hasLocationConnectionCoords_Ext_2**: Indica que tiene las coordinadas externas sobre la localización de la conexión de la estructura 2.
                - **hasLocationConnectionCoords_Int_2**: Indica que tiene las coordinadas internas sobre la localización de la conexión de la estructura 2.
        - **hasLocationConnectionRadius**: Indica que tiene el radio de la localización de la conexión. Tiene las siguientes subpropiedades:
            - **hasLocationConnectionRadius_Structure1**: Indica que tiene el radio de la localización de la conexión de la estructura 1. Tiene las siguientes subpropiedades:
                - **hasLocationConnectionRadius_Ext_1**: Indica que tiene el radio externo sobre la localización de la conexión de la estructura 1.
                - **hasLocationConnectionRadius_Int_1**: Indica que tiene el radio interno sobre la localización de la conexión de la estructura 1.
            - **hasLocationConnectionRadius_Structure2**: Indica que tiene el radio de la localización de la conexión de la estructura 2. Tiene las siguientes subpropiedades:
                - **hasLocationConnectionRadius_Ext_2**: Indica que tiene el radio sobre la localización de la conexión de la estructura 2.
                - **hasLocationConnectionRadius_Int_2**: Indica que tiene el radio sobre la localización de la conexión de la estructura 2.
- **hasConnectionTypeCharacteristics**: Indica las características de una conexión de una pieza con otra. Tiene las siguientes subpropiedades:
    - **CanBeFlipped**: Indica si el tipo de conexión puede invertirse.
    - **hasAligmentType**: Indica el tipo de alineamiento que tiene una conexión. Los valores permitidos son: alineado, no-alineado y cercano.
    - **hasDimensionValue**: Indica que el tipo de conexión tiene un valor.
    - **hasMaximumVariation**: Indica la variación máxima que tiene un tipo de conexión.
    - **hasMinimumVariation**: Indica la variación mínima que tiene un tipo de conexión.

- **hasType**: Indica el tipo de conexión existente entre dos piezas. Los valores que puede tener son los siguientes: coincidente, concéntrico, perpendicular, paralelo, tangente, distante, ángulo, bloqueado, desconocido y simétrico.

Las características de las propiedades de tipo de datos quedan resumidas en la Tabla 23.

Tabla 23: Descripción de las propiedades de tipo de dato de la clase Constraint.

Propiedad de tipo de datos	Dominio	Rango	Rol	Valores permitidos
hasConnectionInfoCharacteristics	Connection_Info	any	-	-
_hasLocationConnection	Connection_Info	any	-	-
__hasLocationConnectionCoords	Connection_Info	string	-	-
___hasLocationConnectionCoords_Structure1	Connection_Info	string	-	-
____hasLocationConnectionCoords_Ext_1	Connection_Info	string	Funcional	-
____hasLocationConnectionCoords_Int_1	Connection_Info	string	Funcional	-
___hasLocationConnectionCoords_Structure2	Connection_Info	string	-	-
____hasLocationConnectionCoords_Ext_2	Connection_Info	string	Funcional	-
____hasLocationConnectionCoords_Int_2	Connection_Info	string	Funcional	-
__hasLocationConnectionRadius	Connection_Info	string	-	-
___hasLocationConnectionRadius_Structure1	Connection_Info	string	-	-
____hasLocationConnectionRadius_Ext_1	Connection_Info	string	Funcional	-
____hasLocationConnectionRadius_Int_1	Connection_Info	string	Funcional	-
___hasLocationConnectionRadius_Structure2	Connection_Info	string	-	-
____hasLocationConnectionRadius_Ext_2	Connection_Info	string	Funcional	-
____hasLocationConnectionRadius_Int_2	Connection_Info	string	Funcional	-
hasConnectionTypeCharacteristics	Connection_Type	Any	-	-
_CanBeFlipped	Connection_Type	boolean	Funcional	-
_hasAligmentType	Connection_Type	string	Funcional	*Aligned*, *Anti-Aligned* y *Closest*
_hasDimensionValue	Connection_Type	float	Funcional	-
_hasMaximumVariation	Connection_Type	float	Funcional	-
_hasMinimumVariation	Connection_Type	float	Funcional	-

Propiedad de tipo de datos	Dominio	Rango	Rol	Valores permitidos
hasType	Connection_Type	string	Funcional	*Coincident, Concentric, Perpendicular, Parallel, Tangent, Distance, Gear, Angle, Unknown y Symmetric*

3.3 Inferencia de conocimiento

Esta sección introduce el formalismo de OntoFaBES para la automatización del proceso de diseño conceptual haciendo uso de marco FBS. El formalismo permite la inferencia de conocimiento y como se ha indicado en el apartado 2.4.8, se utiliza para ello el lenguaje SQWRL, el razonador Racer Pro y la herramienta Jess.

Una de las características formales del modelo FBS es su posible capacidad para la inferencia de conocimiento desde el apartado de función hasta la fase de estructura, tal como han indicado diversos autores con anterioridad (Deng, 1999; Garbacz, 2005; Vermaas, 2010). Este hecho se puede observar a partir de la Figura 13 donde se plantea las relaciones entre las diferentes capas del modelo aportado FaBES.

En el pasado, el cálculo de posibilidades se ha realizado tanto utilizando sistemas KBE, como de sistemas expertos, con variados resultados en este aspecto (Skarka, 2007). La siguiente evolución en este apartado surge a partir del desarrollo de la ingeniería ontológica, la cual como ya se ha mostrado permite la clasificación explícita de la información, pero brinda también la posibilidad de inferir el conocimiento. Para conseguir un resultado óptimo se utiliza la capacidad de razonamiento del lenguaje OWL unido a las características del lenguaje SQWRL.

3.3.1 Funcionamiento

Para la aplicación de las reglas SQWRL se utiliza el programa Protégé en la pestaña referida SWRL Rules (Figura 60).

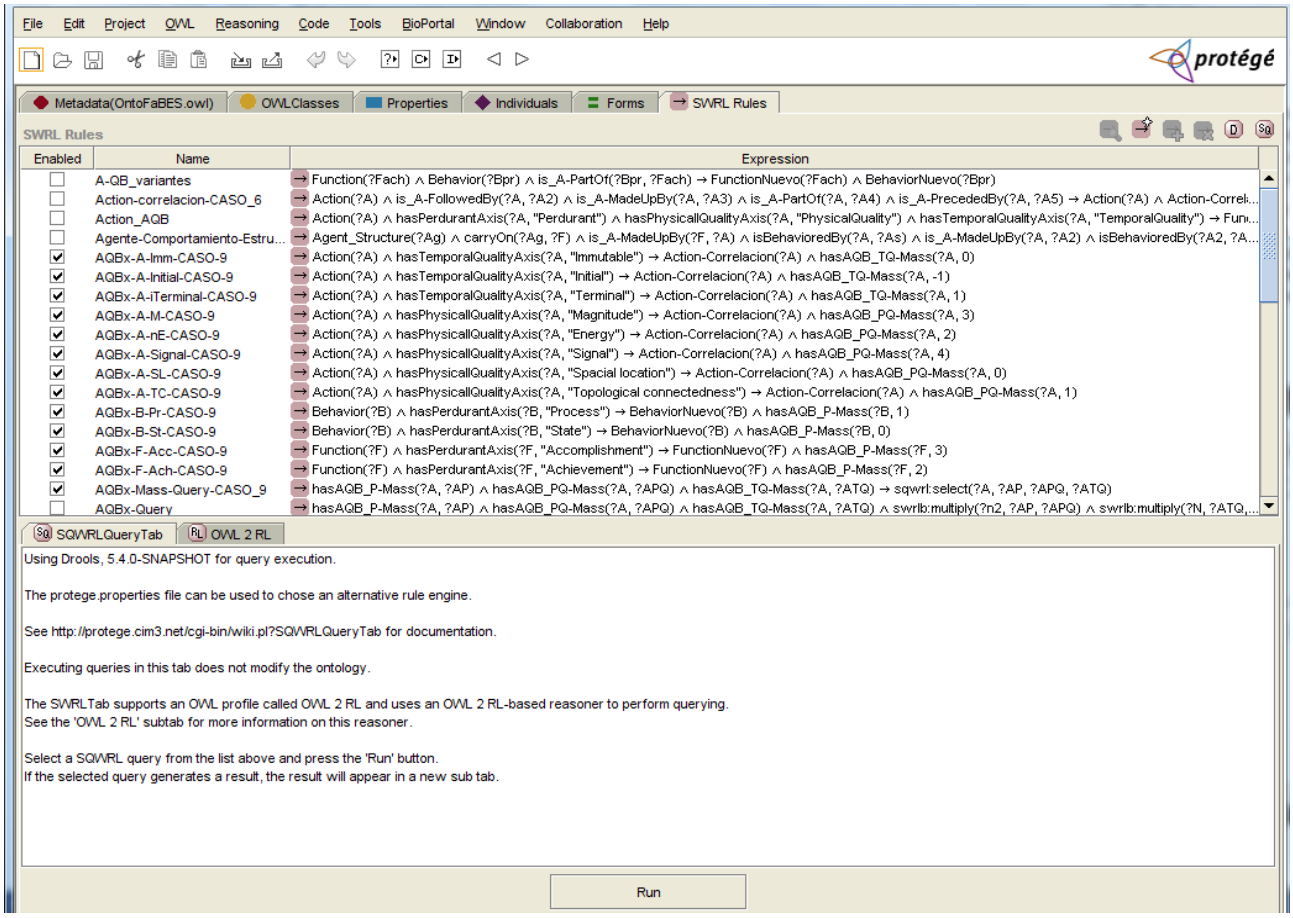


Figura 60: Pestaña de SWRL Rules en Protégé

Para que el ingeniero de conocimiento aplique una regla en concreto sobre la ontología, sólo debe señalarla como permitida y posteriormente ejecutar la orden. Después de un tiempo que suele ser menor del minuto, se obtienen los resultados de la aplicación de las reglas sobre la ontología. Una vez que se disponen de esos datos, el ingeniero de conocimiento puede elegir si incluirlos como nuevas propiedades o individuos dentro de la ontología o no.

3.3.2 Reglas SQWRL

Una vez que el contenido es almacenado en la ontología, OntoFaBES puede capturar el conocimiento obtenido de búsquedas de información para poder relacionar directamente las funciones con la estructura del diseño que cumpliría los requerimientos establecidos.

Para ello utiliza el lenguaje SQWRL, cuyas reglas pueden ser razonadas para acomodar las potenciales búsquedas semánticas y las peticiones de información para la fase de diseño conceptual de un diseño. Con el fin de generar las reglas de OntoFaBES se utilizó el editor SWRL de Protégé.

Al incluir las reglas SQWRL en el árbol previamente indicado en la Figura 13 se incluyen las siguientes clases:

- **Lenguaje SQWRL**: temporal: *Entity; swrla: Entity; swrlxml: Entity*.

La organización a la hora de describir las series de reglas consiste en el mismo sistema utilizado para definir las clases en lenguaje OWL: Primero, se indica la definición de la relación; a continuación se indican las restricciones implicadas y por último se muestra la regla SQWRL.

Relación 1: Función-NAPO

La Regla 1 muestra la inferencia de conocimiento establecida entre una función y un NAPO determinado y que se establece a partir de la propiedad: **hasInferredStructure.**

Relación 2: Estructura-Función

La Regla 2 muestra la inferencia de conocimiento establecida entre una función y una estructura determinada se establece a partir de la propiedad: **isStructureInferred**.

Relación 3: Protección

Esta relación (Regla 3) determina si un elemento de la estructura debe protegerse por otro, para poder cumplir su comportamiento. Para ello se evalúa el material del cual está compuesto. Si un elemento tiene unas propiedades mecánicas inferiores a las necesarias necesita una protección con respecto a otra para evitar su fragilidad.

Está relación es dependiente de la Relación 1.

Regla 1: Función-Estructura

Definición de la relación hasInferredStructure
Una función F_i, y un NAPO S_j tienen una relación si y sólo si una función tiene una relación isCarriedOnBy con un APO A_t, si un comportamiento B_k tiene una relación *is_A-MadeUpBy* con F_i y a la vez tiene otra relación *isBehavioredBy* con S_k
Restricciones implicadas
R1-1: Toda función debe tener otra acción la cual necesita para poder ejecutarse. R1-2: Toda función debe tener un comportamiento al cual está ligado. R1-3: Toda función debe tener un APO que lleve a cabo dicha función. R1-4: Toda estructura debe tener un comportamiento al cual está ligada.
Regla SQWRL
Function(?F) ∧ isCarriedOnBy(?F, ?A) ∧ is_A-MadeUpBy(?F, ?B) ∧ isBehavioredBy(?B, ?S) → hasInferredStructure(?F, ?S)

Relación 4: Tipo de Conexión

Esta Regla 4 determina las diferentes conexiones existentes entre piezas a partir de la información referente a la colocación de las piezas en la estructura y el tipo de conexión existente entre ellas.

Regla 2: Estructura-Función

Definición de la relación isStructureInferred
Una estructura $NAPO_i$, y una función F_j tienen una relación si y sólo si un comportamiento S_i tiene una relación *hasBehavior* con B_k y B_k tiene otra relación *is_A-PartOf* con F_j
Restricciones implicadas
R2-1: Toda estructura debe tener un comportamiento al cual está ligado. R2-2: Todo comportamiento debe tener una función a la cual está ligado.
Regla SQWRL
Structure(?Si) ∧ hasBehavior(?Si, ?Bk) ∧is_A-PartOf(?Bk, ?Fj) → isStructureInferredBy(?Si, ?Fj)

Regla 3: Protección

Definición de la relación Protección
Una NAPO S_i y una NAPO S_j tienen una relación de protección sí y sólo sí S_i tiene una relación *hasInferredStructure* con F_k, F_k tiene una relación *hasMaterial* con un APO A_l, que tanto A_l como S_i tienen la relación *hasMaterial* con un material M_1 y M_2, respectivamente. Dichos materiales tienen una relación *hasElasticModulus* Y_1 y Y_2 respectivamente, y se determina que Y_1 es mayor que Y_2.
Restricciones implicadas
R3-1: Toda función debe ser realizada por un agente. R3-2: Toda estructura como APO deben tener un material del cual están compuestos. R3-3: Todo material tiene un módulo de elasticidad.
Regla SQWRL
NAPO(?S_i) ∧ hasInferredStructure(?S_i, ?F_k) ∧ isActionedBy(?F_k, ?A_l) ∧ hasMaterial(?A_l, ?M_1) ∧ hasMaterial(?S_i, ?M_2) ∧ hasElasticModulus(?M_1, ?Y_1) ∧ hasElasticModulus(?M_2, ?Y_2) ∧ swrlb:greaterThan(?Y_1, ?Y_2) → Entity(?S_i)

Regla 4: Tipo de conexión

Definición de la relación Tipo de Conexión
Una NAPO S_j y una NAPO S_k tienen una conexión y definida por un tipo, sí y sólo sí S_j y S_k tienen una relación *isConnectedType_Structure1* e *isConnectedType_Structure2* con un tipo de conexión C_i, respectivamente. Dichos NAPO tienen una relación *hasType* definida por un "*type of connection*" y se determina entonces que S_j y S_k tienen una relación "*hasTypeOfConnection*".
Restricciones implicadas
R4-1: Todo NAPO debe tener una conexión. R4-2: Toda conexión debe estar clasificada por un tipo de conexión. R4-3: Toda NAPO debe estar conectada a otra por un tipo de conexión.
Regla SQWRL
Connection_Type (?C_i)∧isConnectedType_Structure1 (?C_i, ?S_j)∧isConnectedType_Structure2 (?C_i, ?S_k) ∧ hasType(?C_i, "*type of connection*") → *hasTypeOfConnection* (?S_j, ?S_k)

Relación 5: Prevención de la corrosión atmosférica

Esta relación conformada por la Regla 5, Regla 6 y Regla 7, determina qué piezas en una estructura tienen riesgo de sufrir corrosión atmosférica.

REGLA 5

La Regla 5 determina cuál es un posible entorno de riesgo para la corrosión atmosférica (Vázquez, 1991). Para ello se establecen las siguientes simplificaciones:

- Presión atmosférica.
- La corrosión atmosférica se establece a partir de altos niveles de humedad relativa y formación de rocío pues está en función del incremento de la Temperatura y de la humedad.

  Por tanto, cuanto más seco sea el ambiente, mayor debe ser el descenso de temperatura para que se produzca la condensación. Igualmente la fracción de tiempo con elevados valores de humedad relativa es un indicador de mayor probabilidad de corrosión atmosférica.

- Humedad relativa: Los valores mayores de 70% se consideran propicios a la corrosividad donde la lluvia, niebla o rocío se computa como humedad relativa alta.
- Temperatura: Se consideran temperaturas de operación por encima de 0 ºC. Su aumento produce un aumento de la velocidad de corrosión.
- Contaminantes: Se omite la presencia de contaminantes como dióxido de azufre y otros considerándose sólo la influencia del cloruro sódico considerándose que la atmósfera marina maximiza la corrosión.

REGLA 6

La Regla 6 muestra que un material si es férrico, que tiene una conductividad térmica entre 20 y 100 W/(K·m), se puede corroer.

REGLA 7

La Regla 7 se basa en las dos anteriores para poder ejecutarse e indica que una estructura se puede corroer si tiene un entorno corrosivo y está conformada por un material que puede corroerse.

Regla 5: Entorno favorable para la corrosión atmosférica.

Definición de la relación Tipo de Conexión
Un entorno E_i puede ser corrosivo sí y sólo sí E_i tiene una relación *hasPressure* con hP_j mayor que 0,9 atm., una relación *hasHumidity* con hH_k mayor de 70%, y una relación *hasTemperature* con hT_l menor de 100 ºC.
Restricciones implicadas
R5-1: Todo entorno corrosivo debe tener una presión atmosférica mayor que 0,9 atm. R5-2: Toda entorno corrosivo debe tener una humedad relativa mayor del 70%. R5-3: Todo entorno corrosivo debe tener una temperatura menor de 100 ºC. R5-4: Todo entorno tiene una presión, una temperatura y una humedad relativa.
Regla SQWRL
Environment(?E_i) ∧ hasPressure(?E_i, ?hP_j) ∧ swrlb:greaterThan(?hP_j, 0.9) ∧ hasHumidity(?E_i, ?hH_k) ∧ swrlb:greaterThan(?hH_k, 70.0) ∧ hasTemperature(?E_i, ?hT_l) ∧ swrlb:lessThan(?hT_l, 100.0) → EntornoCorrosivo(?E_i)

Relación 6: Determinación del tipo de acción.

Esta Regla 8 permite inferir el tipo de acción de una estructura según el modelo A-QB, determinando si es una función o un comportamiento.

Regla 6: Material que se puede corroer.

Definición de la relación Material corrosivo
Un material M_i es férrico, y por tanto puede corroerse, sí y sólo sí tiene una relación *hasThermalConductivity* fijando que su valor hTC_j es mayor que 20 W/ (K·m) y menor que 100 W/ (K·m).
Restricciones implicadas
R6-1: Todo material tiene una conductividad térmica. R6-2: Todo metal es férrico si tiene una conductividad térmica entre 20 y 100 W/(K·m)
Regla SQWRL
Material(?M_i) ∧ hasThermalConductivity(?M_i, ?hTC_j) ∧ swrlb:greaterThan(?hTC_j, 20) ∧ swrlb:lessThan(?hTC_j, 100) → MaterialCorrosivo(?M_i)

Regla 7: Piezas de una estructura que se pueden corroer.

Definición de la relación Estructura corrosiva.
Una estructura S_k puede corroerse sí y sólo sí un entorno corrosivo E_i tiene una relación *environTo* con un ensamblaje As_l, As_l tiene una relación *isAssembledBy* con S_k y S_k tiene una relación *hasMaterial* con un material corrosivo M_j.
Restricciones implicadas
R7-1: Todo entorno debe tener una estructura con la que interacciona. R7-2: Toda estructura como APO deben tener un material del cual están compuestos.
Regla SQWRL
EntornoCorrosivo(?E_i) ∧ MaterialCorrosivo(?M_j) ∧ environTo(?E_i, ?As_l) isAssembledBy(?As_l, ?S_k) ∧ hasMaterial(?S_k, ?M_j) → EstructuraCorrosiva(?S_k)

Regla 8: Determinación del tipo de acción según el modelo A-QB.

Definición de la relación Tipo de Acción
Una acción A_i se determina como una función o un comportamiento sí y sólo sí tiene una relación *hasPerdurantAxis* con "*un tipo de perdurant*", una relación *hasPhysicallQualityAxis* con "*un tipo de cualidad física*", y tiene una relación *hasTemporalQualityAxis* con *"un tipo de cualidad temporal"*.
Restricciones implicadas
R1-1; R1-2; R1-3 R8-1: Toda acción debe tener un *perdurant*, una cualidad física y otra temporal que le define. R8-2: Toda acción debe ser una función o un comportamiento. R8-3: Toda función debe tener un *perdurant logro* o *cumplimiento (achievement/accomplishment)*. R8-4: Todo comportamiento debe tener un *perdurant estado* o *proceso (state/process)*.
Regla SQWRL
Action(?A_i) ∧ has*Perdurant*Axis(?A_i, "achievement/accomplishment") ∧ hasPhysicallQualityAxis(?A_i, "PhysicalQuality") ∧hasTemporalQualityAxis(?A_i, "TemporalQuality") → Function(?A_i) Action(?A_i) ∧ has*Perdurant*Axis(?A_i, "state/process") ∧ hasPhysicallQualityAxis(?A_i, "PhysicalQuality") ∧ hasTemporalQualityAxis(?A_i, "TemporalQuality") → Behavior(?A_i)

Relación 7: Correlación entre acciones.

Esta Regla 9 permite conocer las relaciones existentes entre las acciones de una estructura. Esta regla se puede enlazar con las reglas 1 y 8.

Regla 9: Determinación de la correlación entre funciones y comportamientos

Definición de la relación Correlación entre funciones y comportamientos
En una estructura S_i, una acción A_j está relacionada con otra acción A_n sí y sólo sí una estructura tiene una relación *hasAction* con A_j, si la acción A_j tiene una relación *is_A-FollowedBy* con A_n, y tiene una relación *is_A-MadeUpBy* con A_n, y tiene una relación *is_A-PartOf* con A_n, y tiene una relación *is_A-PrecededBy* con A_n.
Restricciones implicadas
R1-1; R1-2; R1-3, R8-1, R8-2,R8-3, R8-4 R9-1: Toda acción debe tener una acción a la cual está ligada.
Regla SQWRL
$Structure(?S_i) \wedge hasAction(?S_i, ?A_j) \wedge is\_A\text{-}FollowedBy(?A_j, ?A_n) \wedge is\_A\text{-}MadeUpBy(?A, ?A_n) \wedge is\_A\text{-}PartOf(?A, ?A_n) \wedge is\_A\text{-}PrecededBy(?A, ?A_n) \rightarrow Action(?A_i) \wedge ActionCorrelacion(?A_n)$

Relación 8: Materiales de un ensamblaje.

Esta Regla 10 permite conocer los materiales que conforman un ensamblaje.

Regla 10: Materiales disponibles en un ensamblaje.

Definición de la relación Materiales de un ensamblaje.
Un Material M_j tiene una relación *hasMaterial* con un ensamblaje AS_k y $AS2_l$, sí y sólo sí, una parte P_i tiene una relación *hasMaterial* con M_j y tiene una relación *isPartOf* con AS_k y AS_k tiene la relación *isPartOf* con $AS2_l$.
Restricciones implicadas
R7-2
Regla SQWRL
$Part(?P_i) \wedge hasMaterial(?P_i, ?M_j) \wedge isPartOf(?P_i, ?AS_k) \wedge isPartOf(?AS_k, ?AS2_l) \rightarrow hasMaterial(?AS_k, ?M_j) \wedge hasMaterial(?AS2_l, ?M_j)$

Relación 9: Porcentaje de un material en un ensamblaje.

Esta Regla 11 permite conocer el porcentaje de un material concreto en un ensamblaje.

Regla 11: Porcentaje de materiales en un ensamblaje.

Definición de la relación Porcentaje de un material en un ensamblaje.
El Porcentaje de un material en un ensamblaje se establece sí y sólo sí un ensamblaje A_i tiene una relación *isAssembledBy* con una parte P_j, P_j y A_i tienen una relación *hasMass* con un masa mA_k y mP_l, respectivamente, se divide mP_l entre mA_k y se multiplica por 100.
Restricciones implicadas
R7-2
Regla SQWRL
$Assembly(?A_j) \wedge isAssembledBy(?A_i, ?p) \wedge hasMass(?p, ?mA_k) \wedge hasMass(?A_i, ?) \wedge hasMaterial(?p,?M) \wedge swrlb:divide(?pc,?mP_l,?mA_k) \wedge swrlb:multiply(?porcentaje,?pc,100) \rightarrow sqwrl:select(?A_i,?p,?M,?porcentaje)$

Relación 10: Acciones según el A-QB.

Esta Regla 12 permite distribuir las acciones según el modelo A-QB.

Regla 12: Acciones según el A-QB.

Definición de la relación Acciones según el A-QB.
Un Acción A_i tiene una relación *hasAQB_Mass* con un peso H_j sí y sólo sí tiene una relación *hasAQB_P-Mass* con "*un tipo de perdurant*", una relación *hasAQB_PQ-Mass* con "*un tipo de cualidad física*", y tiene una relación *hasAQB_TQ-Mass* con *"un tipo de cualidad temporal"*.
Restricciones implicadas
R1-1; R1-2; R1-3, R8-1, R8-2, R8-3, R8-4. R12-1: Toda acción tiene unos pesos asociados respecto a *Perdurant*, Cualidad física o temporal. R12-2: Todo *perdurant* tiene el siguiente peso: State: 1, Process: 2, Achievement:3, Accomplishment: 4. R12-3: Toda Cualidad física tiene el siguiente peso: Spatial Location:0, Topological Connectedness:1, Energy:2, Magnitude:3, Signal:4. R12-4: Toda Cualidad temporal tiene el siguiente peso: Initial SoA:-1, Immutable SoA: 0, Terminal SoA:1.
Regla SQWRL
Action ($?A_i$) ∧ hasAQB_P-Mass($?A_i$, ?AP) ∧ hasAQB_PQ-Mass($?A_i$, ?APQ) ∧ hasAQB_TQ-Mass($?A_i$, ?ATQ) ∧ → hasAQB_Mass($?A_i$, $?H_j$)

3.3.3 Aplicación práctica

Para la aplicación práctica de la inferencia de conocimiento mediante OntoFaBES se va a utilizar el ejemplo del portaminas mostrado en el apartado 3.1.9.4. Para ello, el caso de que se va a tratar va a ser la inferencia de una estructura a partir de una función. En este ejemplo, la función principal es experimentar la escritura con minas (Figura 29) y las secundarias son: escribir, impulsar minas, transportar minas, proteger minas y proporcionar retractilidad.

Para aplicar la inferencia de conocimiento se va a aplicar la Regla 1: Función-Estructura. Los resultados se muestran en la Tabla 24. Si se comparan los resultados obtenidos en dicha tabla con la Figura 29, se podrá observar que la inferencia que para realizar la función de escribir se determinan dos elementos de la estructura: la mina de lápiz y la goma de borrar. A primera vista, se podría considerar como un error de la ontología, sin embargo, si uno se remite a la definición de escribir, una goma, sobre una zona totalmente rallada, las marcas que realice pueden interpretarse como signos, con lo cual puede interpretarse como rallar.

Tabla 24: Inferencia de una estructura a partir de una función

Función	Estructura
Experimentar la escritura con minas	tip
	rubber_cap
	tube
	cone
	mechanical_teeth
	mechanism
	barrel
	sleeve
	soporte
	tip
	spring
	rubber
	pencil_lead
Escribir	rubber
	pencil_lead

Función	Estructura
Impulsar minas	cone
	spring
	sleeve
	soporte
	tip
	spring
Transportar minas	mechanism
Proteger minas	tip
	rubber_cap
	tube
Proporcionar retractilidad	mechanical_teeth
	mechanism
	barrel
	spring

3.4 Discusión y conclusiones

FaBES (*Function-action-Behavior Environment Structure*) se estructura como una posible solución ante las necesidades en el ámbito del diseño teniendo presente la rigurosidad necesaria para generar una base estable para OntoFaBES ante las deficiencias encontradas en modelos como el NIST (Garbacz, 2006) o *FBS Ontology* (Galle, 2009). Para ello, establece una redefinición de los diferentes modelos existentes para poder ser adaptados a la estructura de una ontología.

Ante la posibilidad de considerar el diseño como un proceso de comunicación (Crilly, 2008) para poder traducir los requerimientos en un diseño establecidos en lenguaje natural para su posterior formalización en OntoFaBES, se plantea una analogía (Figura 14). Dicha analogía establece la relación entre funciones y comportamientos como acciones, lo que permite construir posteriormente el A-QB. En dichas acciones se establece por tanto un nuevo tipo de relación para OntoFaBES que es la pertenencia y consecución de éstas (Figura 16).

Al generar la citada analogía se resuelve uno de los interrogantes que generaba el modelo B-Cube del cual el modelo A-QB adopta su estructura, que es la conexión entre comportamientos y funciones. Al plantear que se puede adaptar de la misma forma tanto a funciones como a comportamientos. De la misma forma, compararlo con el modelo del NIST no tiene sentido pues el planteamiento metodológico en el que se basa es el RFB (Garbacz, 2006) quien crítica desde la filosofía lógica el citado modelo tal como se ha explicado en el apartado 2.5.3.

De la misma forma, el modelo A-QB resuelve también la cuestión relativa a la direccionalidad de las acciones al proporcionar dos sentidos trabajo, es decir, cuando una acción está actuando positivamente. Además de la generación de un eje (0, 0, 0) que indica estado neutro. Igualmente se mantiene toda la estructura ontológica planteada en DOLCE (Masolo, 2003).

OntoFaBES traduce la estructura de FaBES manteniendo la coherencia lógica requerida en una ontología de dominio, a través de la definición de diferentes propiedades. Cabe indicar que deliberadamente no se ha evaluado su métrica como por ejemplo con el método de Lozano Mello (2004) al no incluir instancias suficientes para conocer su idoneidad. Por ello, en el apartado 5.3.8 se trata en profundidad.

Finalmente en el apartado 3.3 se muestra la posibilidad de la inferencia de conocimiento a partir del modelo FaBES (indicando el ejemplo de su aplicación en un portaminas), planteamiento novedoso dentro del ámbito de las ontologías sobre diseño funcional.

Capítulo 4. KSS 2.0

En este apartado se introduce KSS 2.0, una evolución del modelo KSS (Sánchez-Moreno, 2004). El modelo KSS integra el modelo MOKA de metodología de gestión del conocimiento para sistemas KBE con los avances en el ámbito de la ingeniería ontológica (Mizoguchi, 2003; Wang, 2004; Fernández-López, 2010).

La denominación KSS 2.0 proviene al considerar el sistema como una evolución del modelo KSS haciendo uso de las nuevas tecnologías web disponibles tales como las RIA en torno al concepto popularmente conocido como Web 2.0.

Con el fin de adaptar MOKA al KSS 2.0 se han realizado una serie de modificaciones en los formularios dando prioridad a la estructura de OntoFaBES. KSS 2.0 establece una conexión entre un formulario ICARE y las diferentes clases e instancias de OntoFaBES. Esto permite asociar dos ámbitos de conocimiento distintos: el ámbito de MOKA y el ámbito ontológico representado por OntoFaBES.

Una de las diferencias en la adaptación de MOKA al KSS 2.0 es la utilización de sólo el nivel informal (Capítulo 2.3.3) a través de los formularios ICARE pues el nivel formal de MOKA tiene una complejidad excesiva (Skarka, 2007) por lo que se genera un nivel formal propio. Otra de las variaciones realizadas es que parte de la información recopilada en los formularios ICARE se obtiene directamente del modelado del diseño en un programa CAE y el diseñador puede conocer si el trabajo realizado es consistente con las premisas de partida gracias a la ontología OntoFaBES.

A continuación, se detalla la estructura de KSS 2.0 para después indicar su arquitectura. Posteriormente se comenta tanto su descripción informática como funcional relacionándolo con otros elementos como pasos previos a mostrar su funcionamiento.

4.1 KSS 2.0

KSS 2.0 es una aplicación para la captura, adquisición, formalización y compartición del conocimiento en el proceso de diseño de producto permitiendo la integración del trabajo del diseñador en el sistema KBE (Figura 61). A esto se aúna la capacidad de disponer de esa información disponible en un entorno de empresa extendida. Además de esto, incorpora un desarrollo ontológico que permite asistir al diseñador en el proceso de rediseño del producto. Para ello KSS 2.0 estructura y captura este conocimiento a partir de:

- La utilización del modelo informal de la metodología MOKA para la captura del conocimiento.
- El empleo del lenguaje ontológico OWL, lenguaje que ofrece multitud de elementos semánticos que permiten el razonamiento de la información que se ha persistido en ontologías y taxonomías.

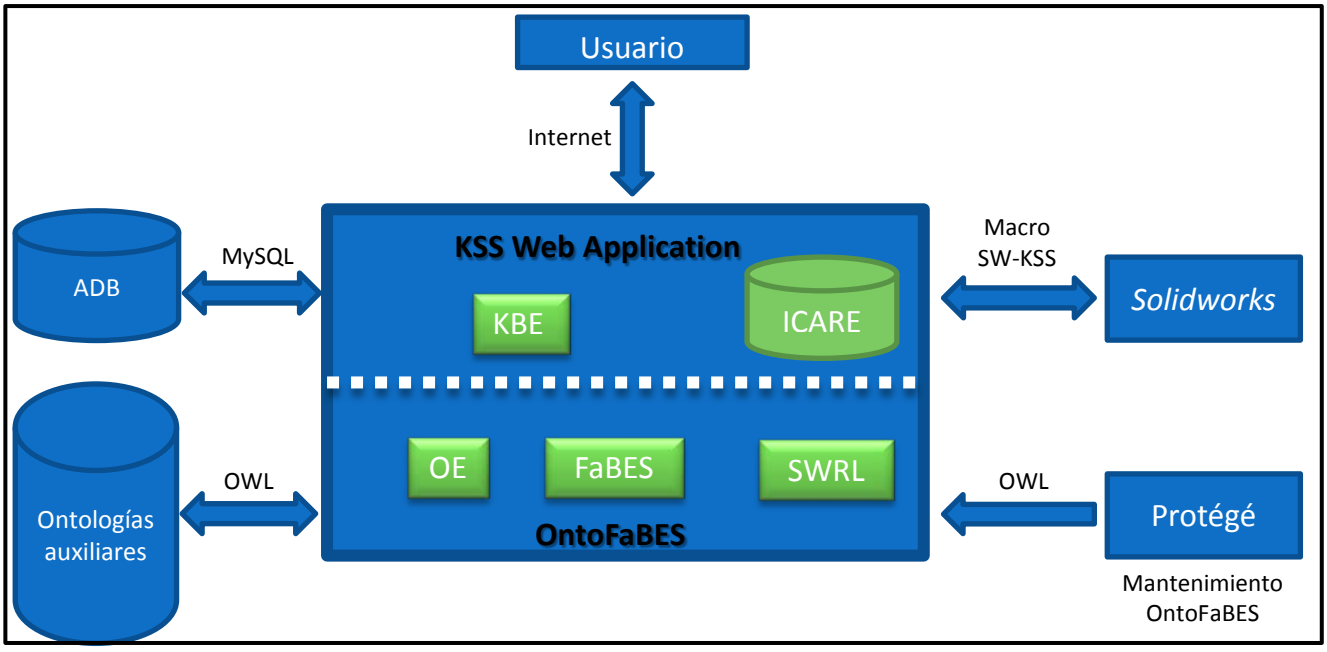


Figura 61: Arquitectura de FaBES – Relevancia de KSS 2.0

KSS 2.0 es una plataforma que está dividida en 3 secciones en la que toda la información sobre el proceso de diseño del producto está disponible abiertamente para todas las personas incluidas en éste. A continuación se indica su estructura.

Tomando como referencia el esquema del proyecto FaBES, se centra este apartado en explicar los elementos conformantes de cada uno de los elementos. De las cuatro capas básicas en las que está estructurada FaBES (Figura 13), KSS 2.0 se ubica como eje central permitiendo almacenar el conocimiento proveniente del diseño. Para que se almacene la mayor información posible KSS 2.0 la formaliza a través de los formularios ICARE de la metodología MOKA. Para ello, a partir de la información proporcionada por *Solidworks*, se ha adaptado dicha metodología al ámbito de KSS con algunas modificaciones que se condensan en la creación de la macro SW-KSS.

La macro SW-KSS permite extraer la información del diseño que se está trabajando en SW persistiéndolo en KSS a través de una API programada en el lenguaje de programación VBA. A la vez, esta información se vuelca en la ontología OntoFaBES en forma de instancias. De esta forma se establece la relación entre KSS Web Application (o también indicado como *KSS Web App*), el sistema KBE utilizado, y OntoFaBES, el sistema de Ingeniería Ontológica. Se crea de esta forma un flujo de información disponible a tres escalas, tal como se describe en la Figura 61.

Una vez persistida la información en OntoFaBES, se puede tratar el conocimiento obtenido para la inferencia de nuevas soluciones aplicando una serie de reglas, para ello se utiliza el software Protégé.

4.2 Funcionamiento

La herramienta KSS 2.0 plantea 3 roles de usuarios: el diseñador, el ingeniero de conocimiento y cualquier persona involucrada en el proyecto de (re)diseño del producto, como por ejemplo, el gestor del proyecto o clientes potenciales del producto, entre otros.

Estos 3 roles van a disponer de permisos diferentes en la utilización de la herramienta (Figura 62):

- **Diseñador**: El diseñador es el tipo de usuario que va a utilizar *Solidworks* y la macro SW-KSS pues conoce la información relativa a la modificación/creación del diseño del producto. Para ello no necesita disponer de ningún conocimiento informático añadido para poder manejar el sistema.
- **Ingeniero de conocimiento:** Es el único que va a poder acceder a todo el sistema KSS 2.0 para establecer cualquier modificación relativa como por ejemplo la edición de OntoFaBES mediante Protégé.
- **Miembro del proyecto:** Este tipo de usuario solo podrá acceder a la KSS Web App y dependiendo de los permisos que el administrador determine podrá visualizar, editar o crear nuevo contenido del proyecto de diseño del producto en el que se esté trabajando.

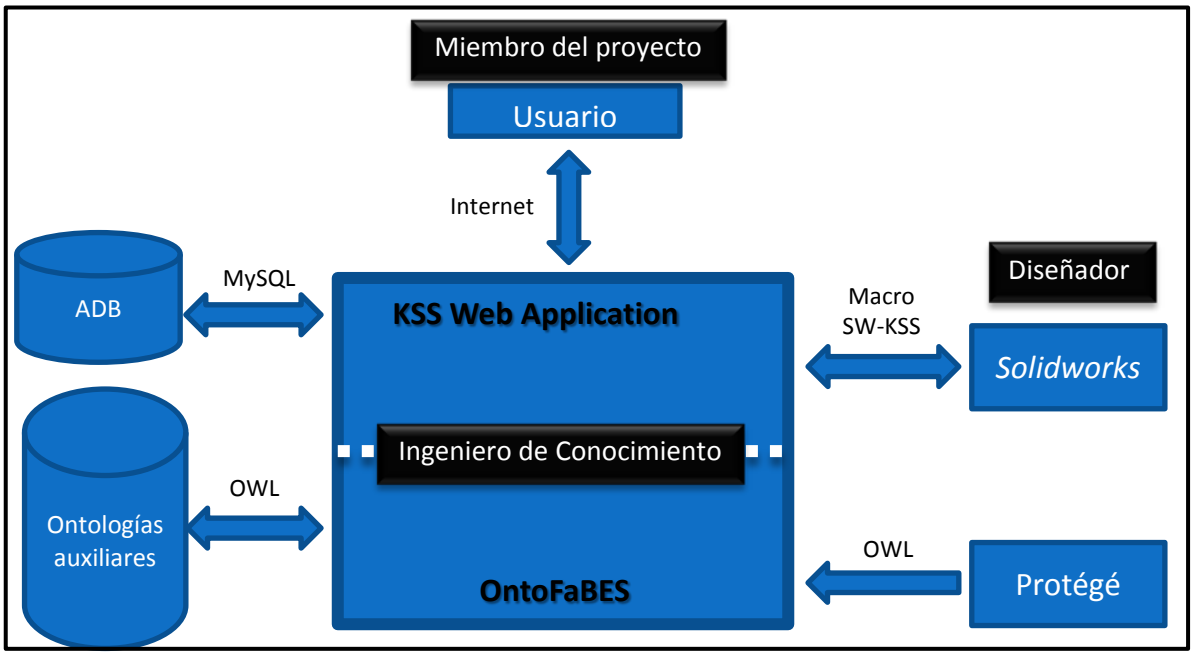


Figura 62: Distribución de roles en el funcionamiento de KSS 2.0

4.2.1 Captura del conocimiento: Macro KSS –*Solidworks*

El objetivo de la macro que se lanza a través de un icono en *Solidworks* es persistir el conocimiento útil para la formalización del conocimiento en KSS 2.0 para ser modelado posteriormente a través de OntoFaBES. Por tanto, se formaliza la información relativa a los formularios Entity y Constraints ya que la información que dispone *Solidworks* de un diseño es la relativa a los elementos que conforman su estructura (formulario Entity) y las restricciones de las posiciones que ocupan dichos elementos (formulario Constraint).

Se constituye así una herramienta que a la vez que persiste y formaliza la información disponible, también se muestra al usuario de *Solidworks*. Esto permite en la etapa de rediseño de un producto mejorar el conocimiento de un diseño creado por otro usuario, porque de manera rápida se pueden conocer las piezas que componen un conjunto y sus grados de libertad. Y además se tiene conocimiento de cuál es la información que se formaliza en cada momento.

Tal como se ha indicado en el capítulo anterior, el usuario que generalmente utilizará esta macro será un diseñador. Para ello, una vez que se ha ejecutado la macro, para persistir la información en el KSS, se debe actualizar, proceso que suele tardar alrededor de 2 minutos dependiendo siempre de la cantidad de información que se quiera persistir[mm].

4.2.2 Compartición del conocimiento: KSS Web Application

Para crear un nuevo proyecto de diseño, se accede al menú *File* y se selecciona la opción *New Project…* Aparecerá un diálogo en el que se introducirá el nombre o referencia del proyecto de diseño y una descripción breve.

Si lo que se desea es cargar un proyecto de diseño ya existente, que será la opción más común debido a la información almacenada a través de la macro SW-KSS, se accede al menú *File* y se selecciona la opción *Open Project* ... y aparecerá un diálogo en el que se mostrarán todos los proyectos de diseño disponibles.

Una vez creado o cargado un proyecto de diseño, el usuario de KSS puede crear un nuevo formulario ICARE o asociar al proyecto actual uno ya existente asociado a otro proyecto de diseño. Para crear un nuevo formulario ICARE, se accede al menú *ICARE Forms*, se selecciona *Create New ICARE Form...* y se escoge qué tipo de formulario se quiere crear. Dependiendo del tipo de formulario, aparecerá un diálogo de creación ligeramente distinto.

Desde este diálogo se puede dar de alta un nuevo formulario ICARE y asociarlo al proyecto de diseño actual, introduciendo la información necesaria para cumplimentar el mismo así como establecer las relaciones con otros formularios y con las instancias de la ontología que se estimen oportunas. En la Figura 63 se puede ver el diálogo de creación de un formulario *Constraint* así como el subdiálogo que permite seleccionar los formularios *Entity* a los que se asociará el formulario Constraint (Figura 64).

Si lo se quiere es asociar un formulario existente a un proyecto de diseño, se dispone de la opción *Add an Existing ICARE Form...*del menú *ICARE Forms* (Figura 65). Se mostrará un diálogo en el que se ha de seleccionar uno de los proyectos existentes en el sistema y seleccionar los formularios ICARE que se quieren asociar al proyecto de diseño actual. Mediante los botones de selección de la parte izquierda se pueden filtrar los formularios ICARE por tipo, haciendo más sencilla la búsqueda de los mismos. De esta manera, se pueden reutilizar los formularios ICARE ya completados en distintos proyectos de diseño.

Para modificar o eliminar un formulario ICARE, es suficiente con seleccionarlo en la vista de árbol lateral, el formulario ICARE se cargará en la parte derecha. Se seleccionarán los botones *Modify* si se quiere modificar el formulario o *Delete* en el caso que se quiera eliminar permanentemente el formulario.

[mm] Las imágenes de los diseños se persisten en un ordenador local y no en el servidor para agilizar el procesamiento de la información.

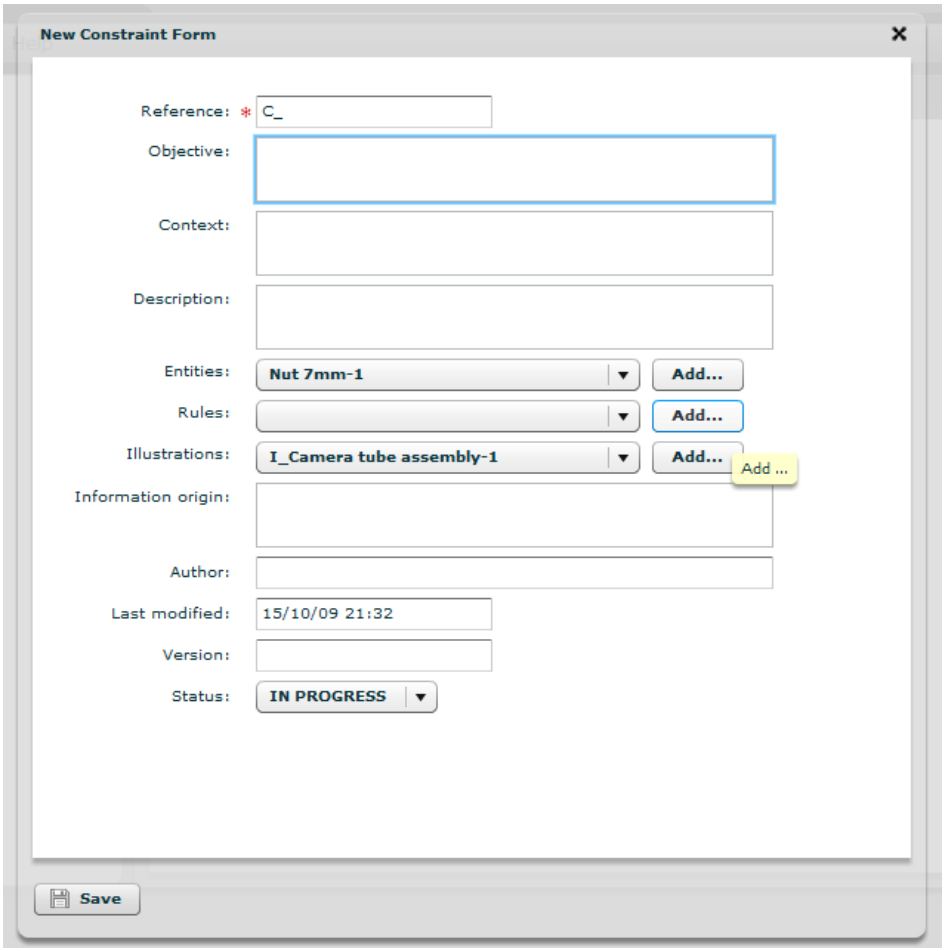


Figura 63: Diálogo Nuevo Formulario ICARE-Constraint

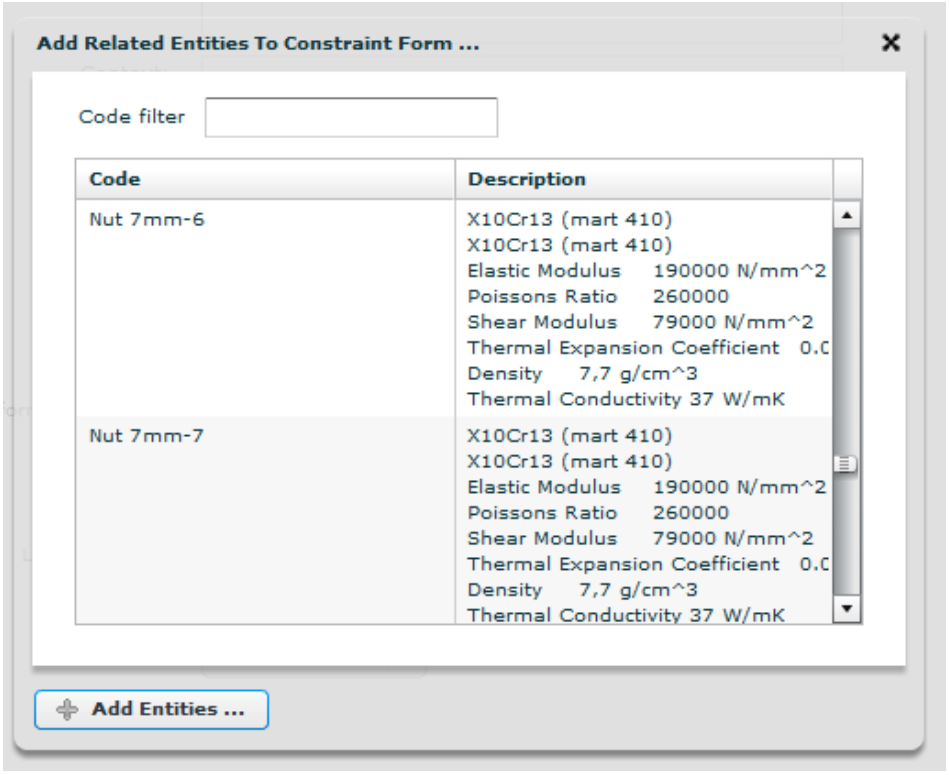


Figura 64: Subdiálogo que permite la selección de formularios Entity

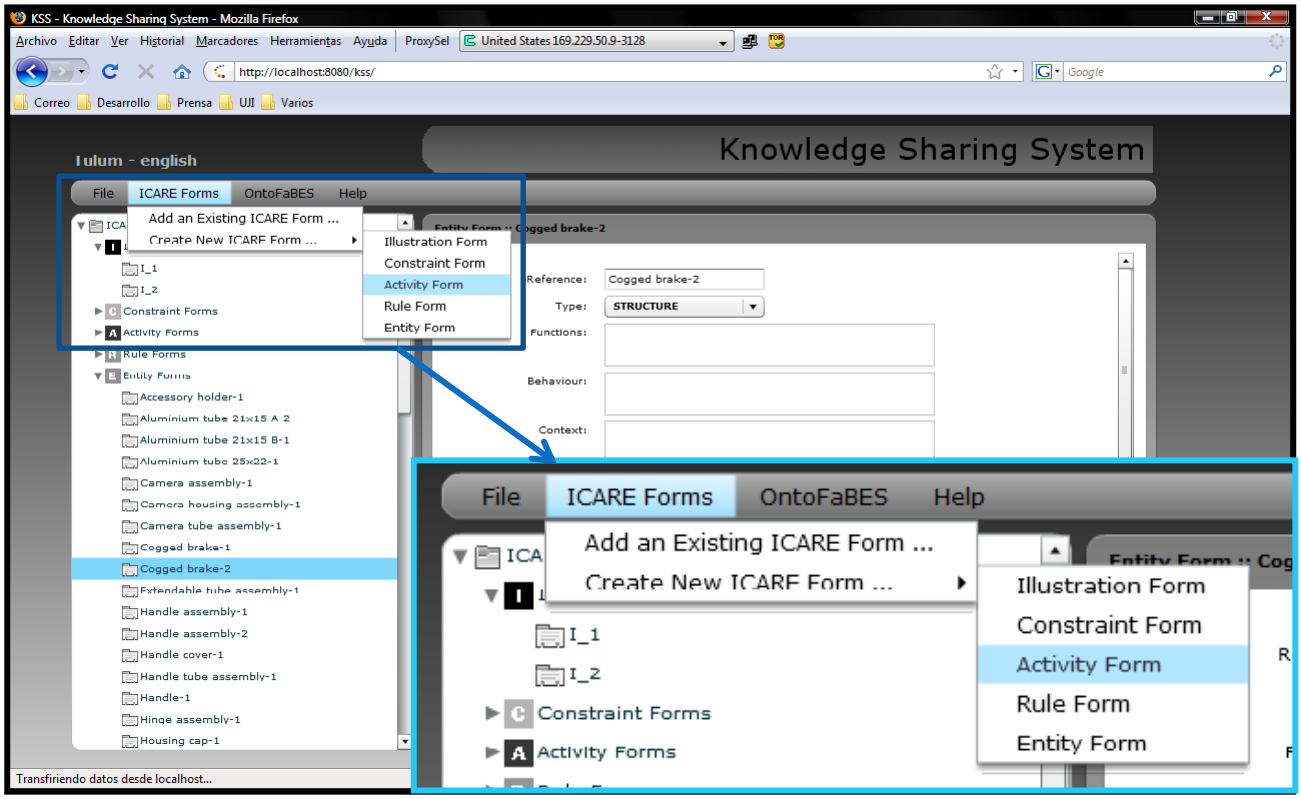


Figura 65: KSS 2.0 – Menú ICARE Forms

Visualización

KSS dispone de una herramienta visual que permite la visualización de los formularios ICARE en forma de grafo, permitiendo de una forma rápida ver las relaciones entre los formularios de un proyecto de diseño determinado. Ésta es accesible desde el menú *ICARE Forms*, que al señalar un formulario determinado se selecciona la opción *Graph View*. Esta visualización permite ver las relaciones existentes entre el formulario y aquellos que están relacionados.

En esta visualización se ha seguido la nomenclatura de colores recomendada por el consorcio MOKA, nodos de color azul para los formularios Entity, de color rojo para los formularios Activity, negro para los formularios *Illustration*, naranja para los formularios Rule y rosa para los formularios Constraint.

De forma similar a la que se pueden visualizar el histórico y las relaciones entre formularios ICARE, KSS permite visualizar OntoFaBES así como ver qué instancias de la ontología están asociadas a un formulario ICARE.

La visualización de OntoFaBES asociada al proyecto de diseño se accede desde el menú *OntoFaBES*, seleccionando la opción *Visualize* (Figura 66). Esta visualización muestra en forma descendente la ontología asociada y las instancias que se relacionan a cada una de las clases de ésta.

Una vez persistida la información en OntoFaBES, se puede tratar el conocimiento obtenido para la inferencia de nuevas soluciones aplicando una serie de reglas, para ello se utiliza el software Protégé (Figura 67).

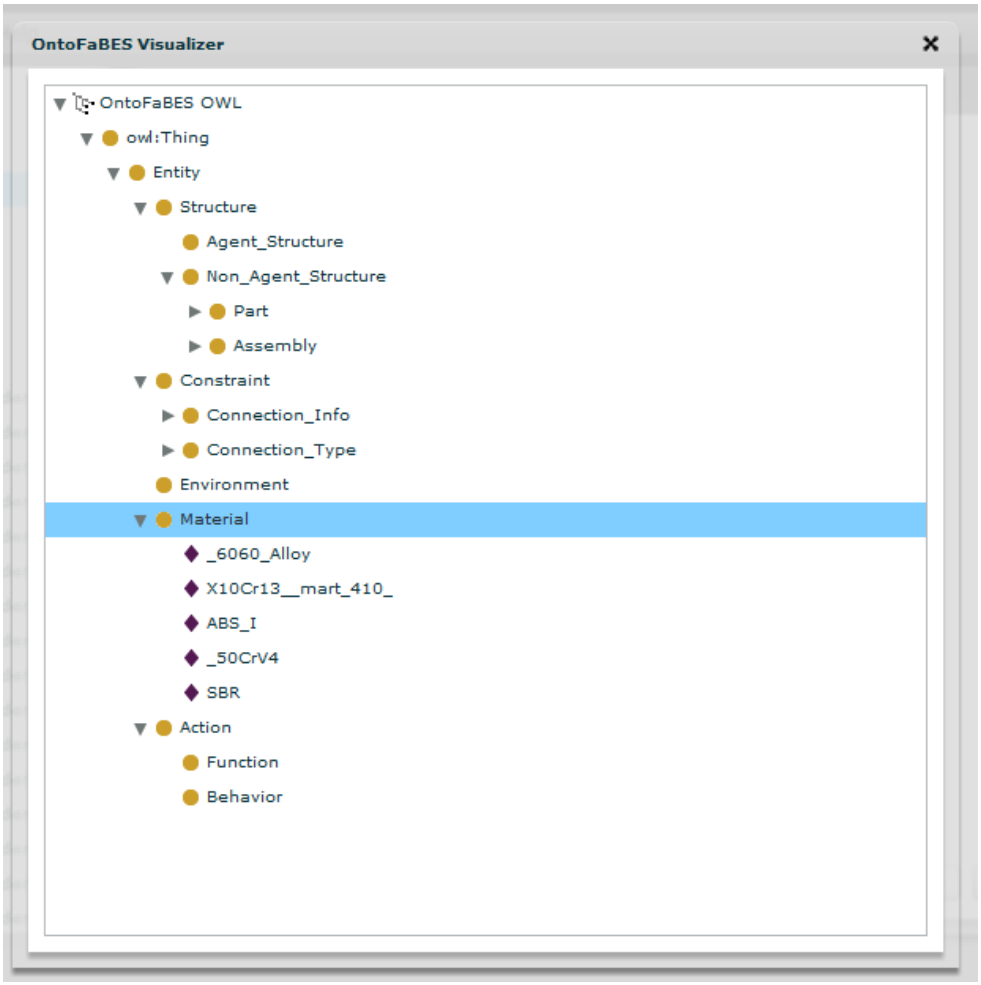


Figura 66: Visualizador de KSS en OntoFaBES

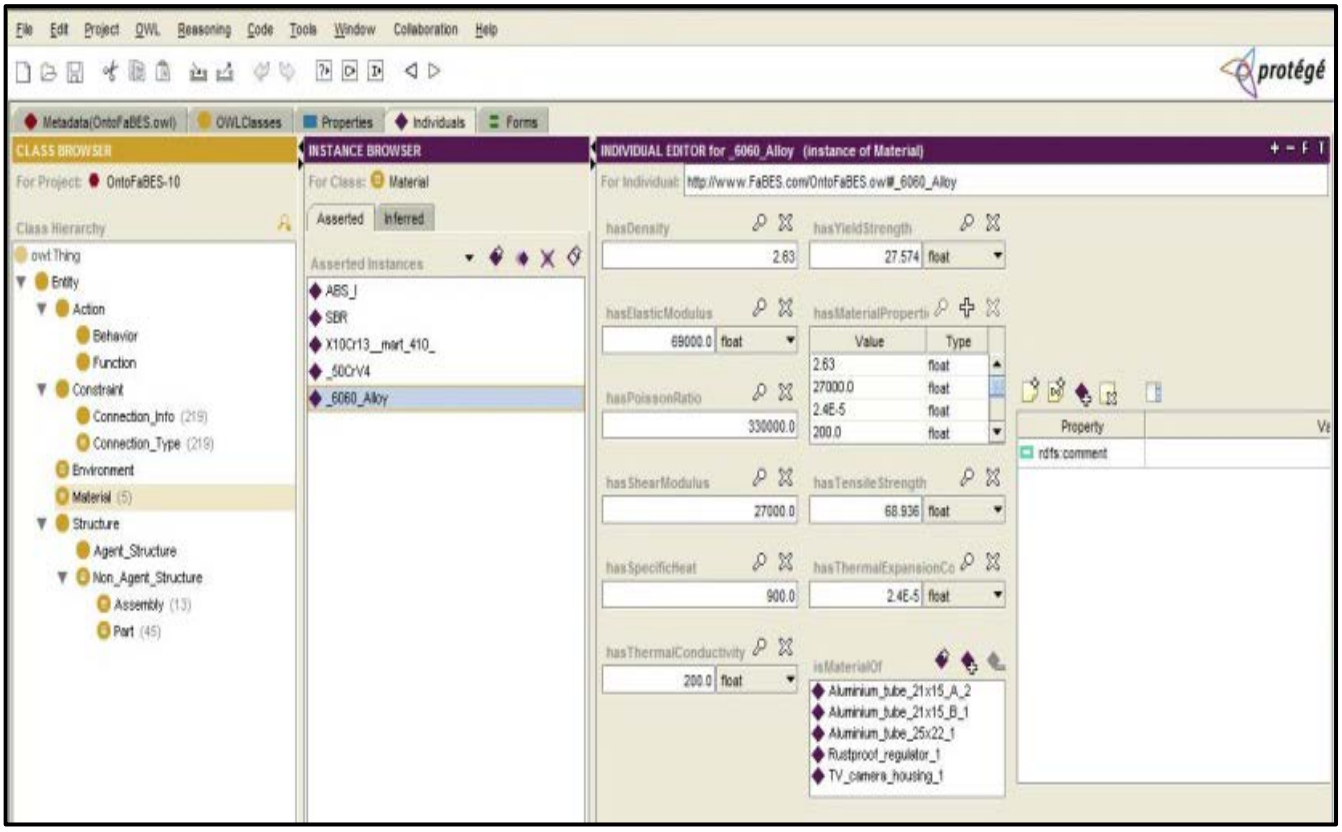


Figura 67: Visualización de la ontología en Protégé

4.3 Descripción funcional

En este apartado se trata de la funcionalidad de la aplicación consistente en tres apartados diferenciados según el flujo de información detallado en la Figura 61:

1. Se trata la captura de conocimiento a partir de la macro KSS que se establece en *Solidworks*.
2. Una vez que se dispone del conocimiento formalizado, se explica cómo se comparte a través de la aplicación web KSS.
3. Finalmente se explica el significado de los formularios ICARE, cuya información posteriormente será organizada y modelada según la ontología OntoFaBES.

4.3.1 Captura del conocimiento: Macro KSS –*Solidworks*

Esta macro está dividida en tres secciones principales: una donde se muestra el módulo *EntityForms*, una segunda indicada para el apartado Project referido a la denominación y descripción del proyecto y la parte derecha donde se muestra los módulos de información sobre el proyecto, dividido en tres pestañas: *Entity*, *Constraints* y *Illustrations Info*. A continuación se va a explicar en profundidad cada uno de los módulos:

EntityForms: Jerarquía de las partes y ensamblajes del proyecto de diseño en estudio (Figura 68).

Entity Info: Información sobre los distintos materiales del modelo así como de sus propiedades físicas relevantes (Figura 69). En este apartado se muestra la imagen de la pieza, la denominación proporcionada por el SW (*Entity Name*), su masa (*Mass (gr.)*) y las propiedades físicas si está compuesto por sólo un material (*Material Info*).

Constraints Info: En este apartado se indican las restricciones existentes. (*ConstraintsForms*) (Figura 70).

Respecto a cada una de las restricciones se indica la información disponible. Generalmente al consistir en las relaciones de posición de cada uno de los ensamblajes del proyecto, se indica el tipo de asociación (*Mate Type*) y de alineación (*AlignmentType*, *Can be flipped*?).

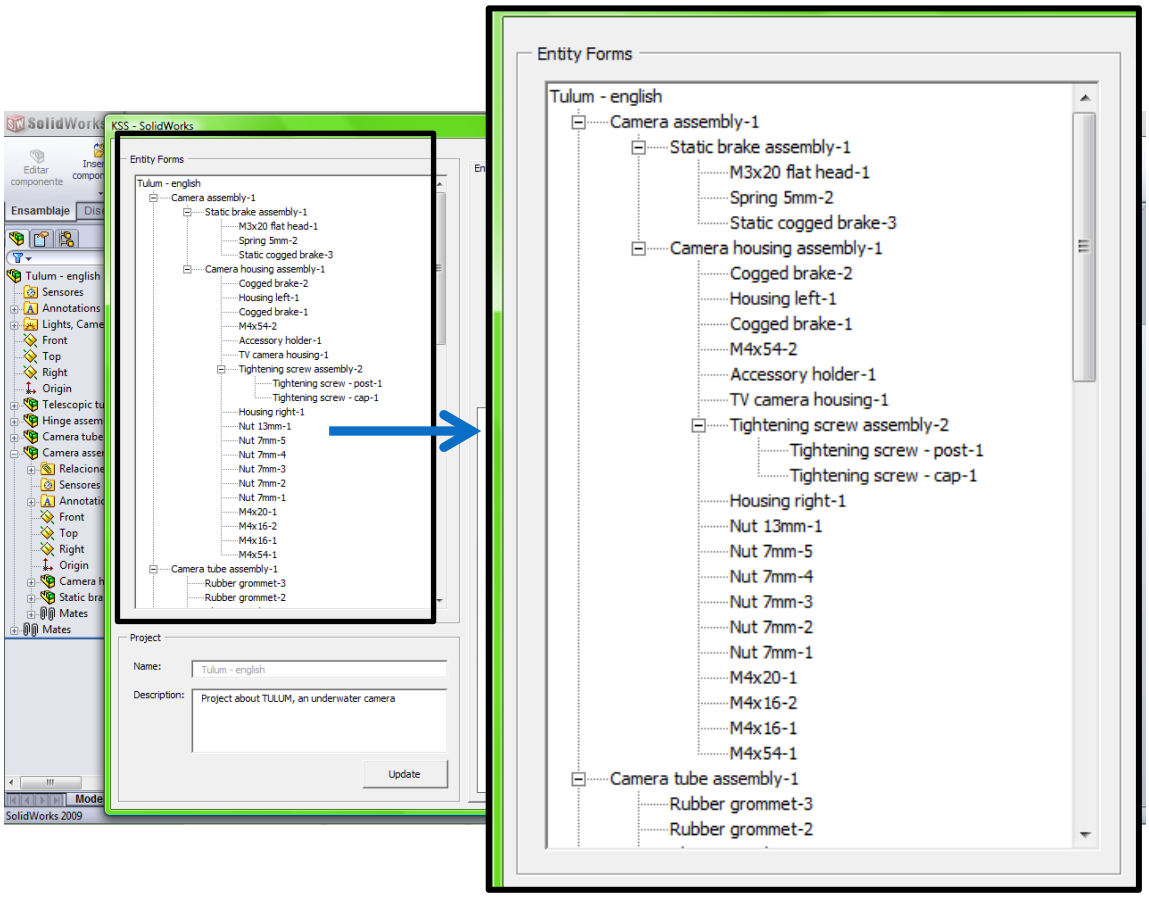


Figura 68: Entity Forms

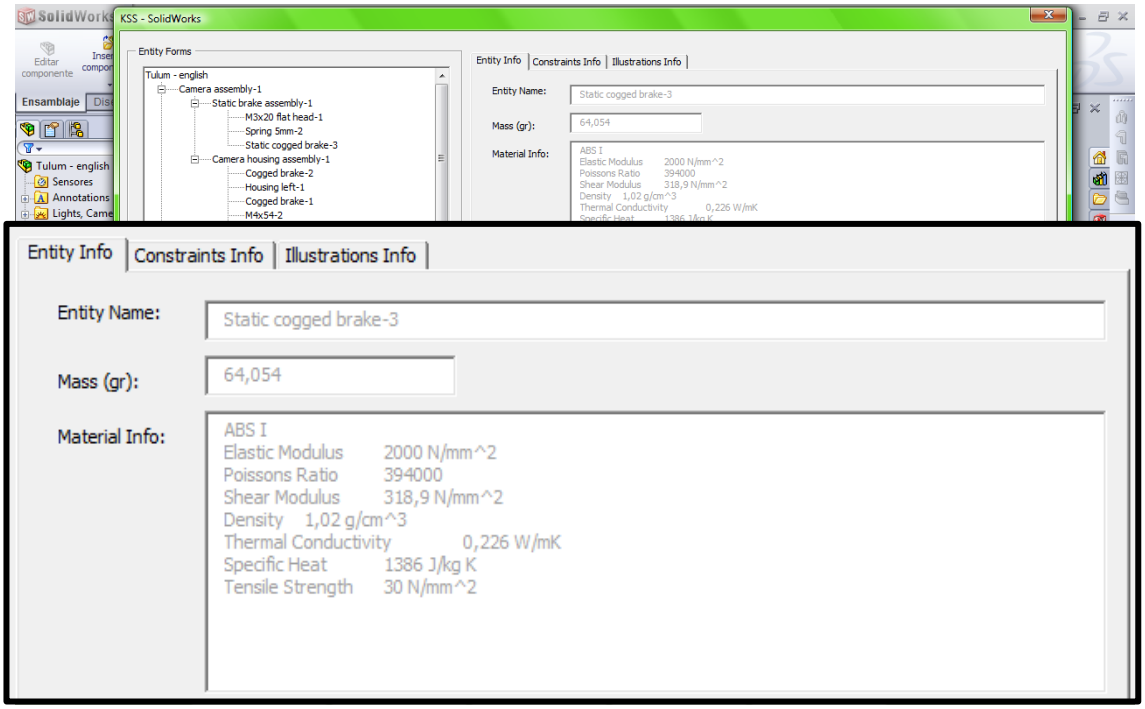


Figura 69: Entity Info

A partir de los datos almacenados en el formulario de entidades relacionadas (*RelatedEntityForms*) se muestra también la denominación e imagen de las dos piezas que conforman la restricción, indicando las coordenadas relativas, la relación de posición y si el tipo de asociación es concéntrica, mostrando el radio.

Illustrations Info: Este formulario (Figura 71) indica las ilustraciones utilizadas en la definición de las distintas piezas del proyecto (*Illustrations*) y la ruta de acceso (*ResourcePath*).

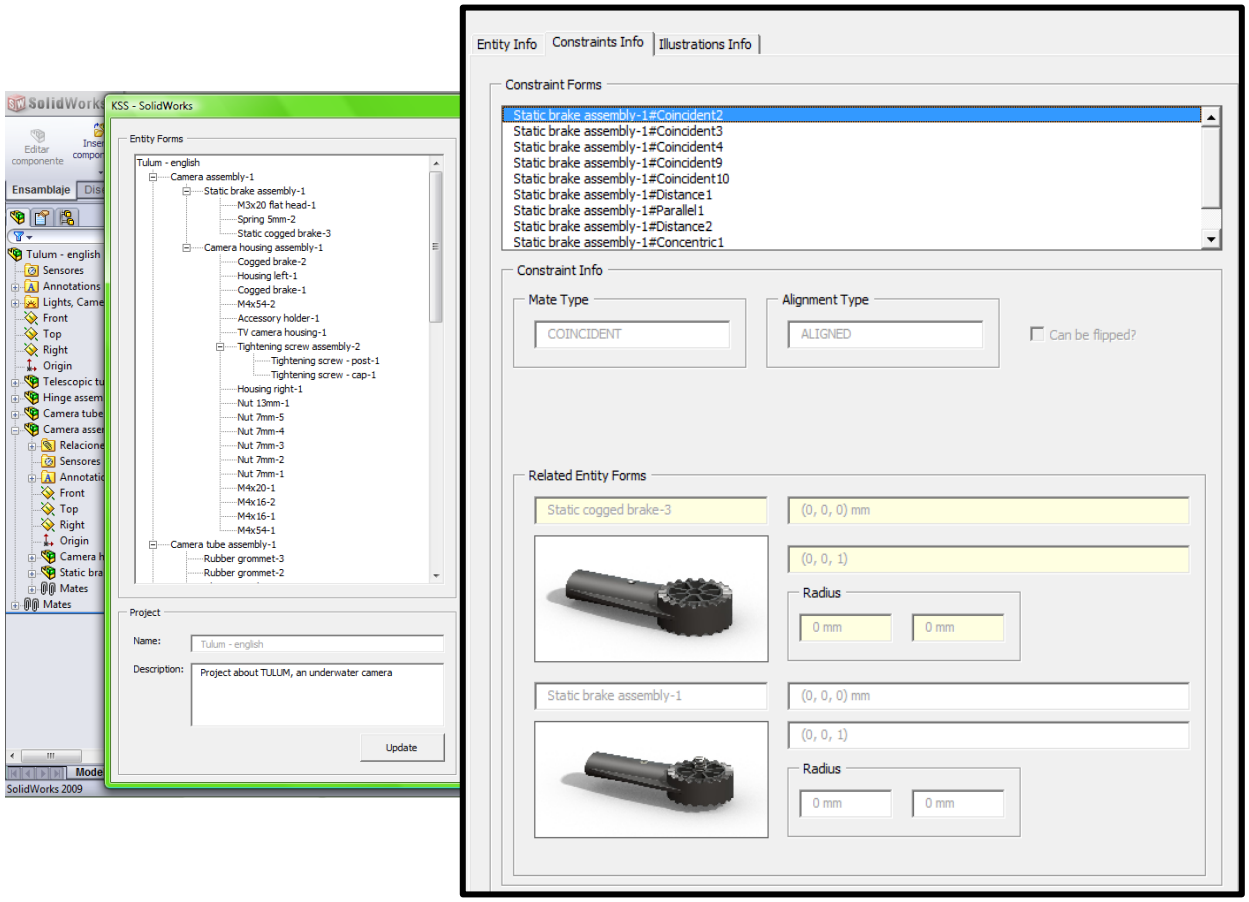


Figura 70: Constraints Info

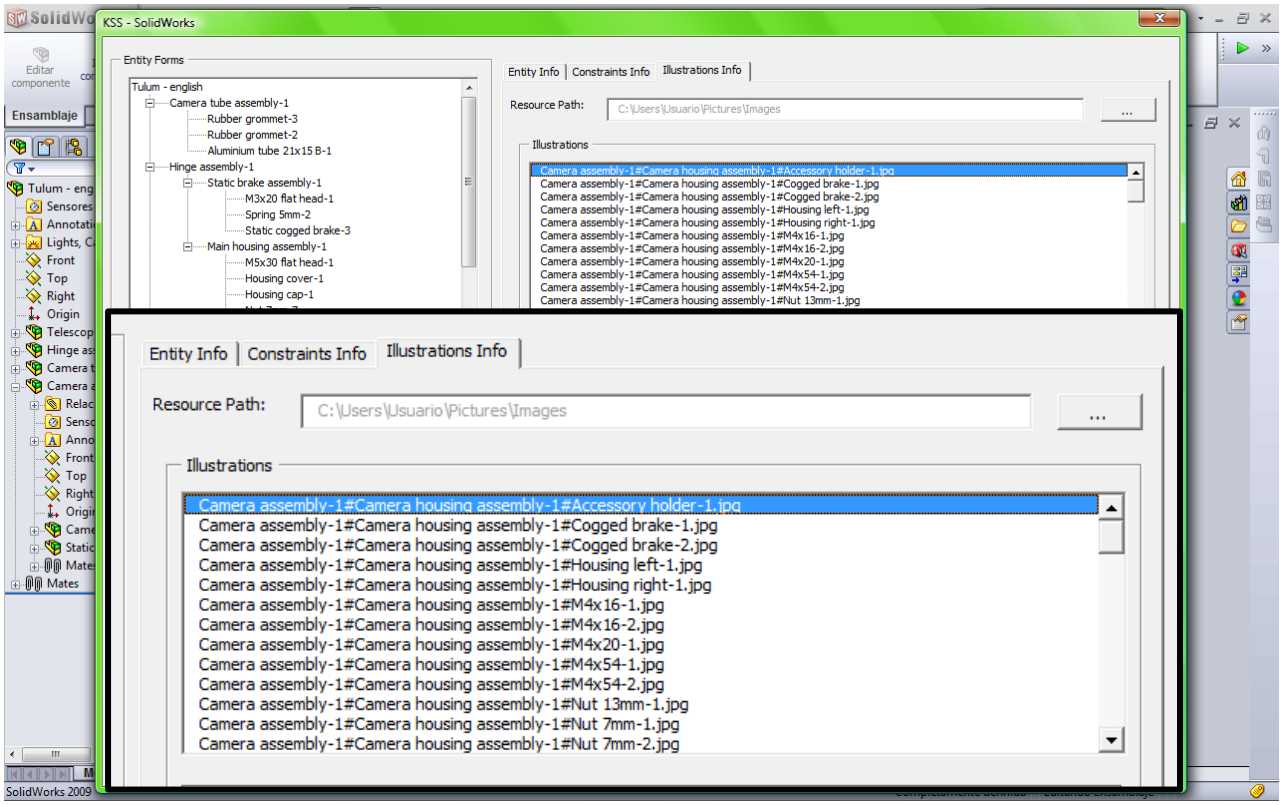


Figura 71: Illustrations Info

4.3.2 Compartición del conocimiento: KSS Web Application

Para la captura del conocimiento se ha empleado el modelo informal que nos ofrece la metodología MOKA, formada por los formularios ICARE. Con el objetivo de automatizar la obtención de información, se obtiene la información a partir de la macro SW-KSS tal como se ha indicado en el capítulo anterior.

La vista principal de KSS (Figura 72) está compuesta por tres secciones claramente diferenciadas:

- Un *menú principal* situado en la parte superior que permite realizar todas las tareas disponibles en el sistema.
- Una *vista de formularios ICARE* de forma descendente en la parte izquierda, que permite navegar y visualizar cada uno de los formularios ICARE asociados al proyecto de diseño actual.
- Una vista de *detalle/edición de formularios ICARE* en la parte derecha.

El menú principal está compuesto por las siguientes opciones:

- Menú *File*: podemos crear un nuevo proyecto de diseño *New Project... (*Figura 73*)* o cargar uno ya existente *Open Project... (*Figura 74*)*.
- Menú *ICARE Forms*: desde este menú se puede añadir un formulario ICARE desde otro proyecto de diseño *AddAnExisting ICARE Form ...*, crear un nuevo formulario ICARE *Create New ICARE Form ...*, ver el histórico de formularios ICARE *View Form Historial* o visualizar todos los formularios con sus relaciones *View Project Forms*.
- Menú *Ontology*: permite asociar una ontología OWL al proyecto de diseño o modificar una ya asociada: *Add/ModifyRelatedOntology*; visualizarla gráficamente: *View RelatedOntology;* y visualizar los formularios ICARE conectados con las instancias de las clases de la ontología: *View RelatedOntology / ICARE Forms*.
- Menú *Help*: muestra información adicional sobre la aplicación *About ...*

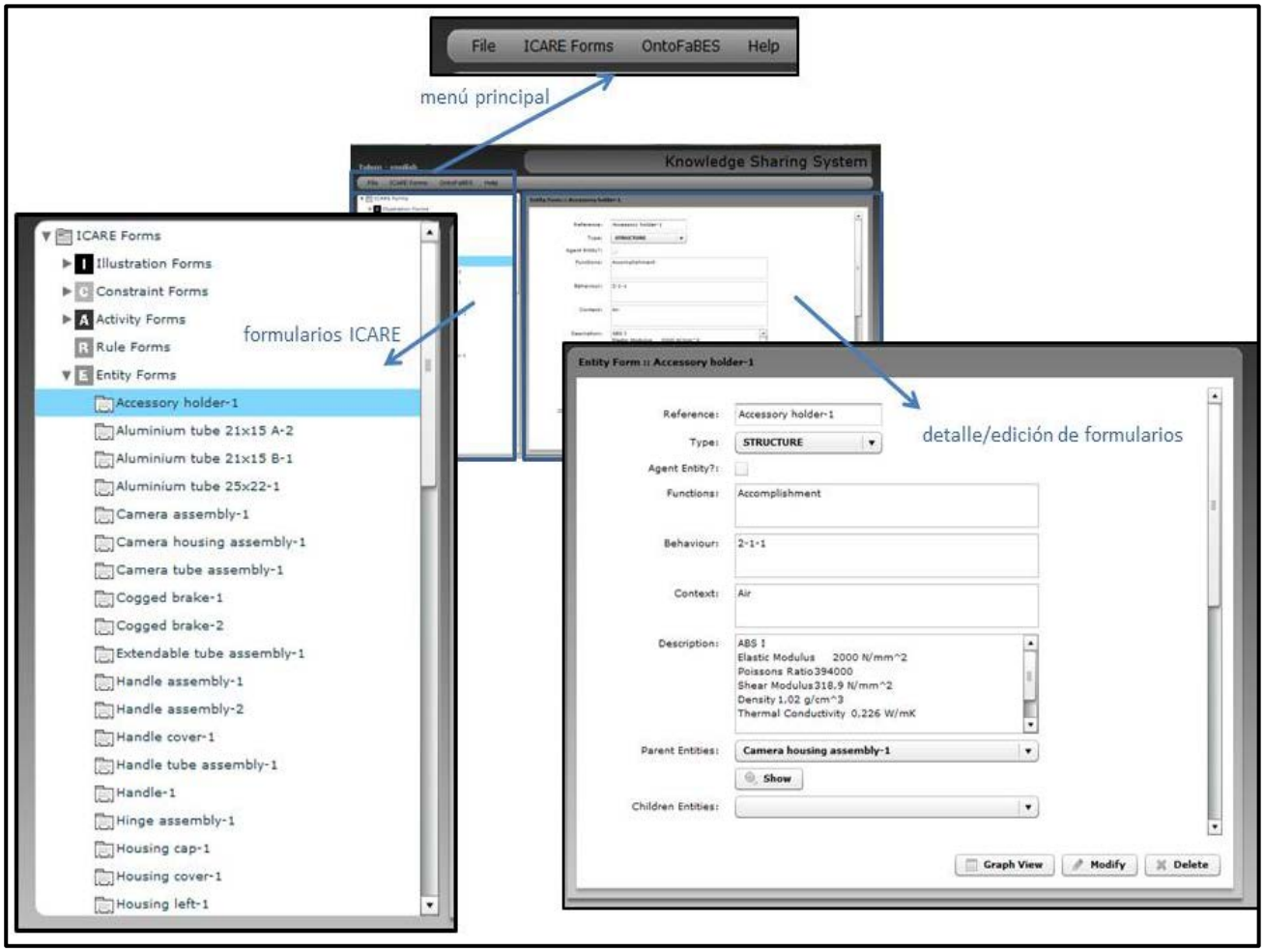


Figura 72: Vista general de KSS

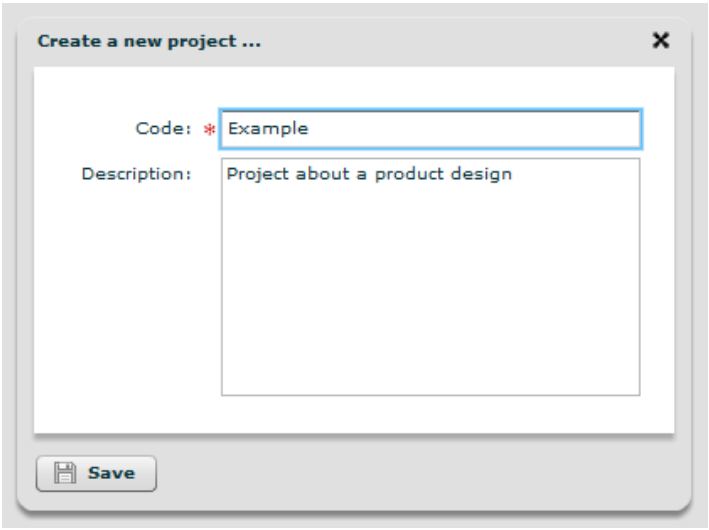


Figura 73: Diálogo del nuevo proyecto de diseño

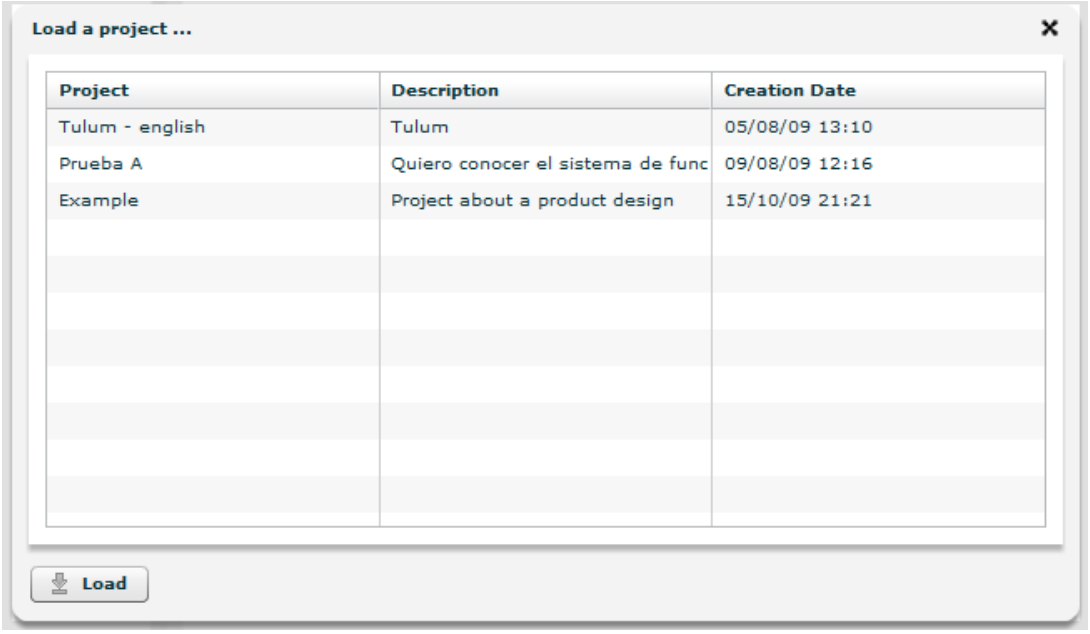


Figura 74: Diálogo Carga Proyecto de Diseño

4.3.3 Formalización del conocimiento: KSS-OntoFaBES

Para poder relacionar OntoFaBES con KSS 2.0 se estableció la siguiente analogía entre elementos de OntoFaBES para cada uno de los formularios ICARE (capítulo 2.3.3.2) de MOKA: *Illustrations*, *Constraints*, *Activities*, *Rules* and *Entities* (Tabla 25). Para poder entender el formato de dicha tabla es necesario explicar el significado de cada una de las columnas:

- **FORM**: En esta columna se indica qué tipo de formulario ICARE se trata en concreto. Por ejemplo, *Illustrations*.
- **CHARACTERISTICS**: Se indican los diferentes apartados que conforman un tipo de formulario ICARE en KSS. Por ejemplo para el formulario *Illustrations* puede ser *Context.*
- **DESCRIPTION**: Se establece la explicación resumida del significado de la característica referida. Para el ejemplo indicado la descripción sería *"Contexto, información, validez; Notas de referencia; Campo de texto libre a partir de notas".*
- **OntoFaBES**: La relación existente con la ontología OntoFaBES, que puede relacionarse como una clase o una propiedad. En este caso es la clase *"RelatedIllustrations: VisualResource."*

Tabla 25: Plantilla de la analogía entre OntoFaBES con los formularios ICARE en KSS 2.0

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
Formulario ICARE	Características del formulario ICARE.	Descripción del formulario.	Clase en OntoFaBES
Illustrations	*Context*	*Contexto, información, validez;….*	*RelatedIllustrations: VisualResource*

A continuación se detallan los apartados relativos a cada uno de los formularios ICARE en base a la plantilla indicada (Tabla 20).

4.3.3.1 Formulario ICARE: Illustrations

En este formulario (Tabla 26) se obtiene el conocimiento referido a las ilustraciones que completan el conocimiento adquirido del proyecto de diseño utilizado en el programa CAE *Solidworks*.

La información consiste en las imágenes de las diferentes partes y ensamblajes que conforman el diseño conformado en *Solidworks* y notas relativas a ello (*Context y Description*) que son editadas en el *KSS Web*

Application, mientras que el origen de la información, autor y versión se obtienen de manera automática de *Solidworks*.

Este conocimiento se formaliza en OntoFaBES a través de la clase *RelatedIllustrations: VisualResource*.

Tabla 26: Descripción del formulario Illustrations

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
Illustrations			∗*RelatedIllustrations: VisualResource*
	Context	Contexto, información, validez; Notas de referencia; Campo de texto libre a partir de notas	
	Description	Cuestiones técnicas del diseño. Campo de texto libre	
	Information Origin	De *Solidworks*	
	Author	*Solidworks*	
	Version	*Solidworks*	

4.3.3.2 Formulario ICARE: Constraints

En este formulario (Tabla 27) se muestra el conocimiento referido a las restricciones que se establecen a partir de las relaciones de posición entre las diferentes piezas en el diseño del producto.

La información consiste, por una parte, en la finalidad de la restricción y su relación entre los formularios *Activity*, las notas descriptivas de cada una de las restricciones que se haya en el diseño del producto (*Context* y *Description*) y las relaciones existentes de este formulario con los restantes (*Entities, Rules, Illustrations*). Dicha información se edita en el *KSS Web Application*. Y por otra parte, el origen de la información, autor y versión se obtienen de manera automática de *Solidworks*.

Esta información queda englobada en OntoFaBES en la clase *Constraint y subclases Connection_Info y Connection_Type.*

Tabla 27: Descripción del formulario Constraints

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
Constraints		En el *Solidworks* se consideran como Relaciones de Posición // Mates	∗ Englobado en la clase *Constraint* y subclases *Connection_Info* y *Connection_Type* ∗ Se obtienen de las relaciones de posición que establece el diseñador en la pieza.
	Objective	La finalidad de la regla, relación entre actividades en texto plano. Salida del KSS	
	Context	Contexto, información, validez; Notas de referencia; Campo de texto libre a partir de notas	
	Description	Cuestiones técnicas del diseño en un campo de texto libre	
	Entities	-	
	Rules	-	
	Illustrations	-	
	Information Origin	De *Solidworks*	
	Author	*Solidworks*	
	Version	*Solidworks*	

4.3.3.3 Formulario ICARE: Activities

En este formulario (Tabla 28) se obtiene el conocimiento referido a los procesos de diseño del producto.

La información que muestra este formulario proviene de *KSS Web Application* en todos los apartados excepto el origen de la información, autor y versión se obtienen de manera automática de *Solidworks*. A la vez que desde OntoFaBES se puede proporcionar el conocimiento de las relaciones entre acciones: *Parent Activities*; *Children Activities*; *Precedent Activities*; y *Following Activities*.

Esta información queda englobada en OntoFaBES en la clase *Action,* sus subclases respectivas y las propiedades *isBehavioredBy / isF-CarriedByOn; isA-partOf; is_A-MadeUpBy; is_A-PrecededBy* e *is_A-FollowedBy*.

Tabla 28: Descripción del formulario Activities

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
Activities	*Trigger*[nn]	Conocimiento proveniente de KSS Web Application	*isBehavioredBy / isF-CarriedByOn*
	Input	Conocimiento proveniente de KSS Web Application	
	Output	Conocimiento proveniente de KSS Web Application	
	Potential Failure Modes	Explicación en lenguaje natural de reglas; Salida de KSS; Atención a las restricciones. Información que rellenaría el fabricante del producto	
	Objective	• La finalidad de la acción en texto plano. • KSS Web Application	
	Input requirements	• Explicación en lenguaje natural de las restricciones • KSS Web Application	
	Context	• Contexto, información, validez. • Notas de referencia • Campo de texto libre • A partir de notas	
	Description	• Cuestiones técnicas del diseño • Campo de texto libre	
	Entities	-	
	Parent Activities	• Ontología OntoFaBES • Ingeniero de conocimiento	*isA-PartOf*
	Children Activities	*Ídem*	*is_A-MadeUpBy*
	Precedent Activities	*Ídem*	*is_A-PrecededBy*
	Following Activities	*Ídem*	*is_A-FollowedBy*
	Rules	-	
	Illustrations	-	
	Information Origin	De *Solidworks*	
	Author	*Solidworks*	
	Version	*Solidworks*	

4.3.3.4 Formulario ICARE: Rules

En este formulario (Tabla 29) se muestran aquellas reglas, fórmulas y algoritmos para optimizar la inferencia de conocimiento en OntoFaBES. La información que muestra este formulario proviene de OntoFaBES, editable en *KSS Web Application*. Esta información queda englobada en OntoFaBES en el apartado de Reglas SQWRL (Aptdo. 3.3.1).

4.3.3.5 Formulario ICARE: Entities

En este formulario (Tabla 30) se obtiene el conocimiento referido a los objetos del dominio de conocimiento del producto que se está diseñando como pueden ser componentes, ensamblajes, partes y características.

La información que muestra este formulario proviene de *KSS Web Application* en todos los apartados excepto el origen de la información, autor y versión se obtienen de manera automática de *Solidworks*. A la vez que desde OntoFaBES se puede proporcionar el conocimiento de las relaciones entre acciones: *Parent Entities*; *Children Entities*; y *Other Related Entities*.

Esta información queda englobada en OntoFaBES en la clase *Structure,* sus subclases respectivas y las propiedades *isAssembledBy; isPartOf; hasAction;* y *Visual Resource*.

[nn] El modelo de conocimiento formal puede ser entendido no sólo por personas sino también por sistemas informáticos desencadenando así acciones controladas (*trigger*).

Tabla 29: Descripción del formulario Rules

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
Rules		Son fórmulas/algoritmos de cálculo Reglas en la ontología Ej. La cámara está estanca porque está en un medio acuático	• Las reglas pueden estar relacionadas tanto con la información proveniente de las acciones como de las estructuras.
	Objective	La finalidad de la regla, relación entre actividades Texto plano Salida del KSS	
	Context	Contexto, información, validez Notas de referencia Campo de texto libre A partir de notas	
	Description	De OntoFaBES	
	Entities	*Ídem*	
	Activities	*Ídem*	
	Rules	*Ídem*	
	Illustrations	*Ídem*	
	Information Origin	*Ídem*	
	Author	*Ídem*	
	Version	*Ídem*	

Tabla 30: Descripción del formulario Entities

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
Entity			*Structure*
	Functions	• Ontología OntoFaBES • Ingeniero de conocimiento	
	Behaviors	• Ontología OntoFaBES • Ingeniero de conocimiento	
	Context	• Contexto, información, validez • Notas de referencia • Campo de texto libre • A partir de notas	
	Description	• Cuestiones técnicas del diseño • Campo de texto libre	
	Parent Entities	taxonomía: jerarquía del ensamblaje	*isAssembledBy*
	Children Entities	taxonomía: jerarquía del ensamblaje	*IsPartOf*
	Other Related Entities	taxonomía: jerarquía del ensamblaje están al mismo nivel	*(campo de estructuras pasivas)*
	Activities	-	*hasAction*
	Constraints	-	*Solidworks*
	Rules	-	*Rules: KBE ->SWRL*

FORM	CHARACTERISTICS	DESCRIPTION	ONTOFABES
	Illustrations	-	*VisualResource*
	Information Origin	De *Solidworks*	
	Author	*Solidworks*	
	Version	*Solidworks*	

4.4 Arquitectura del KSS 2.0

La arquitectura del KSS 2.0 está dividida en dos partes claramente diferenciadas: una parte cliente y otra parte servidora (Figura 75).

La parte cliente la forman los distintos clientes que se benefician de las funcionalidades de la arquitectura. OntoFaBES proporciona dos clientes:

- KSS Web Client
- *Solidworks* Client

El cliente KSS Web Client es el cliente nativo del sistema KSS. Los usuarios podrán consultar proyectos de diseño y gestionar los formularios ICARE de los mismos desde esta aplicación web. Esta aplicación web es el front-end genérico de KSS, y está desarrollada utilizando tecnologías Web 2.0 como es la RIA Flex de Adobe.

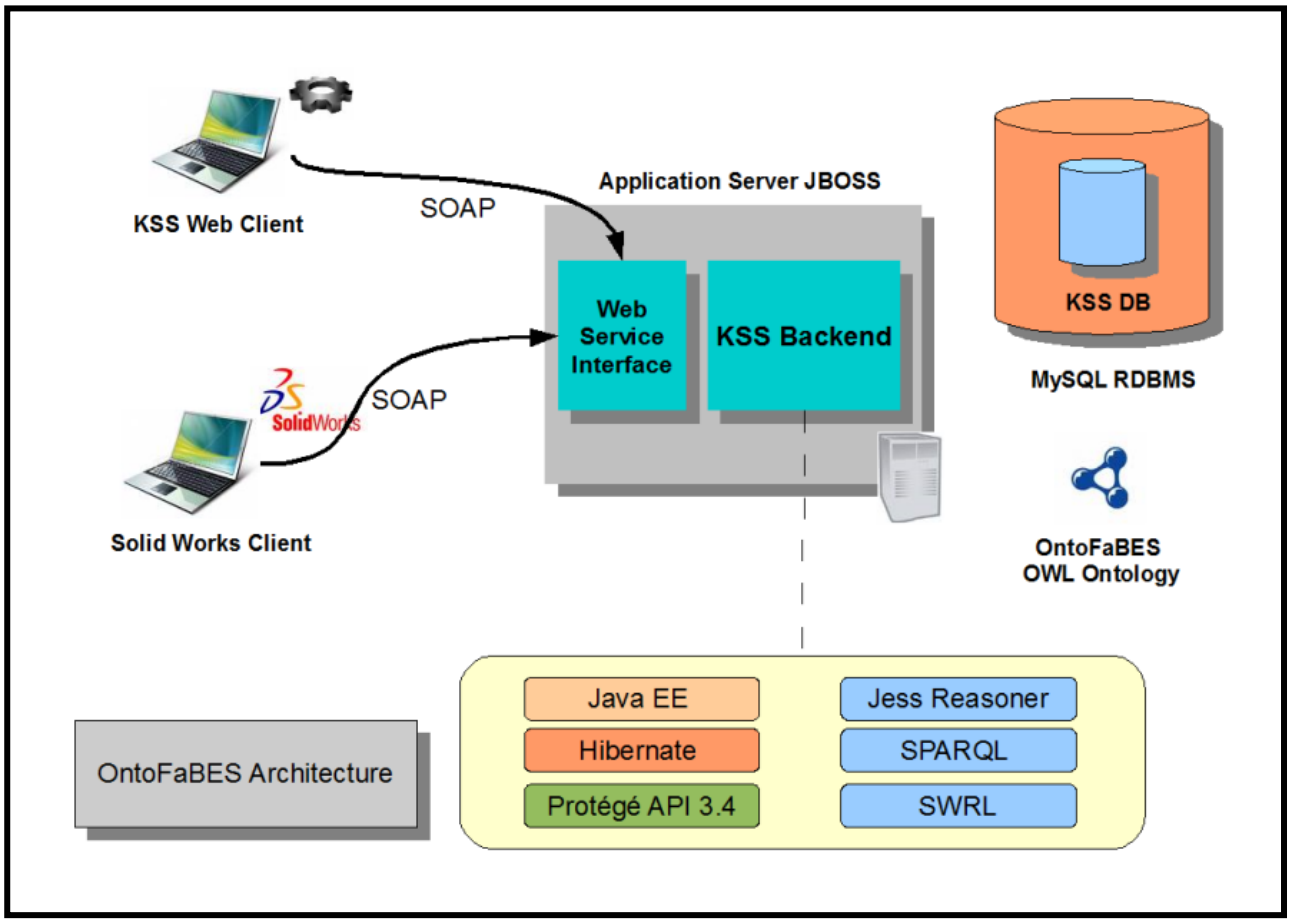


Figura 75: Implementación de la arquitectura OntoFaBES

El cliente *Solidworks* Client es un cliente auxiliar del sistema KSS. Su principal función es la conversión y persistencia en el sistema, de forma rápida y ágil, de formularios ICARE. En este caso, los usuarios de *Solidworks* cargarán un determinado proyecto de diseño y de forma semi-asistida se persistirán los formularios ICARE asociados a cada estructura del proyecto. Este cliente se trata de una macro de la herramienta de CAD *Solidworks*, desarrollada en VBa.

La parte servidora está compuesta por el núcleo de la aplicación, la funcionalidad y lógica de negocio del sistema. Podemos distinguir dos capas dentro de esta parte de la arquitectura:

- Interfaz Web Service
- Back end

La interfaz Web Service de la parte servidora posibilita la interoperabilidad y la integración entre distintos clientes, permitiendo de esta manera poder trabajar con distintas herramientas o sistemas CAD. La comunicación cliente – Web Service se logra utilizando el protocolo SOAP, protocolo empleado en los servicios web. La interfaz Web Service está desarrollada en Java, utilizando el API JAX-WS.

La capa de Back end implementa toda la lógica de negocio del sistema, como es la gestión de los formularios ICARE y la de la ontología OntoFaBES. Para la gestión de la ontología OntoFaBES se optado por utilizar el API de Protégé 3.4, junto al razonador JESS de esta forma tener soporte para ejecutar consultas SPARQL y ejecutar reglas SWRL.

El desarrollo de esta capa también ha sido realizado en Java, en este caso implementado bajo la plataforma Java EE. La capa de back end incorpora también una capa de persistencia, que se ha desarrollado utilizando el *framework* ORM Hibernate (capítulo 2.2.4.6), utilizando como base de datos MySQL RDBMS 6.0 (capítulo 2.2.3).

El cliente KSS Web Client y la parte servidora van desplegadas en un servidor de aplicaciones, en este caso JBOSS Application Server debido a su buena operatividad con la tecnología Hibernate, apoyo comunitario y una elevada cuota de mercado (Roth, 2006).

La interfaz para el usuario del sistema KSS 2.0 queda reflejada en la Figura 76, donde se puede observar la disposición de los diferentes apartados de la aplicación web y la información a complementar por parte de una entidad que se dispone como ejemplo.

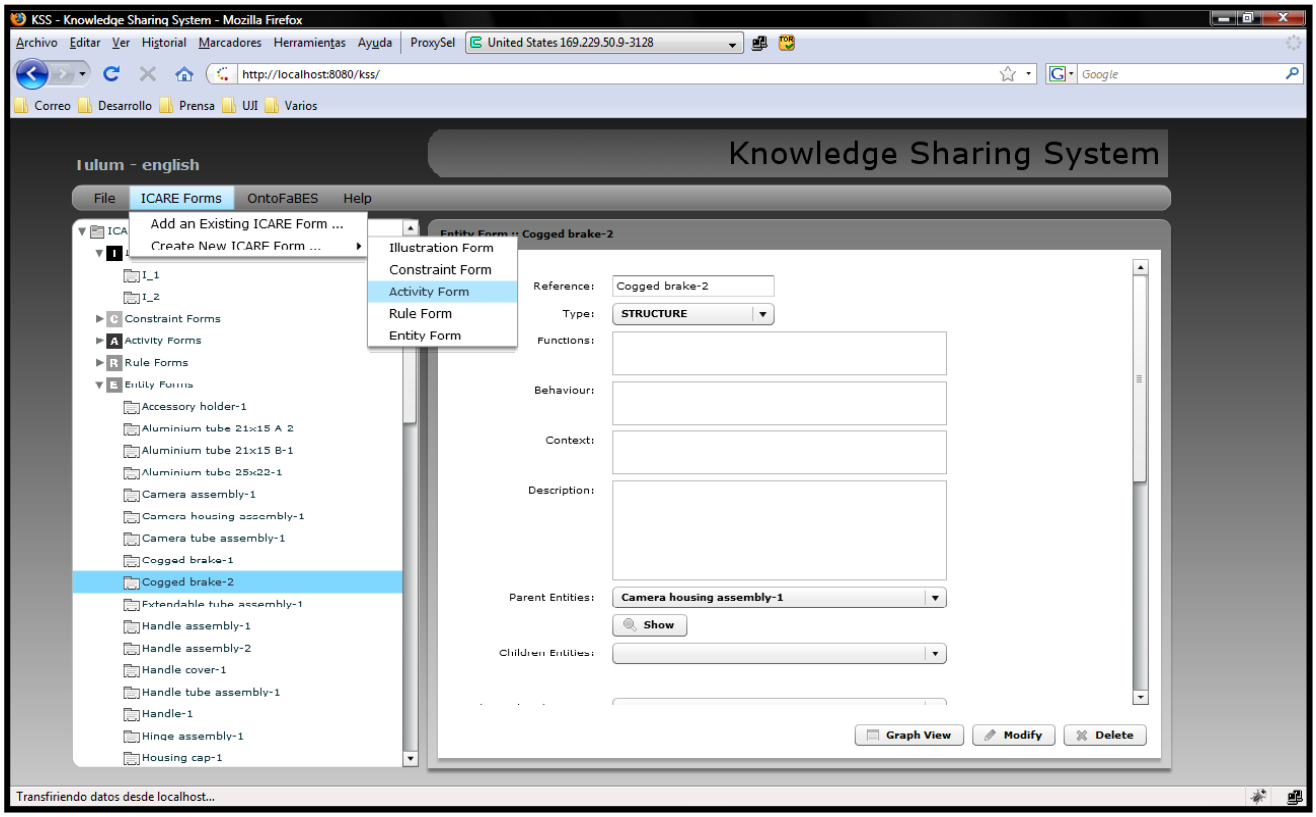


Figura 76: KSS 2.0

4.5 Descripción informática

4.5.1 Macro KSS –*Solidworks*

La macro de KSS hace uso del lenguaje programación VBA de *Solidworks*. Dicho lenguaje utiliza una API para lograr el reconocimiento de las partes del modelo del diseño creado. Para ello se basa en las características del modelo en la forma de 3D. La Figura 77 indica el procedimiento relativo a la integración de la preparación de la parte sólida, el reconocimiento de sus piezas y las conexiones establecidas entre ellas basado en el Modelo de objetos *Solidworks* API (Capítulo 2.2.1.3).

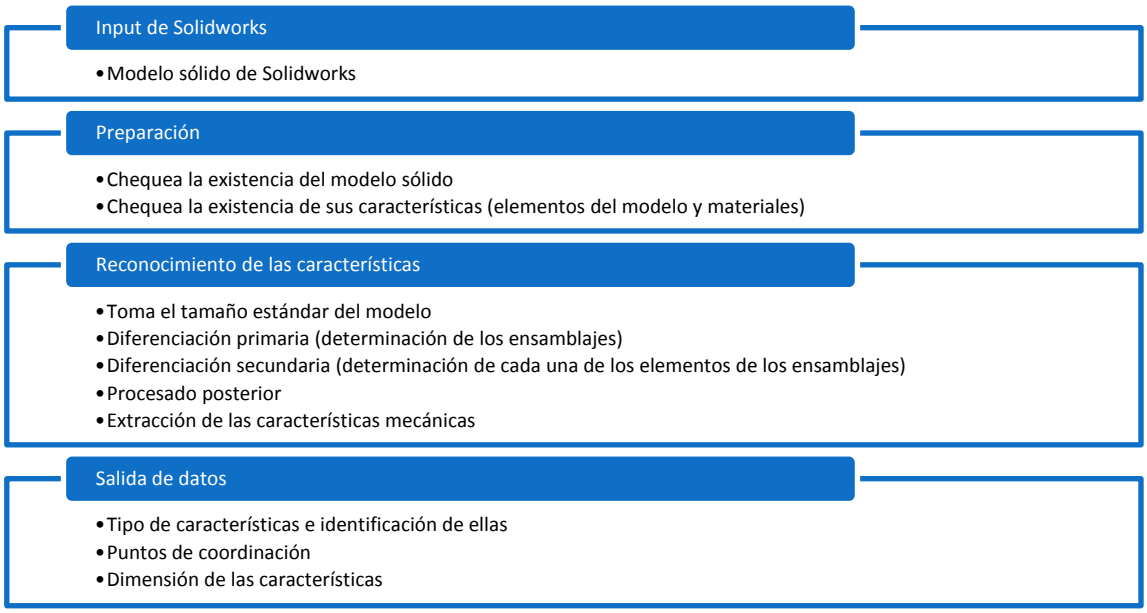


Figura 77: Procedimiento de reconocimiento del modelo de *Solidworks*

Con la información disponible por parte de *Solidworks* a través de la macro KSS (Figura 78) se generan dos apartados diferenciados:

- *Formulario de la interfaz del usuario*: A través de ese formulario se muestra la información obtenida (Figura 76).
- *Módulos de gestión de la información*: Permiten la disposición de la información relevante para el sistema KSS 2.0, la cual se estructura en 4 partes:
    - *Módulo de conexión*: Estructura los tipos de conexión y alineamiento que se pueden realizar entre las distintas piezas.
    - *Tabla de restricciones*: Indica las diferentes restricciones existentes cuando se unen dos o más piezas.
    - *Árbol de entidades*: Estructura la jerarquía de las piezas del diseño de un objeto en forma de árbol.
    - *Permiso de conexiones*: Establece una serie de reglas que permiten las conexiones adecuadas entre piezas.

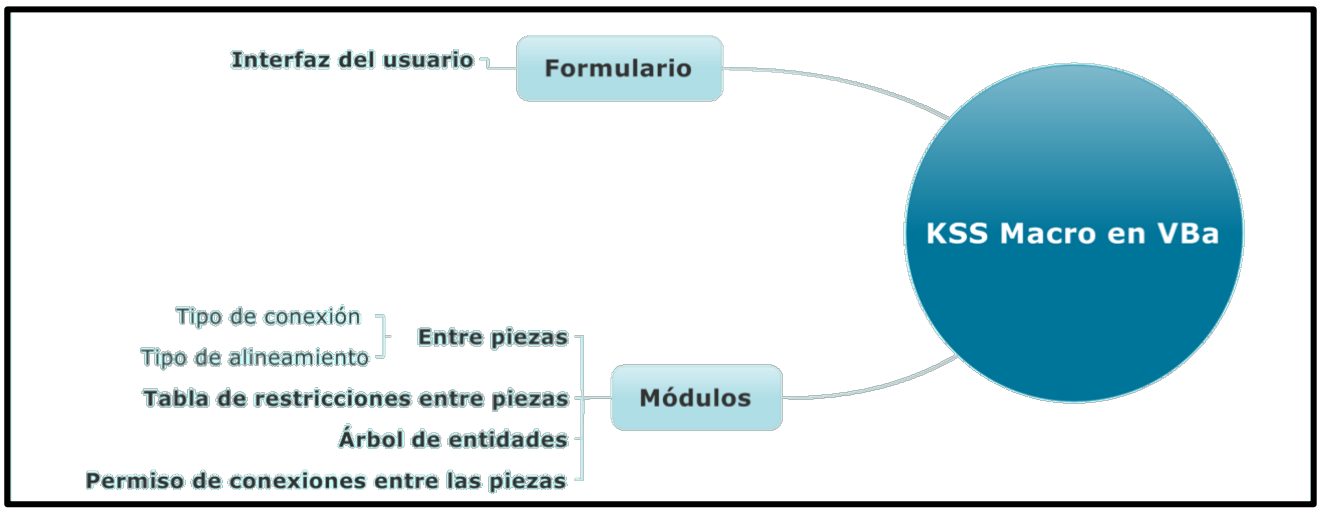


Figura 78: Esquema de la macro KSS

4.5.2 Programación en Java – KSS Java Client

En este apartado se explica cómo se ha programado en Java la aplicación KSS Web Application indicando las diferentes capas en que se ha desarrollado (Figura 79):

- **Data**: Esta capa se encarga de gestionar todos los datos necesarios para el adecuado funcionamiento del programa KSS 2.0 relativos tanto a la información recopilada de los formularios ICARE, los datos provenientes de la macro KSS-*Solidworks*, como los del proyecto de diseño y su programación informática. Esta información se detalla en el apartado 4.5.2.1.
- **Back-end Persistence**: Se encarga de persistir la información de todo el proyecto, es decir, de mantener una copia en el servidor de la información con la que se está trabajando. Por tanto, está formada por los mismos apartados de los que consta la capa Data más el apartado que se encarga de la realización de la propia persistencia de la información. Esta capa se explica en mayor profundidad en el apartado 4.5.2.2.
- **Util**: Se encarga de organizar la información según la estructura de MOKA y la ontología OntoFaBES. Es decir establece de enlace entre la información que se almacena en formularios ICARE según la metodología MOKA y la ontología OntoFaBES según lo explicado en el apartado 4.3.3. En el apartado 4.5.2.3 se explican los dos paquetes que conforman esta capa.
- **BI (*Business Intelligence*)**: Esta capa implementa la lógica de negocio de la aplicación KSS 2.0. Por un lado, implementa la programación relativa a la interfaz que se muestra al usuario en la KSS Web Application y, por otro lado, implementa los formularios ICARE dentro de la aplicación. Esta capa se explica con más detalle en el apartado 4.5.2.4.
- **Web Service**: Implementa el servicio web de la aplicación.

Las distintas capas están interrelacionados con diferentes dependencias debido a las necesidades de información del sistema. Tal como se observa en la Figura 80 a partir de la capa de datos y la de la persistencia de la información fundamentalmente se estructuran las conexiones del sistema.

A continuación se explicará en profundidad las características de cada una de las capas. En las imágenes que acompañan los siguientes apartados se indican las dependencias de las funciones respecto a las capas establecidas.

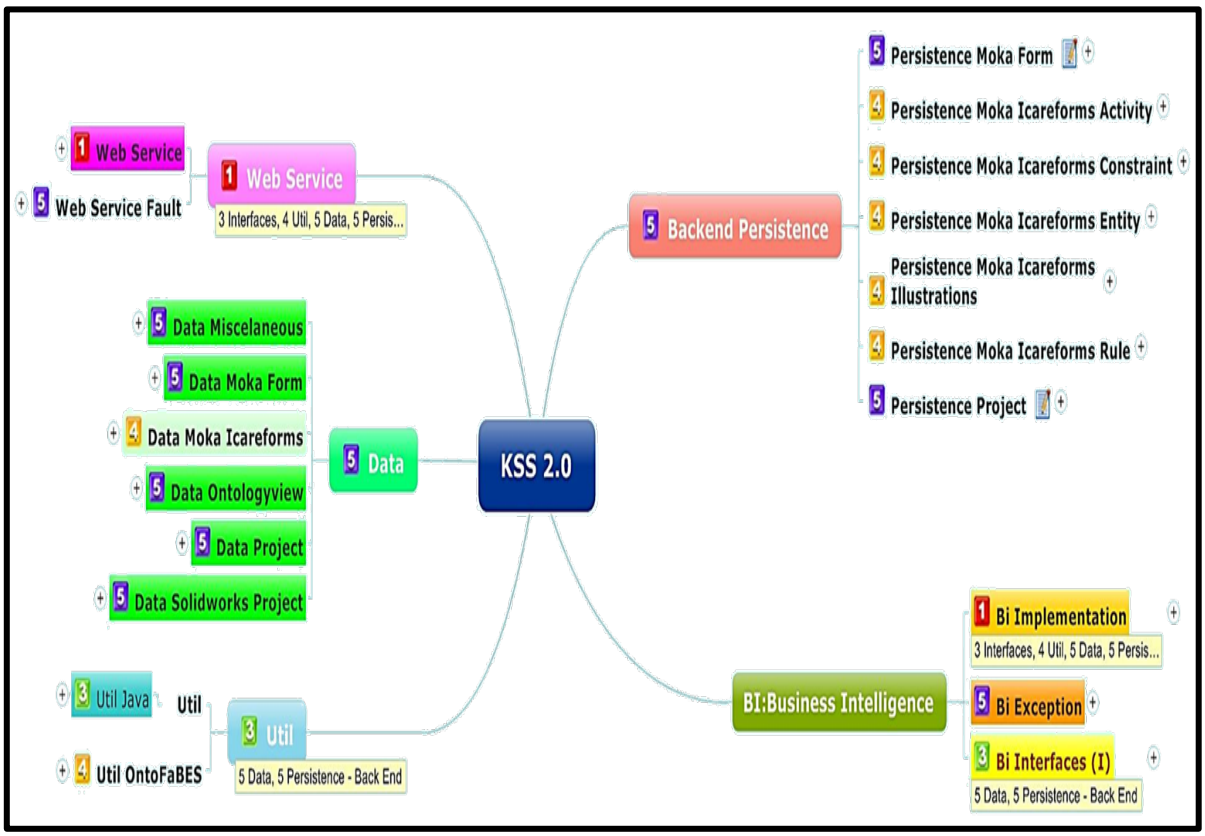


Figura 79: Organización del sistema KSS 2.0

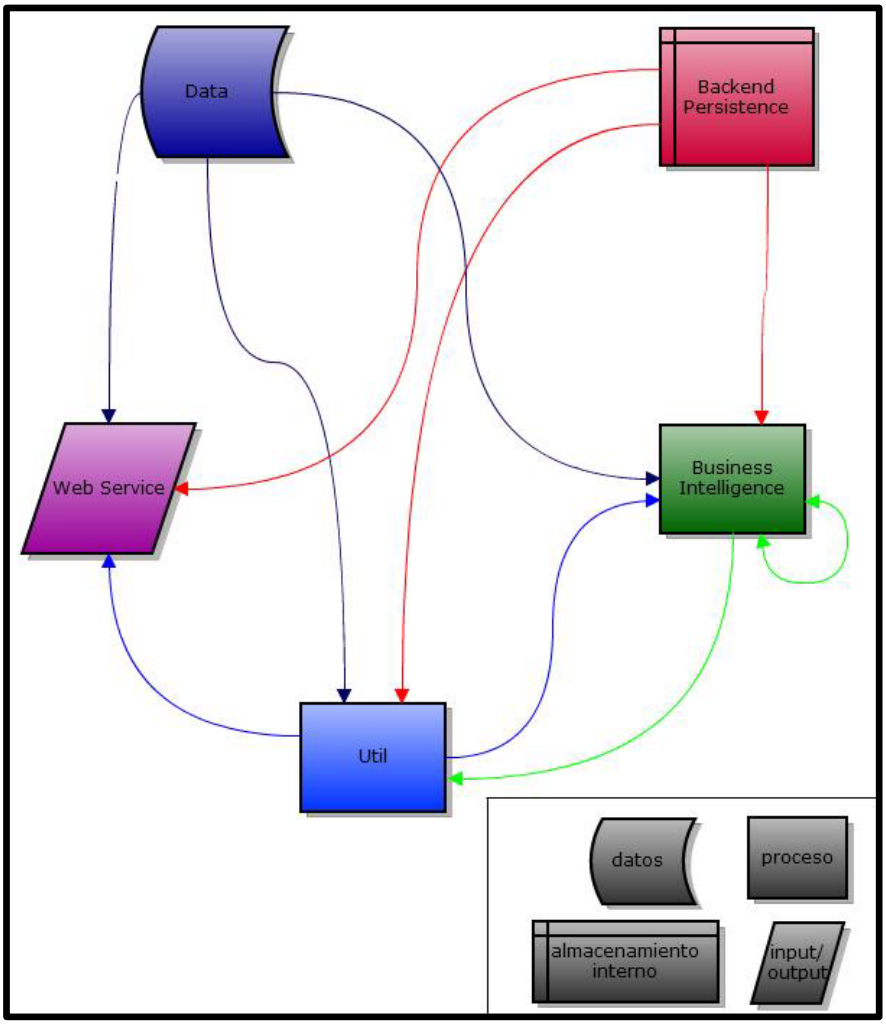


Figura 80: Conexiones del sistema KSS 2.0

4.5.2.1 Capa de Data

La capa de datos (Data) está conformada por 6 paquetes distintos (Figura 81):

- *Data Miscelaneous*: Este paquete se encarga de implementar las funciones necesarias para visualizar los grafos de los formularios ICARE (Apartado 2.3.3.2) en la aplicación web.
- *D. Moka Form*: Gestiona la implementación de la plantilla del formulario necesario para estructurar la información de los distintos formularios ICARE.
- *D. Moka Icareforms*: Gestiona los datos de los formularios ICARE.
- *D. Ontologyview*: Organiza la información sobre la visualización de la ontología en el sistema.
- *D. Project*: Gestiona el proyecto referido a los datos del sistema KSS 2.0.
- *D. Solidworks Project*: Gestiona el proyecto referido a los datos provenientes de *Solidworks* referidos a la macro KSS (Apartado 4.5.1).

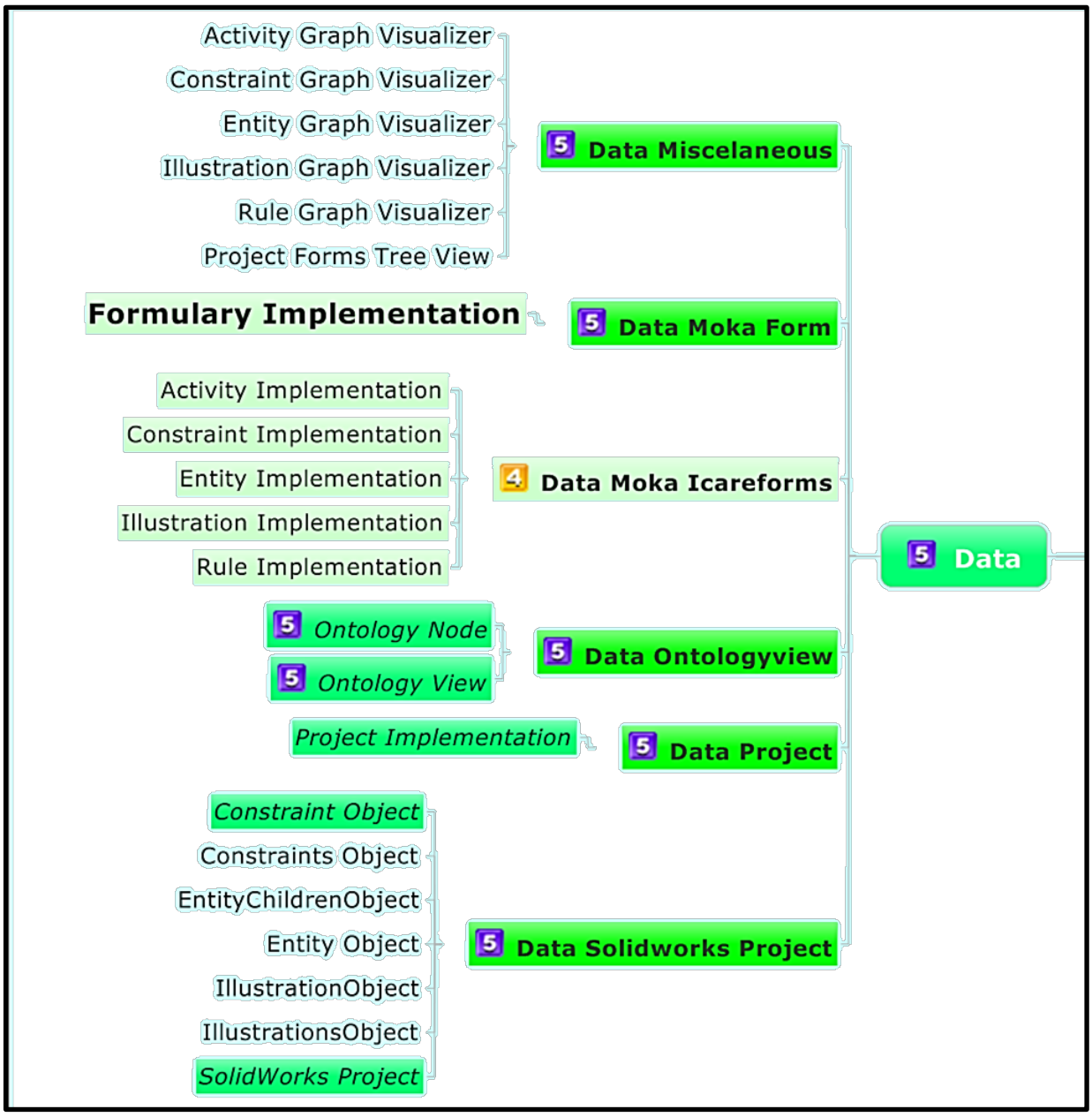


Figura 81: Módulo de datos[oo]

[oo] En la figura 5 están resaltadas las siguientes clases: *OntologyNode, Ontology View, Project Implementation, Constraint Object* ya que son importadas en otros paquetes. De la misma forma en los siguientes módulos están indicadas de la misma forma otras dependencias.

4.5.2.2 Capa de Back End– Persistencia

La capa de Back End está conformada por 7 paquetes distintos (Figura 82):

- *Persistence Moka Form*: Gestiona la plantilla del formulario necesario para estructurar la persistencia de información de los distintos formularios ICARE.
- *P. M. IcareformsActivity*: Gestiona la persistencia de datos del formulario *Activity*.
- *P. M. I. Constraint*: Gestiona la persistencia de datos del formulario *Constraint*.
- *P. M. I. Entity*: Gestiona la persistencia de datos del formulario *Entity*.
- *P. M. I. Illustrations*: Gestiona la persistencia de datos del formulario *Illustrations*.
- *P. M. I. Rule*: Gestiona la persistencia de datos del formulario *Rule*.
- *Persistence Project*: Gestiona el proyecto referido a la persistencia de datos del sistema.

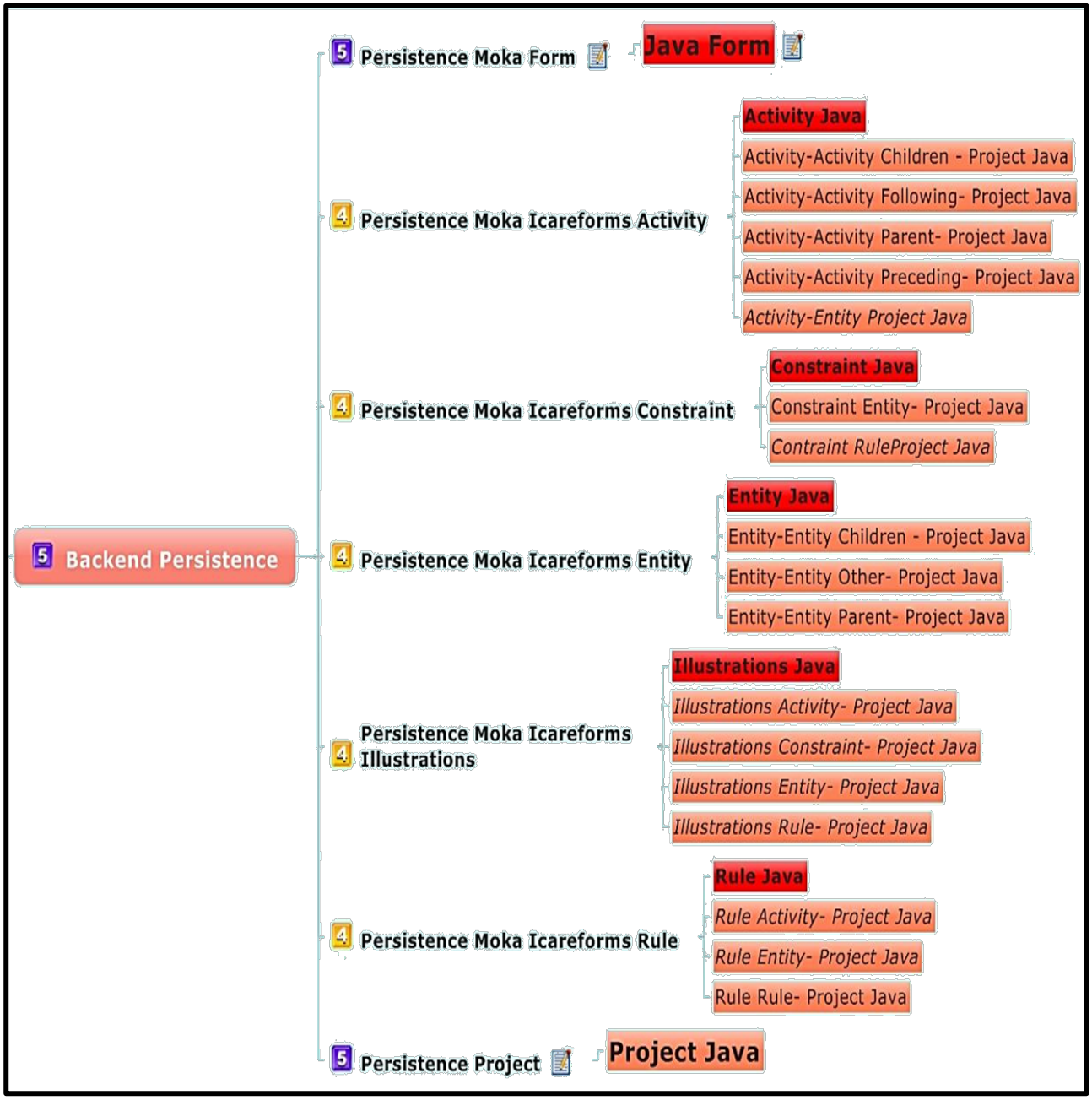


Figura 82: Módulo de persistencia

4.5.2.3 Capa *Util*

La capa de la parte *Util* está conformada por 2 paquetes distintos (Figura 83):

- *Util*: Establece las funciones necesarias para la distribución de la información entre los formularios ICARE en la aplicación web.
- *Util OntoFaBES*: Permite la ejecución de la ontología OntoFaBES en el sistema KSS 2.0. Por tanto sus dependencias están basadas en la API Protégé y en la información proveniente de *Solidworks*.

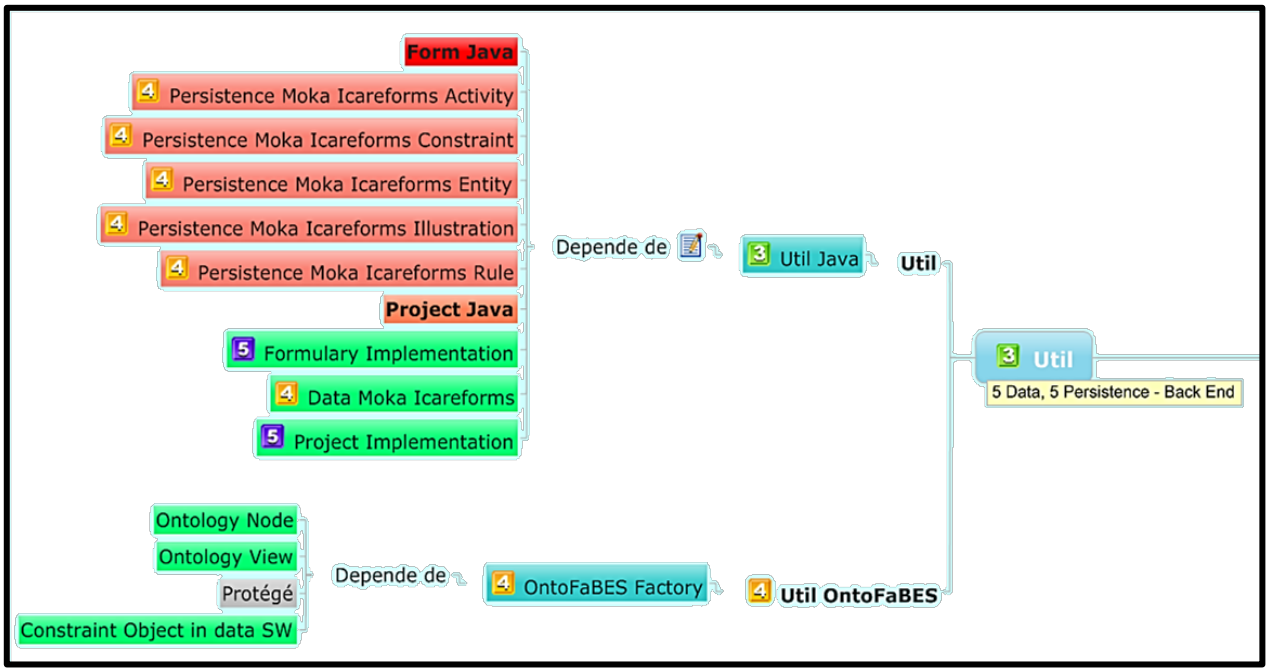


Figura 83: Módulo de parte útil

4.5.2.4 Capa de la lógica de negocio

La capa de la lógica de negocio está conformada por 10 paquetes distintos (Figura 84) organizados en 3 grupos:

- *Bi Implementation*: Este grupo de 3 paquetes se encargan de implementar la lógica de negocio del KSS 2.0 (Figura 85). Estos paquetes son los siguientes:
  - Bi Implementation Moka Form: Gestiona la implementación de la plantilla del formulario necesario para los distintos formulario ICARE.
  - B. I. Moka Icareforms: Gestiona la implementación de los formularios ICARE.
  - B. I. Project: Es el paquete que gestiona el proyecto de implementación.
- *Bi Exception*: Crea una función para el adecuado tratamiento de las excepciones[pp].
- *Bi Interfaces*: Está conformado por un grupo de 3 paquetes (Figura 88) que se encargan de gestión de los formularios en la interfaz de KSS 2.0.

El paquete de implementación de los formularios ICARE (B. I. Moka Icareforms) y la referida a la interfaz (*Bi Interfaces*), se explican en profundidad a continuación. Así, la parte de Implementación está conformada por 5 clases distintas (Figura 86 y Figura 87) dedicadas a cada uno de los formularios ICARE y la parte de interfaz está conformada por 3 paquetes distintos (Figura 88):

- *Bi Interfaces Moka Form*: Gestiona la plantilla de la interfaz necesaria para los distintos formularios ICARE.
- *Bi Interfaces Moka Icareforms*: Gestiona la interfaz de los formularios ICARE.
- *Bi Interfaces Project*: Es el paquete que gestiona el proyecto de la interfaz.

[pp]Una excepción se refiere un problema que ocurre con poca frecuencia durante la ejecución de un programa, generalmente cuando existe algún dato o instrucción que no se coordina con el funcionamiento del programa por lo que se produce un error. El manejo de excepciones permite al usuario crear aplicaciones tolerantes a fallas y robustos (resistentes a errores) para controlar estas excepciones y que pueda seguir ejecutando el programa sin verse afectado por el problema.

4.5.2.5 Capa Web Service

- La capa de Web Service está conformada por 2 paquetes distintos (Figura 89): *Web Service* y *Web Service fault* que constituyen el apartado del servidor web del KSS 2.0.

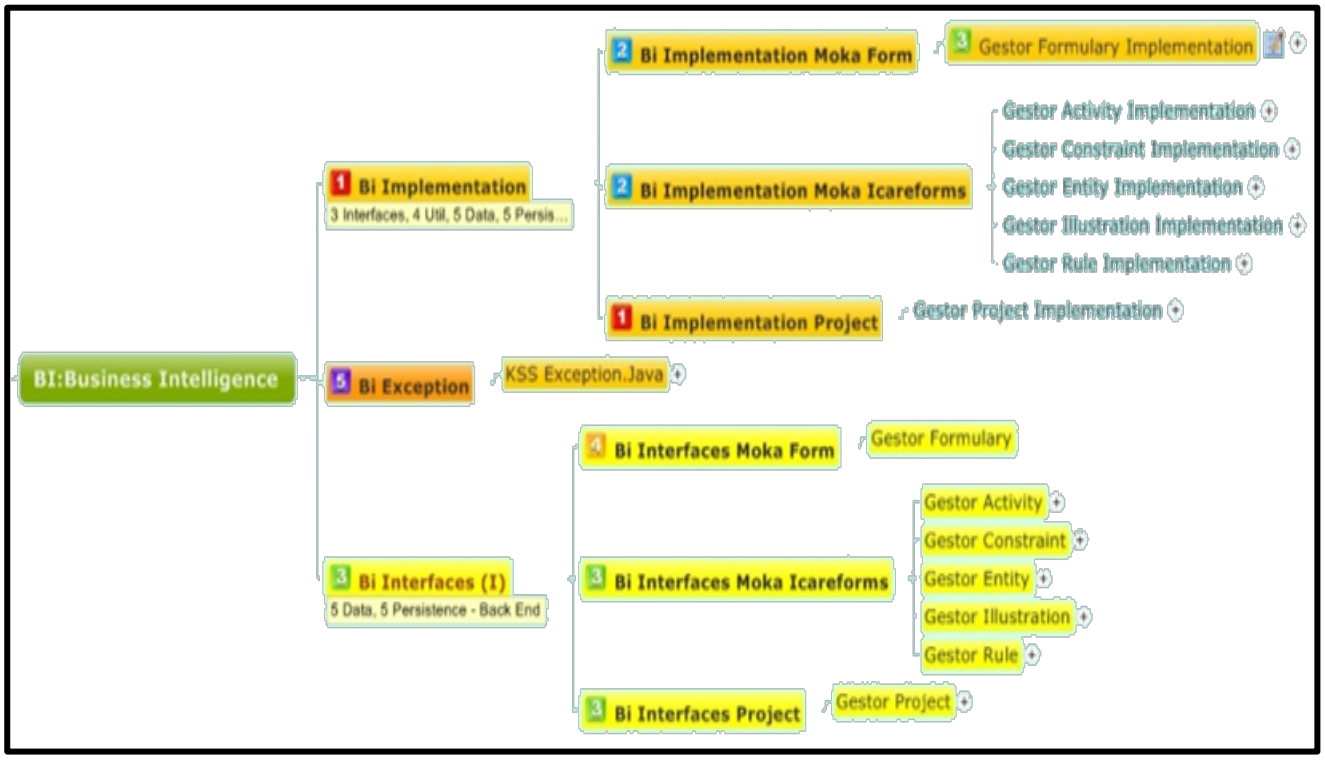


Figura 84: Módulo de lógica de negocio

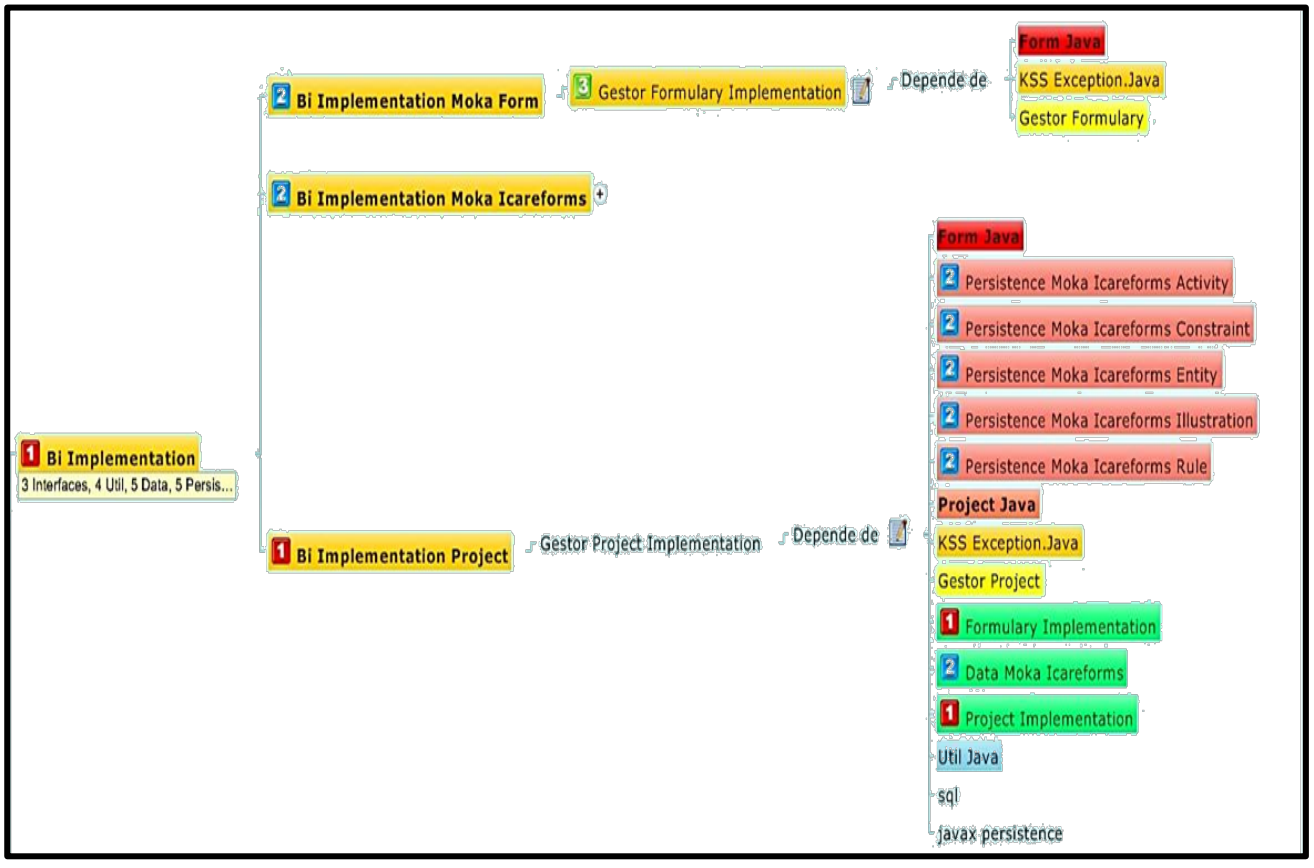


Figura 85: Módulo de lógica de negocio. Implementación

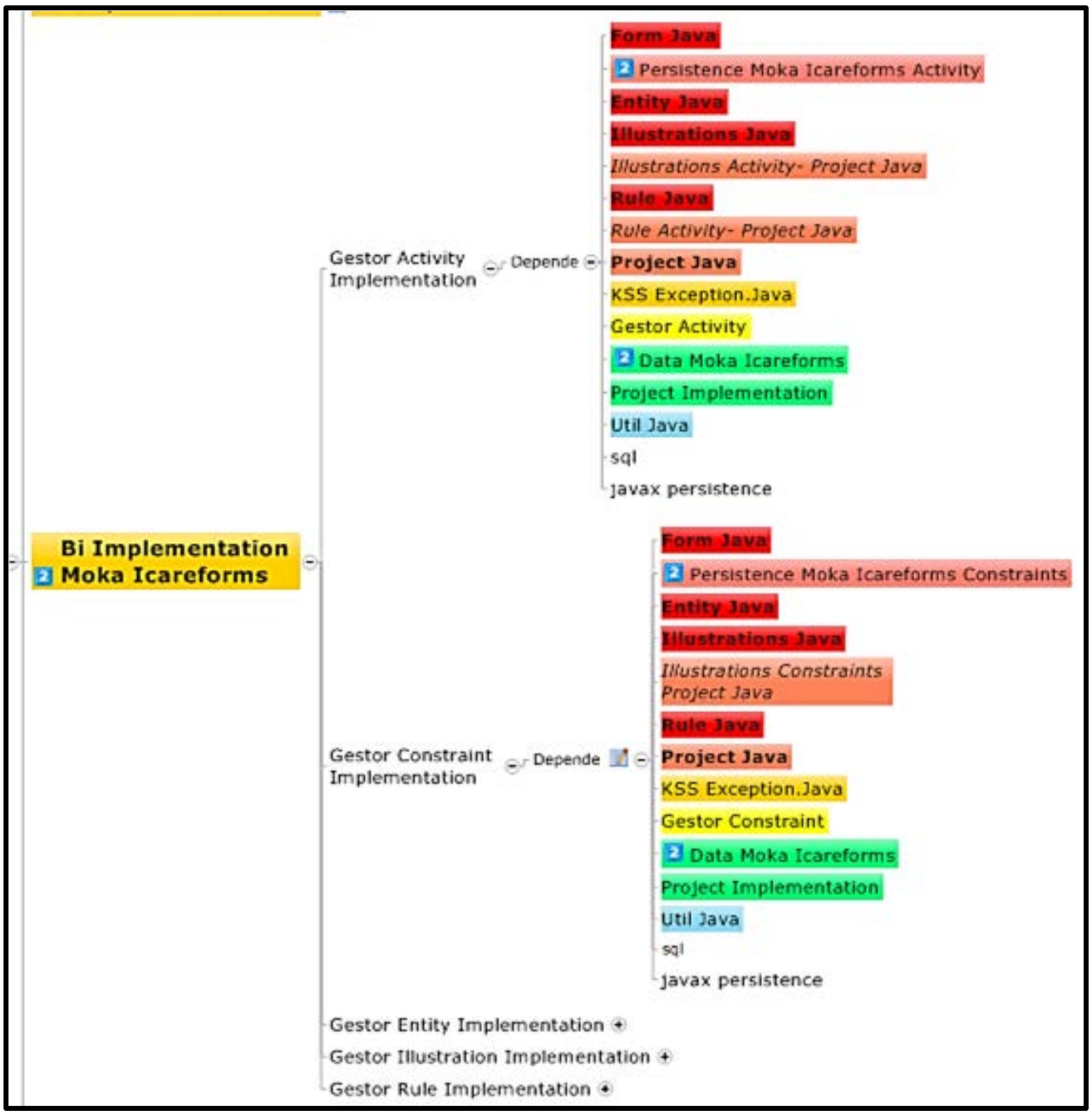


Figura 86: Módulo de lógica de negocio. Implementación de los formularios ICARE (Parte 1)

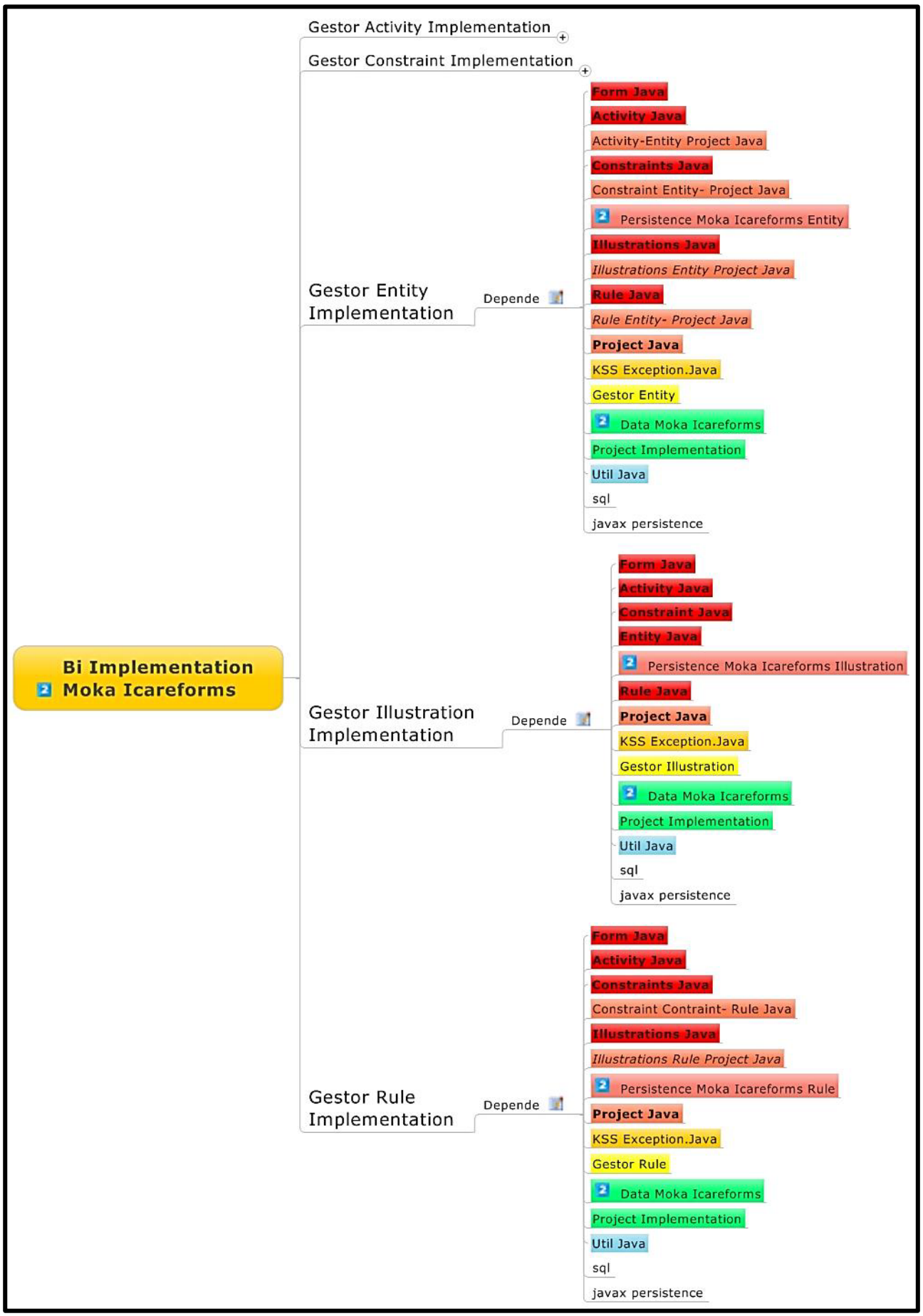


Figura 87: Módulo de lógica de negocio. Implementación de los formularios ICARE (Parte 2)

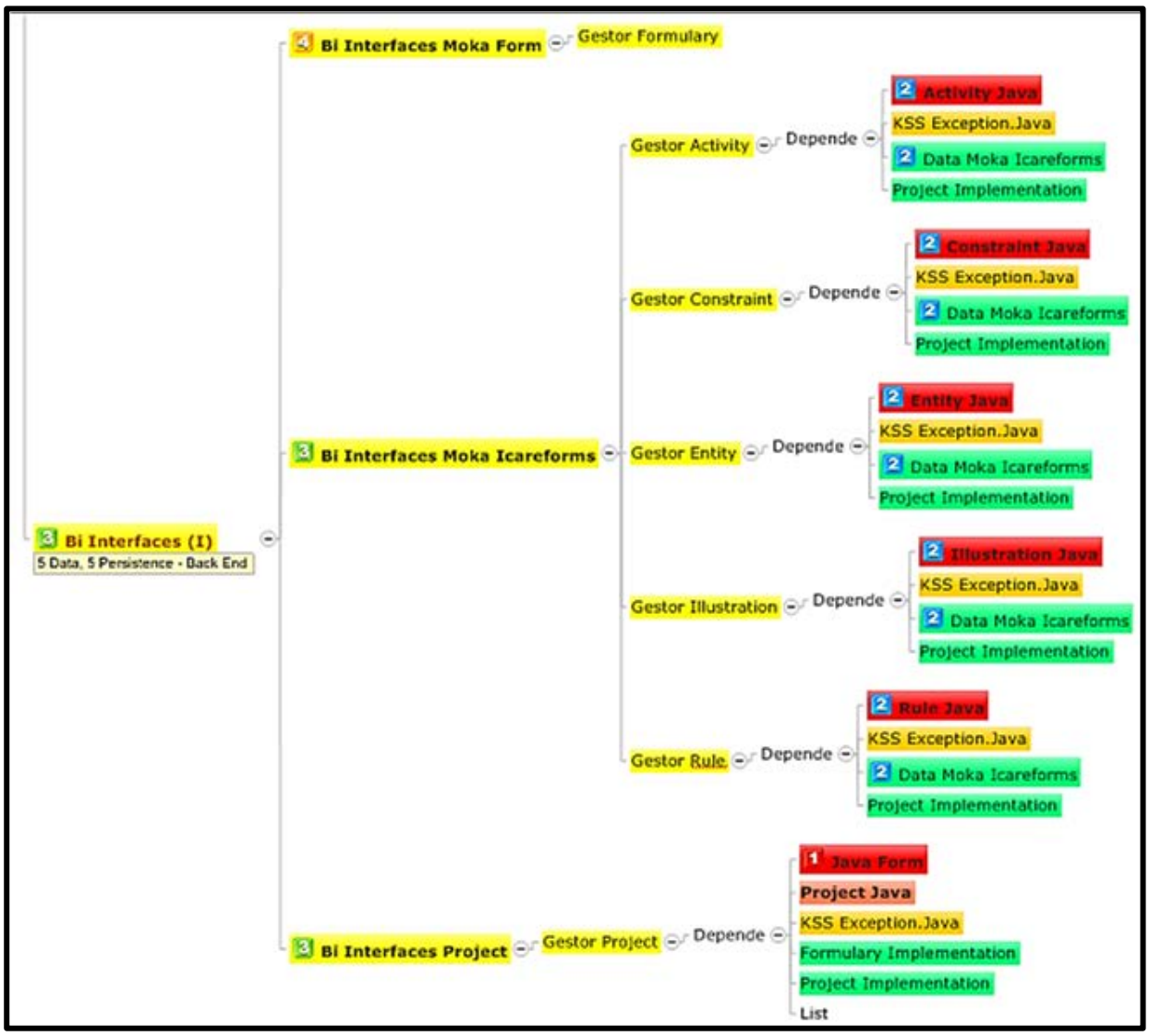


Figura 88: Módulo de lógica de negocio. Interfaz

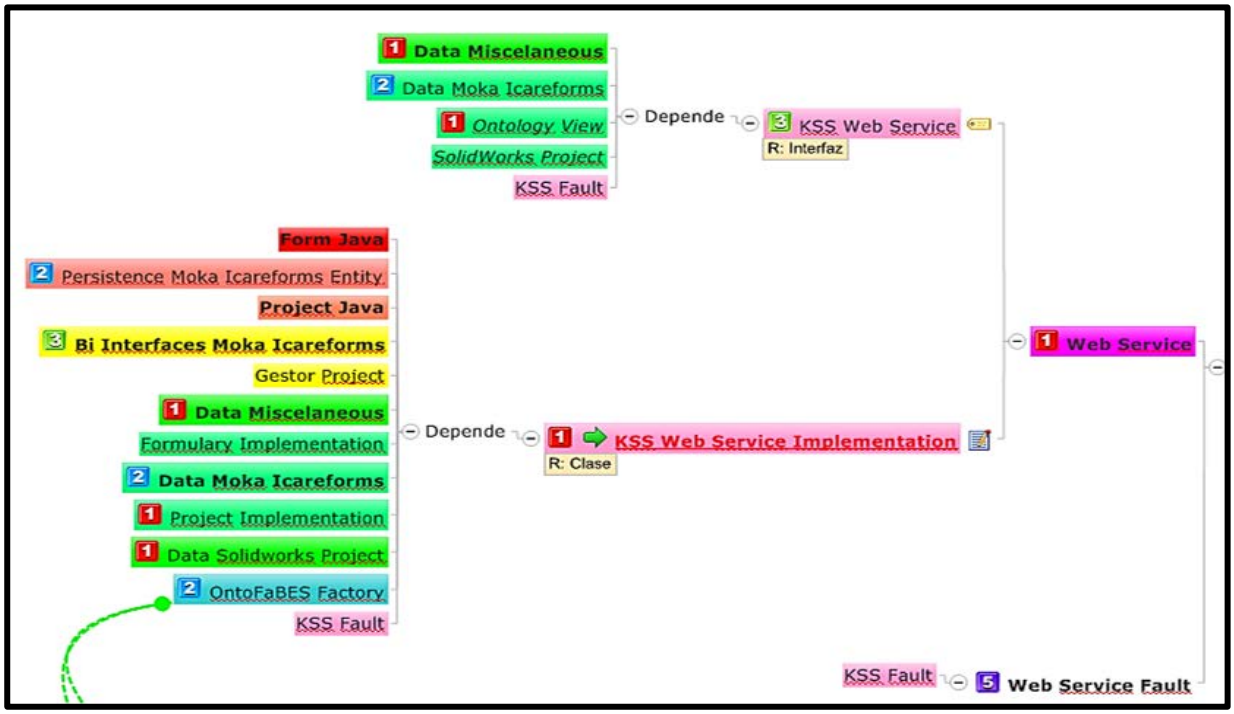


Figura 89: Módulo del Servicio Web

4.6 Discusión y conclusiones

Para la discusión del KSS 2.0 se va a seguir el razonamiento planteado por Wang (2004) en la que indica 11 puntos necesarios en una herramienta de gestión del conocimiento.

1. KSS 2.0 constituye un repositorio de comprensión común al proporcionar una interfaz disponible en internet vía página web logrando generar un marco estructural general para todos los usuarios de la herramienta. Además el tratamiento de información por parte de la ontología se ejecuta en paralelo, permitiendo conocer todo lo que ocurre en tiempo real. Eso facilita que cualquier usuario pueda reutilizarlo y compartirlo. El ingeniero ontológico sólo debe realizar alguna modificación si existe algún problema interno en la ontología.
2. KSS 2.0 al adaptarse a la información disponible en el programa CAD y utilizar una ontología de dominio (en este caso de diseño funcional) automatizando el proceso y dejando que se introduzca aquel conocimiento nuevo de forma libre en los campos correspondientes, permite que el usuario pueda crear el conocimiento en su propio modo usando sus propios términos y conceptos. Únicamente debe conocer unas nociones simples respecto a cómo rellenar los formularios ICARE. Posteriormente OntoFaBES con el conocimiento recopilado realizará una clasificación e inferencia de éste.
3. KSS 2.0 al basarse en OntoFaBES, ontología desarrollada en lenguaje OWL, directamente genera el mapa de conocimiento para asistir a sus usuarios (Figura 66). Eso permite que puedan definir y estructurar sus propios conceptos de forma global yendo de conceptos generales a concretos.
4. Al utilizar la base conceptual de MOKA y las herramientas de clasificación de la ontología se permite que el conocimiento se recopile segmentado y pueda organizarse *ad hoc*, y de forma inmediata y espontánea. Esto se puede comprobar en el Capítulo 5.4 donde se aplica el KSS 2.0 al caso de un ensamblaje de una cámara submarina.
5. La propia estructura de KSS 2.0 está establecida para que el diseñador del producto pueda formalizar la información del modelo en el programa CAD cuando considere necesario tal como se ha indicado en el capítulo 4.2.1. Además tal como se explica en el apartado 4.2, el rol del miembro de proyecto sólo puede ver ya la información formalizada a través de la aplicación web sin poder interactuar con la ontología de dominio, herramienta editable sólo para el ingeniero de conocimiento mediante *Protégé*. Esta estructura definida en diferentes niveles permite la utilización de segmentos adecuados a la experiencia por parte del usuario.
6. El modelo MOKA permite un proceso de revisión que el sistema KSS 2.0 adapta al mostrar el estado del formulario, quien es su autor, versión y cuándo se ha modificado por última vez (Figura 63).
7. La consistencia del sistema de conocimiento se mantiene a través de la utilización de un razonador aplicado a OntoFaBES, el cual es gestionado por el ingeniero de conocimiento, capaz de notificar a los usuarios cuando surjan los conflictos, determinados por análisis de consistencia ejecutados mediante la herramienta *Protégé*.
8. El objetivo de KSS 2.0 al utilizar OntoFaBES, basado en un lenguaje libre como es OWL y disponer de su aplicación en un entorno web es que sea compatible con el mayor rango posible de sistemas existentes. Para este caso se ha utilizado el programa *Solidworks* como programa que sirve de fuente del modelado del producto, pero se plantea para un futuro exportar la macro para poderse utilizar en otros programas CAE.
9. Al disponer del conocimiento completamente estructurado gracias a la combinación de MOKA y OntoFaBES, su mantenimiento se automatiza en gran medida simplificando su utilización. Únicamente, la intervención puntual por parte del ingeniero de conocimiento es necesaria para que el funcionamiento de la ontología sea el correcto.
10. La propiedad de la información pertenece al equipo de diseñadores quienes trabajan en el modelo del producto y en quienes posteriormente evalúan su diseño a través de la plataforma web. El ingeniero de conocimiento únicamente modificaría el sistema si se halla algún problema y se dedica a su mantenimiento nunca a la generación de conocimiento como tal.

11. El sistema KSS 2.0 al estar basado en una ontología de dominio como es OntoFaBES y en una metodología de adquisición del conocimiento como es MOKA, se permite que fácilmente se puedan realizar cambios oportunos para cada caso.

Finalmente, se debe indicar que KSS 2.0 constituye una herramienta compleja en su desarrollo con el objetivo de facilitar la explicitación del conocimiento en el proceso de rediseño de un producto. Como tal, debido a su estructura modular, sería posible su mejora y adaptación a nuevas tecnologías sin prejuicio de la utilización de dicha herramienta por parte de sus usuarios:

- La macro SW-KSS se podría enlazar a otros programas CAE o versiones siguientes con una pequeña modificación.
- El programa de mantenimiento de OntoFaBES, podría ser sustituido por cualquier otro para evaluar la consistencia del contenido de la ontología.
- Y la información proporcionada al sistema se puede ampliar tanto gracias a ontologías auxiliares que se podrían aunar al conocimiento de OntoFaBES como bases de datos que se podrían clasificar y explicitar para su utilización.

Capítulo 5. Modelado del TULUM®

En este capítulo se evalúa la consistencia de KSS 2.0 y por tanto de OntoFaBES. Para ello, se considera el ejemplo real de una cámara submarina que recibe el nombre comercial de TULUM®, instrumento que sirve para inspeccionar el casco de veleros sin necesidad de sacarlo del agua (Figura 90).

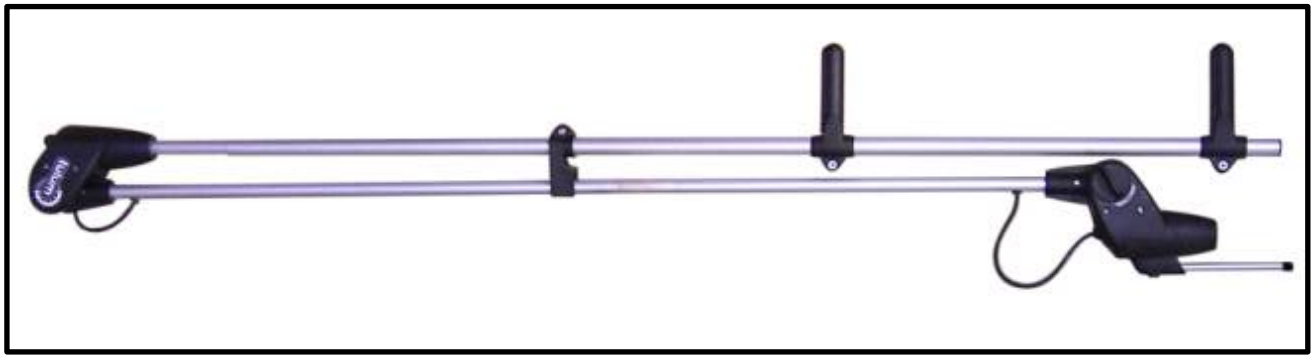


Figura 90: TULUM®

Para poder entender la aplicación informática primero se va a determinar el análisis de funciones y comportamientos haciendo uso del marco FaBES y del modelo A-QB. Posteriormente, se describe el proceso de obtención de información del diseño del TULUM® utilizando el programa de CAD *Solidworks*® 2009, su posterior formalización a través de la macro KSS-*Solidworks*, edición en la aplicación web KSS 2.0 y conversión a la ontología OntoFaBES.

Finalmente, se muestran los resultados obtenidos al aplicar las reglas de inferencia definidas en OntoFaBES.

5.1 TULUM®: Cámara submarina

TULUM®[qq] es una herramienta formada por una pequeña cámara submarina fijada al extremo de una larga pértiga ajustable en cuyo extremo superior está ubicado un monitor antirreflectante que permite visionar la señal de vídeo captada por la cámara (Figura 90) (TULUM®, 2008).

La pértiga está formada por dos tubos de aluminio tratado, unidos por un codo que ayuda a manejar la cámara en cualquier ángulo para llegar incluso a la parte más interior de la quilla y conseguir inspeccionar el estado de pernos, pasadores o cualquier zona del casco.

El brazo extensible y articulado le permite realizar inspecciones de objetos o instalaciones sumergidas y poco accesibles, y de forma sencilla. Tiene regulador de altura y permite definir la posición de trabajo adaptándose a la curvatura del casco.

La prolongación máxima del brazo es de unos 6 metros lo que permite un rango de profundidad variable. Esto hace que TULUM® pueda trabajar hasta con barcos de considerable calado, siendo un instrumento adecuado para talleres de reparación o incluso para el personal de puertos que necesiten verificar el estado de pantalanes, fijaciones a muertos, o cualquier instalación submarina susceptible de ser inspeccionada.

El peso total de TULUM® es de unos 5 Kilos y gracias a sus dos brazos de sujeción se maneja y controla su posición. La articulación permite ajustarlo y adaptar la posición de trabajo a la curvatura de cualquier casco y embarcación. En la cabeza donde está situada la cámara es posible adaptar varios utensilios para realizar incluso trabajos de limpieza y mantenimiento.

En el anexo 4 se puede encontrar el listado de todas las piezas del TULUM® y los ensamblajes de éstas acompañadas de una imagen ilustrativa.

[qq] El TULUM® es un producto industrial real que fue comercializado por una empresa. No se indica más información al respecto para preservar la privacidad de la propiedad intelectual del producto. Dicha información se encuentra en inglés. Si surge alguna duda sobre el significado de alguna pieza, se recomienda al lector dirigirse al anexo 4.

5.2 Aplicación de FaBES al TULUM®

A continuación se realiza la aplicación del FaBES al TULUM®. De manera análoga a los ejemplos del Aptdo. 3.1.9, se muestran las acciones, funciones, comportamientos, estructuras y entornos.

5.2.1 Acciones

La acción fundamental es la inspección de cascos de barcas sin sacarlas del agua. Según el modelo A-QB aunado al esquema FaBES, las acciones del TULUM® se muestran en la Figura 91.

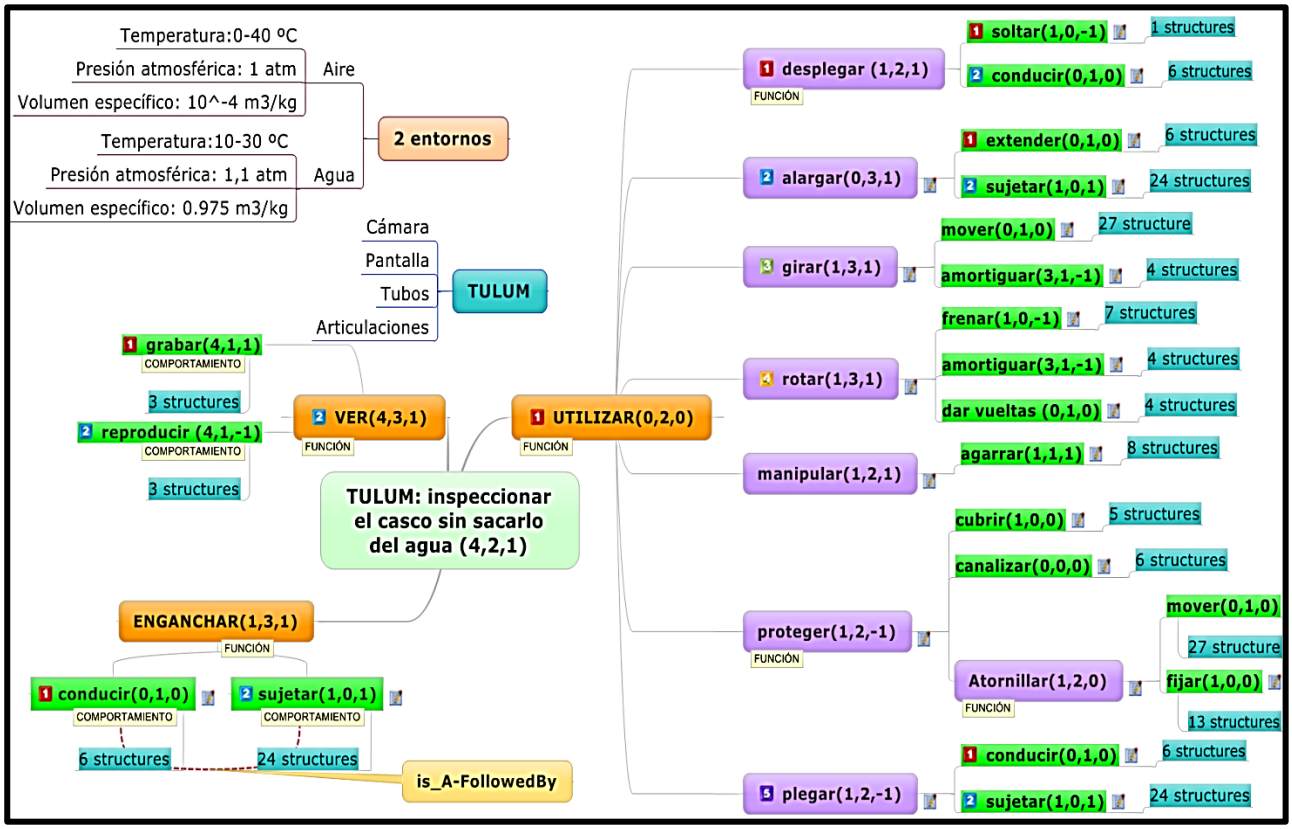


Figura 91: Esquema FaBES del TULUM®.

5.2.2 Funciones

La función objetivo del TULUM® es como se ha citado la inspección del casco de barcas. A la hora de definir las restantes funciones se utiliza el marco FaBES y el esquema A-QB tal como se ha explicado en el Aptdo. 3.1. Dicha función se puede dividir en 3 subfunciones: enganchar, ver y utilizar, tal como se muestra en la Figura 91. Igualmente se observa la jerarquía de funciones donde se debe mencionar que la función *UTILIZAR* lleva el número 1 con respecto a la función *VER* ya que para ejecutar la segunda, primero se debe ejecutar la primera y de forma independiente existe la función *ENGANCHAR*. De la misma manera, las subfunciones de *UTILIZAR* se organizan de manera análoga.

En la citada figura también se puede observar la explicación de los puntos elegidos según el modelo A-QB. En el caso de la función objetivo el detalle es el siguiente:

Endurants: Una persona y el TULUM®. En el primer caso, la persona es el APO de la función y el TULUM® es el NAPO (Figura 90).

Cualidad física (X): *Signal* (4)*;* ya que el hecho de inspeccionar corresponde a la recepción de una señal, concretamente una salida de señales coordinadas interpretadas en imágenes a través de un visor.

***Perdurant* (Y):** Accomplishment (2); al ser una acción dependiente del tiempo cuyo cumplimiento ocupa una fracción del tiempo.

Cualidad temporal (Z): *Terminal SoA* (1); ya que las propiedades deseadas, el visionado de la imagen, se obtienen al final del proceso.

Siguiendo este procedimiento, se obtiene la serie de funciones que describen al TULUM® en forma de tabla (Tabla 31) donde se muestra las coordenadas según el modelo A-QB y el sujeto pasivo de la acción tomada (*complemento*).

Tabla 31: Funciones del TULUM® según FaBES

Definición de la función	Cualidad física (x)	*Perdurant* (y)	Cualidad Temporal (z)	Complemento
Inspeccionar	Signal (4)	Accomplishment (2)	Terminal SoA (1)	Casco
Enganchar	Topological Connectedness (1)	Achievement (3)	Terminal SoA (1)	Elementos externos
Utilizar	Spatial Location (0)	Accomplishment (2)	Immutable SoA (0)	Estructura del TULUM
Ver	Signal (4)	Achievement (3)	Terminal SoA (1)	Casco
Desplegar	Topological Connectedness (1)	Accomplishment (2)	Terminal SoA (1)	Estructura del TULUM
Alargar	Spatial Location (0)	Achievement (3)	Terminal SoA (1)	Estructura del TULUM
Girar	Topological Connectedness (1)	Achievement (3)	Terminal SoA (1)	Estructura del TULUM
Rotar	Topological Connectedness (1)	Achievement (3)	Terminal SoA (1)	Estructura del TULUM
Manipular	Topological Connectedness (1)	Accomplishment (2)	Terminal SoA (1)	Estructura del TULUM
Proteger	Topological Connectedness (1)	Accomplishment (2)	Initial SoA (-1)	Estructura del TULUM
Plegar	Topological Connectedness (1)	Accomplishment (2)	Initial SoA (-1)	Estructura del TULUM
Atornillar [rr]	Topological Connectedness (1)	Accomplishment (2)	Immutable SoA (0)	Estructura del TULUM

En la Tabla 31 se puede ver que todos los *perdurant*s son Accomplishment o Achievement manteniendo la regla del Aptdo. 3.1.3 donde se indicaba que una función siempre será una acción de tipo eventual. Igualmente cabe destacar que hay una serie de funciones existentes en este diseño comunes a otros en los que hayan uniones mecánicas, como son las relativas a **atornillar**, **proteger** o **manipular**.

5.2.3 Comportamientos

En este caso, los comportamientos derivados de las funciones son diversos destacando que dependiendo de la complejidad de la función hay un mayor número de comportamientos asociados (Figura 91). En el caso del comportamiento referido al de reproducir las imágenes del casco de la barca que se esté

[rr] Esta función es la única que su *endurant* no es el TULUM sino los tornillos y tuercas de éste. El motivo es porque dicha función no define el uso del TULUM sino que es común a cualquier diseño.

inspeccionando según el modelo A-QB el valor es (4, 1, -1), es decir, el visor reproduce las imágenes de una sección del casco de la barca:

- ***Endurant*s**: Visor de imágenes y sección del casco. Ambas estructuras son NAPO.
- **Cualidad física (X)**: *Signal* (4)*;* ya que el hecho de reproducir corresponde a la recepción de una señal, concretamente una salida de señales coordinadas interpretadas.
- ***Perdurant* (Y):** Process (1); al ser una acción independiente del tiempo cuya acción no tiene un punto de finalización natural, pues depende del usuario que indique en qué momento se inicia tanto la reproducción de la señal como su finalización.
- **Cualidad temporal (Z)**: *Initial SoA* (-1); ya que la señal se obtiene al inicio del proceso.

Siguiendo este procedimiento, se obtiene la serie de comportamientos que describen al TULUM® en forma de tabla (Tabla 32) donde se muestra las coordenadas según el modelo A-QB, el *endurant* de la acción y el sujeto pasivo de la acción tomada (*complemento*). En la columna de *endurant*, se ha simplificado deliberadamente la definición de las estructuras implicadas para poder comprender más fácilmente el sentido de los citados comportamientos.

Tabla 32: Comportamientos del TULUM® según FaBES

Definición de la función	*Endurant*	Cualidad física (x)	*Perdurant* (y)	Cualidad Temporal (z)	Complemento
Reproducir	Cámara	Signal (4)	Process (1)	Initial SoA (-1)	Imágenes del casco
Grabar	Cámara	Signal (4)	Process (1)	Terminal SoA (1)	Imágenes del casco
Conducir	Ensamblajes	Spatial Location (0)	Process (1)	Immutable SoA (0)	La inclinación del TULUM
Sujetar	Bisagras	Topological Connectedness (1)	State (0)	Terminal SoA (1)	TULUM
Soltar	Clip de junta	Topological Connectedness (1)	State (0)	Initial SoA (-1)	Tubo del TULUM
Extender	Tubos	Spatial Location (0)	Process (1)	Immutable SoA (0)	TULUM
Mover	Ensamblaje bisagra	Spatial Location (0)	Process (1)	Immutable SoA (0)	La inclinación del TULUM
Amortiguar	Muelles	Energy (3)	Process (1)	Initial SoA (-1)	Ensamblajes
Frenar	Freno de dientes	Topological Connectedness (1)	State (0)	Initial SoA (-1)	Ensamblajes
Dar vueltas	Tornillos	Spatial Location (0)	Process (1)	Immutable SoA (0)	Bisagras
Agarrar	Mango	Topological Connectedness (1)	Process (1)	Terminal SoA (1)	TULUM
Cubrir	Cubiertas	Topological Connectedness (1)	State (0)	Immutable SoA (0)	Ensamblajes
Canalizar	Tubos	Spatial Location (0)	State (0)	Immutable SoA (0)	Cables
Fijar	Tuercas	Topological Connectedness (1)	State (0)	Immutable SoA (0)	Tornillos y estructuras

En la Tabla 32 se puede ver que todos los *perdurant*s son *Process* o *State* manteniendo la regla del Aptdo. 3.1.3 donde se indicaba que un comportamiento siempre será una acción de tipo estático.

5.2.4 Estructuras

Una vez que quedan relacionadas las capas relativas al apartado de acciones, se establecen las subsiguientes relaciones entre la capa de comportamientos con la capa estructural (Figura 92). Para ello se hace uso de la Tabla 33 donde se muestra la correlación de las diferentes estructuras del TULUM® con los comportamientos a partir del A-QB.

Tabla 33: Relación entre la estructura del TULUM® y sus comportamientos

Estructura	Comportamientos por B-Cube
Accessory holder	(0,1,0), (1,0,1)
Aluminum Tube (21x15 A2, B1 & 25x22)	(0,0,0), (0,1,0)
Cogged Brake (1&2)	(1,0,-1)
Handle	(1,1,1)
Handle cover	(1,0,0)
Housing cap	(1,0,0)
Housing cover	(1,0,0)
Housing (left & right)	(1,0,0), (1,0,1)
M3x20 flat head (1 & 2), M4x16 (1 & 2), M4x20, M4x54 (1 & 2), M5x16 (1, 2, 3), M5x30 flat head	(0,1,0), (1,0,1)
Nut 13 mm, 7 mm (1 - 9), 8 mm (1- 3)	(0,1,0), (1,0,0)
Plug D 25 mm	(0,0,0)
Rubber grommet (2 & 3)	(0,0,0)
Rustproof regulator	(1,0,1)
Spring 5 mm (2 & 3)	(3,1,-1)
Static Cogged Brake	(1,0,-1), (1,0,1)
TV Camera Housing	(4,1,1), (4,1,-1)
Tightening screw cap	(1,1,1), (0,1,0)
Tightening screw post	(1,0,1)
Tube joining clip	(1,0,-1),(1,0,1)
Camera Assembly	(0,1,0), (1,0,1), (4,1,1), (4,1,-1),
Camera Housing Assembly	(0,1,0), (1,0,1)
Camera Tube Assembly	(0,1,0)
Extendable Tube Assembly	(0,1,0)
Handle Assembly (1 & 2)	(1,1,1)
Handle Tube Assembly	(1,1,1), (0,1,0)
Hinge Assembly	(1,0,1), (0,1,0), (1,0,-1)
Main Housing Assembly	(1,1,1), (0,1,0)

Estructura	Comportamientos por B-Cube
Static Brake Assembly (1 & 2)	(1,0,-1), (3,1,-1), (0,1,0)
Telescopic tubes Assembly	(1,1,1), (0,1,0)
Tightening Screw Assembly ®	(1,1,1), (0,1,0), (1,0,0), (0,1,0)
TULUM	(1,1,1), (0,1,0), (4,1,1), (4,1,-1), (1,0,1)

Para mostrar gráficamente los resultados, se va a dividir su representación en 4 grupos:

- **TULUM®**
  - ***Camera Assembly*** y ***Camera Tube Assembly***
- ***Hinge Assembly***: Que para su representación se subdivide en:
  - ***Static Brake Assembly*** (1 & 2) y ***Tightening Screw Assembly***
  - ***Main Housing Assembly***
- ***Telescopic Tubes Assembly***

Sólo se representa la relación de los ensamblajes principales del TULUM® con sus respectivos comportamientos (Figura 93). Los demás gráficos se presentan en el Anexo 1.

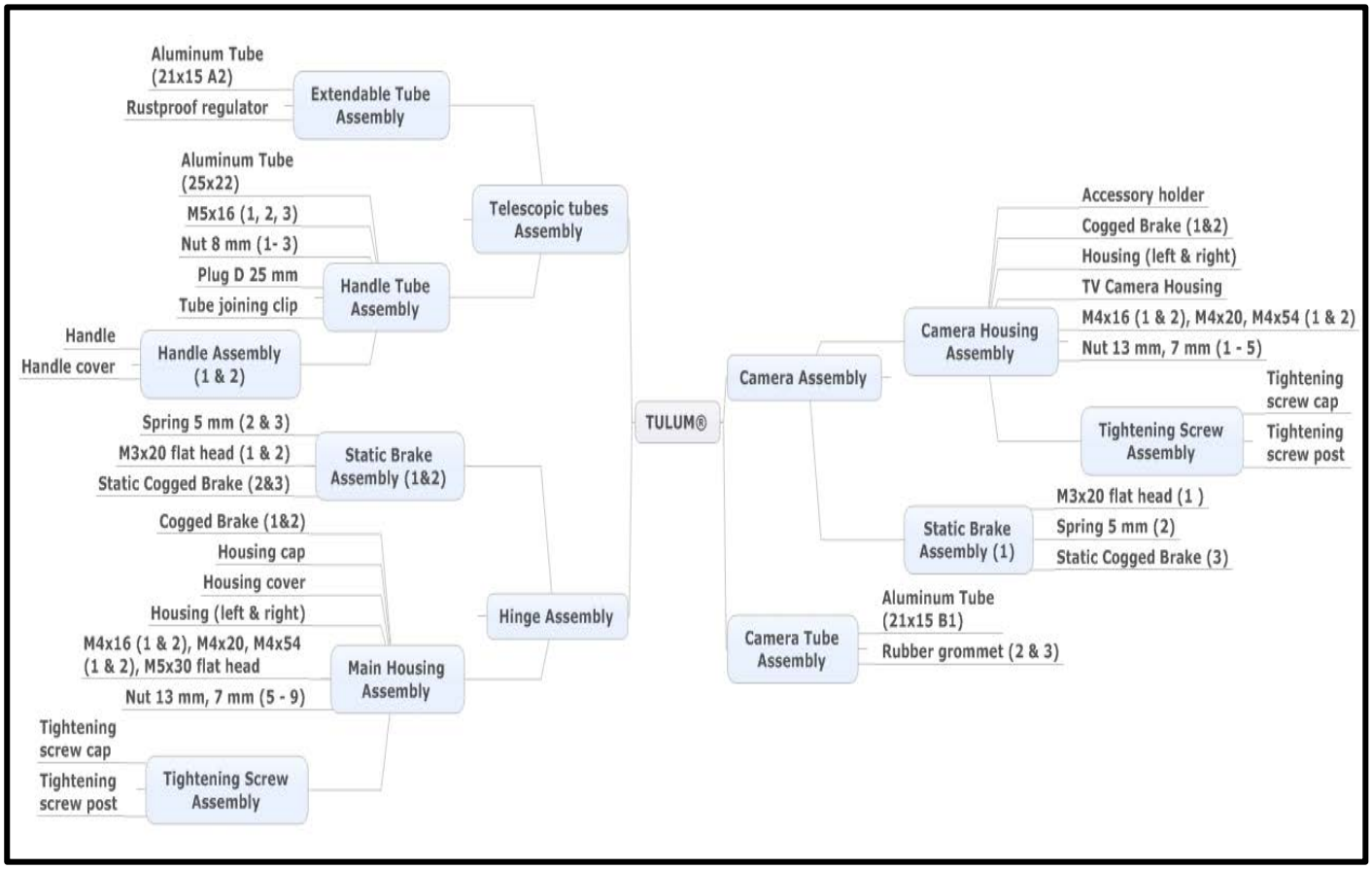


Figura 92: Estructura del TULUM®

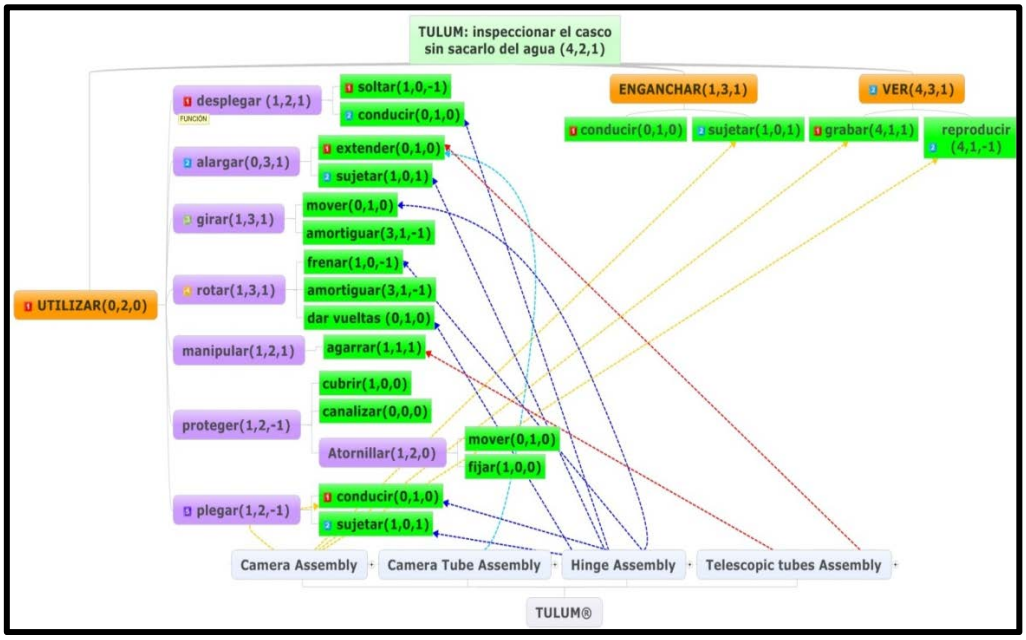


Figura 93: Gráfico de la relación de los ensamblajes principales del TULUM® con sus respectivos comportamientos

5.2.5 Entornos

En este caso, hay dos ambientes bien diferenciados. Por un lado, el entorno atmosférico que tiene el usuario que maneja el TULUM® que se encuentra fuera del agua y, por el otro, el entorno marino en el que se encuentra sumergido parte del instrumento. Estos dos entornos condicionan la elección de los materiales conformantes de la estructura así como su vida útil. Las variables físicas que se han tenido en cuenta son las siguientes:

AIRE:

Temperatura: 0-*45 ºC.*

Presión*: Atmosférica.*

Volumen específico del aire*: 0,83 m^3/kg.*

AGUA de MAR:

Temperatura: *10-25 ºC.*

Presión*: 1,1 atm.*

Volumen específico del agua*: 9,7·10^{-4} m^3/kg.*

A partir de esos dos entornos se generan 3 variantes: utilización (en el que intervienen el ambiente aéreo y el acuoso), marino (ambiente acuoso) y reposo (ambiente aéreo).

5.3 Aplicación de OntoFaBES al TULUM®

En este apartado se muestran los resultados obtenidos a través de la ontología OntoFaBES al caso del TULUM®. En términos de ingeniería ontológica, se aplica la ontología de dominio explicada en el apartado 3.1.10 para crear una ontología de tarea sobre el TULUM®.

Para ello, para mantener la concordancia en el trabajo, se sigue el mismo esquema utilizado en la descripción de OntoFaBES. Con la información indicada con anterioridad, se puede añadir la información necesaria para la evaluación de la ontología y de su capacidad de inferencia de conocimiento.

Cabe mencionar que en el apartado referido a la soluciones es donde se indican los resultados de la aplicación de las reglas lógicas.

5.3.1 Esquema general

Se parte del esquema de la ontología de dominio con la información descrita en el apartado 5.2. A partir de ahí, se rellenan las instancias correspondientes para cada clase utilizando para ello la barra *Individuals* de Protégé. La representación de las instancias creadas consta del nombre y a continuación de las instancias referidas a las propiedades que definen la clase.

En el Anexo 2 se indica la información relativa a la clase *Constraint* debido a que se centra en la información obtenida directamente del tipo de conexión entre cada una de las piezas.

5.3.2 Capa de acción

Según la información comentada en el apartado anterior la clase *Action* no tiene ninguna instancia directa.

5.3.3 Capa de función

Según la información creada en la Figura 91 se conforma la Tabla 34 de instancias de la clase Function teniendo en cuenta las Tabla 11 y Tabla **12** pues hereda dichas propiedades de la clase Action. Se debe indicar que en la Tabla 34 se omiten las propiedades:

- *isFunctionedBy*: Todas las instancias tienen en esta propiedad la instancia Tulum_english exceptuando el caso de la función *Atornillar* cuyas instancias son las relativas a los tornillos y tuercas del TULUM®.
- *isCarriedOnBy:* Todas las instancias de esta clase tienen como propiedad la instancia *User* pues para el manejo del TULUM® solo es necesaria una persona.
- *hasPerdurantAxis, hasPhysicalQualityAxis, hasTemporalQualityAxis y* se resumen a través de la propiedad *hasAQB_Axis*, donde los valores están organizados por las propiedades citadas respectivamente. Ej. La instancia Agarrar que tiene como *hasPerdurantAxis "Process", hasPhysicalQualityAxis a "Topological Connectedness"* y *hasTemporalQualityAxis* a *"Terminal",* se representará en la instancia *hasAQB_Axis* como *"Process; Topological Connectedness; Terminal".*

Tabla 34: Instancias de la clase Function

Nombre de la instancia	**Propiedades**				
	Is_A-FollowedBy	**Is_A-MadeUpBy**	**Is_A-PartOf**	**Is_A-PrecededBy**	**hasAQB_Axis**
Inspeccionar_casco	-	Enganchar, Utilizar, Ver	-	-	Accomplishment Signal Terminal
Enganchar	-	Conducir, Sujetar	Inspeccionar_casco	-	Achievement Topological Connectedness Terminal
Utilizar	Ver	Alargar, Desplegar, Girar, Manipular, Plegar, Rotar	Inspeccionar_casco	-	Accomplishment Spatial Location Immutable
Ver	-	Grabar, Reproducir	Inspeccionar_casco	Utilizar	Achievement Signal Terminal
Desplegar	Alargar	Conducir, Soltar	Utilizar	-	Accomplishment Topological Connectedness Terminal

Nombre de la instancia	Propiedades				
	Is_A-FollowedBy	**Is_A-MadeUpBy**	**Is_A-PartOf**	**Is_A-PrecededBy**	**hasAQB_Axis**
Alargar	Girar	Extender, Sujetar	Utilizar	Desplegar	Achievement Spatial Location Terminal
Girar	Rotar	Amortiguar, Mover	Utilizar	Alargar	Achievement Topological Connectedness Terminal
Rotar	Plegar	Amortiguar, Dar_vueltas, Frenar	Utilizar	Girar	Achievement Topological Connectedness Terminal
Manipular	-	Agarrar	Utilizar	-	Accomplishment Topological Connectedness Terminal
Proteger	-	Atornillar, Canalizar, Cubrir	-	-	Accomplishment Topological Connectedness Initial
Plegar	-	Conducir, Sujetar	Utilizar	Rotar	Accomplishment Topological Connectedness Initial
Atornillar	-	Fijar, Mover	Proteger	-	Accomplishment Topological Connectedness Immutable

5.3.4 Capa de comportamiento

De manera análoga al apartado anterior, se conforma la tabla de instancias de la clase *Behavior*. Se mantiene por ello la simplificación de la propiedad *hasAQB_Axis* y se omite la propiedad *is_A-MadeUpBy* ya que no hay ninguna instancia que tenga un valor en la citada propiedad.

5.3.5 Capa de estructura

De manera análoga, a partir de la información mostrada en Tabla 15, Tabla 16 y Tabla 17 y la Figura 92 se conforma la tabla de instancias de la clase *Structure*.

Tal como se observa en la Figura 48, esta clase tiene como subclases: *Agent_Structure* y *Non_Agent_Structure*. En el primer caso, *Agent_Structure* sólo dispone de una instancia (*User*) que está relacionada con una única propiedad (*carryOn*) que está relacionada con todas las funciones del TULUM® debido a lo explicado en el Aptdo. 5.3.2. Por lo tanto, la descripción de las instancias se centrará en las subclases de *Non_Agent_Structure: Assembly* y *Part*.

Debido al gran número de propiedades que tienen dichas clases, se va a realizar una tabla con las más relevantes, encontrándose el conjunto completo en el Anexo 3. La Tabla 36 recoge las instancias de la clase *Assembly* así como la Tabla 37 recoge las relativas a la clase *Part*.

5.3.6 Capa de entorno de la acción

De manera análoga al apartado anterior, a partir de la información creada se conforma la tabla de instancias de la clase Environment. En este caso sólo hay tres instancias: Marino, Reposo y Utilización.

Tabla 35: Instancias de la clase comportamiento

Nombre de la instancia	Propiedades			
	Is_A-FollowedBy	**Is_A-PartOf**	**Is_A-PrecededBy**	**hasAQB_Axis**
Conducir	Sujetar	Desplegar, Enganchar, Plegar	Soltar	Process Spatial Location Immutable
Dar_vueltas	-	Rotar	-	Process Spatial Location Immutable
Extender	Sujetar	Alargar	-	State Spatial Location Immutable
Reproducir	-	Ver	Grabar	Process Signal Initial
Canalizar	-	Proteger	-	State Spatial Location Immutable
Grabar	Reproducir	Ver	-	Process Signal Terminal
Agarrar	-	Manipular	-	Process Topological Connectedness Terminal
Frenar	-	Rotar	-	State Topological Connectedness Initial
Sujetar	-	Alargar, Enganchar, Plegar	Conducir, Extender	State Topological Connectedness Terminal
Soltar	Conducir	Desplegar	-	State Topological Connectedness Initial
Mover	Fijar	Atornillar, Girar	-	Process Spatial Location Immutable
Cubrir	-	Proteger	-	State Topological Connectedness Immutable
Amortiguar	-	Girar, Rotar		Process Energy Immutable
Fijar	-	Atornillar	Mover	State Topological Connectedness Initial

Tabla 36: Instancias de la clase *Assembly* del TULUM®

Nombre de la instancia	Propiedades			
	hasBehavior	**isAssembledBy**	**isPartOf**	**hasMass**
Camera_assembly_1	Conducir, Grabar, Reproducir, Sujetar	*Camera_housing_assembly_1, Static_brake_assembly_1*	Tulum___english	605,6 gr.
Camera_housing_assembly_1	Conducir, Sujetar	*Accessory_holder_1, Cogged_brake_1, Cogged_brake_2, Housing_left_1, Housing_right_1, M4x16_1, M4x16_2, M4x20_1, M4x54_1, M4x54_2, Nut_13mm_1, Nut_7mm_1, Nut_7mm_2, Nut_7mm_3, Nut_7mm_4, Nut_7mm_5, TV_camera_housing_1, Tightening_screw_assembly_2*	*Camera_assembly_1*	372,8 gr.
Camera_tube_assembly_1	Extender	Aluminium_tube_21x15_B_1, Rubber_grommet_2, Rubber_grommet_3	Tulum___english	261,8 gr.
Extendable_tube_assembly_1	Extender	Aluminium_tube_21x15_A_2, Rustproof_regulator_1	*Telescopic_tubes_assembly_2*	249,2 gr.
Handle_assembly_1 & 2)	Agarrar	Handle_1, Handle_cover_1	*Handle_tube_assembly_1*	83,7 gr.
Handle_tube_assembly_1	Agarrar, Extender	Aluminium_tube_25x22_1, Handle_assembly_1, Handle_assembly_2, M5x16_1, M5x16_2, M5x16_3, Nut_8mm_1, Nut_8mm_2, Nut_8mm_3, Plug_D25mm_2, Tube_joining_clip_1	*Telescopic_tubes_assembly_2*	676,8 gr.
Hinge_assembly_1	Conducir, Dar_vueltas, Frenar, Mover, Sujetar	Main_housing_assembly_1, Static_brake_assembly_1, Static_brake_assembly_2	Tulum___english	482,2 gr.
Main_housing_assembly_1	Conducir	Cogged_brake_1, Cogged_brake_2, Housing_cap_1, Housing_cover_1, Housing_left_1, Housing_right_1, M4x16_1, M4x16_2, M4x20_1, M4x54_1, M4x54_2, M5x30_flat_head_1, Nut_13mm_1, Nut_7mm_5, Nut_7mm_6, Nut_7mm_7, Nut_7mm_8, Nut_7mm_9, Tightening_screw_assembly_2	*Hinge_assembly_1*	249,5 gr.
Static_brake_assembly_1	Amortiguar, Dar_vueltas, Frenar	M3x20_flat_head_1, Spring_5mm_2, Static_cogged_brake_3	*Camera_assembly_1, Hinge_assembly_*	232,7 gr.

Nombre de la instancia	Propiedades			
	hasBehavior	**isAssembledBy**	**isPartOf**	**hasMass**
			1	
Static_brake_assembly_2	Amortiguar, Dar_vueltas, Frenar	M3x20_flat_head_2, Spring_5mm_3 Static_cogged_brake_4	*Hinge_assembly_1*	232,7 gr.
Telescopic_tubes_assembly_2	Agarrar, Extender	*Extendable_tube_assembly_1, Handle_tube_assembly_1*	Tulum___english	926,0 gr.
Tightening_screw_assembly_2	Agarrar, Dar_vueltas, Fijar, Mover	*Tightening_screw___cap_1, Tightening_screw___post_1*	*Camera_housing_assembly_1, Main_housing_assembly_1*	38,3 gr.
Tulum___english	Agarrar, Conducir, Grabar, Reproducir, Sujetar	Camera_assembly_1, Camera_tube_assembly_1, Hinge_assembly_1, Telescopic_tubes_assembly_2	-	2275,6 gr.

Tabla 37: Instancias de la clase Part del TULUM®

Nombre de la instancia	Propiedades			
	hasBehavior	**isPartOf**	**hasMaterial**	**hasMass**
Accessory_holder_1	Conducir, Sujetar	*Camera_housing_assembly_1,*	ABS_I	39,34 gr.
Aluminum Tube (21x15 A2, B1 & 25x22)	Canalizar, Extender	*Camera_tube_assembly_1, Extendable_tube_assembly_1, Handle_tube_assembly_1,*	_6060_Alloy	247,5 gr.
Cogged Brake (1&2)	Frenar	*Camera_housing_assembly_1, Main_housing_assembly_1*	ABS_I	10,095 gr.
Handle_1	Agarrar	*Handle_ assembly_1, Handle_ assembly_2*	ABS_I	75,12 gr.
Handle_cover_1	Cubrir	*Handle_ assembly_1, Handle_ assembly_2*	ABS_I	83,7 gr.
Housing cap	Cubrir	*Main_housing_assembly_1*	ABS_I	18,137 gr.
Housing cover	Cubrir	*Main_housing_assembly_1*	ABS_I	16,048 gr.
Housing (left & right)	Cubrir, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	ABS_I	63,345 gr. 62,634 gr.
M3x20 flat head (1 & 2),	Mover, Sujetar	*Static_brake_assembly_1, Static_brake_assembly_2*	X10Cr13__mart_410_	1,207 gr.
M4x54 (1 & 2),	Mover, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	5,843 gr.

Nombre de la instancia	Propiedades			
	hasBehavior	**isPartOf**	**hasMaterial**	**hasMass**
M5x16 (1,2,3),	Mover, Sujetar	*Handle_tube_assembly_1*	X10Cr13__mart_410_	2,934 gr.
M4x16 (1 & 2),	Mover, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	2,166 gr.
M4x20,	Mover, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	2,553 gr.
M5x30 flat head	Mover, Sujetar	*Main_housing_assembly_1*	X10Cr13__mart_410_	5,02 gr.
Nut 7 mm	Fijar, Mover	*Camera_housing_assembly_1,*	X10Cr13__mart_410_	0,817 gr.
Nut 13 mm	Fijar, Mover	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	4,984 gr.
Nut 8 mm	Fijar, Mover	*Camera_housing_assembly_1,*	X10Cr13__mart_410_	1,32 gr.
Plug D25mm	Canalizar	*Handle_tube_assembly_1*	SBR	12,888 gr.
Rubber_grommet(2 & 3)	Canalizar	*Camera_tube_assembly_1*	SBR	7,118 gr.
Rustproof regulator	Sujetar	*Extendable_tube_assembly_1*	_6060_Alloy	1,845 gr.
Spring 5 mm (2 & 3)	Amortiguar	*Static_brake_assembly_1, Static_brake_assembly_2*	_50CrV4	167,44 gr.
Static Cogged Brake	Frenar, Sujetar	*Static_brake_assembly_1, Static_brake_assembly_2*	ABS_I	64,054 gr.
TV Camera Housing	Grabar, Reproducir	*Camera_housing_assembly_1*	_6060_Alloy	123,165 gr.
Tightening screw cap	Agarrar, Mover	*Tightening_screw_assembly_2*	ABS_I	17,634 gr.
Tightening screw post	Fijar	*Tightening_screw_assembly_2*	X10Cr13__mart_410	20,671 gr.
Tube_joining_clip_1	Soltar, Sujetar	*Handle_tube_assembly_1*	ABS_I	47,567 gr.

Tabla 38: Instancias de la clase *Environment* del TULUM®.

Nombre de la instancia	Propiedades				
	hasPressure	**hasTemperature**	**hasSpecificVolume**	**hasHumidity[ss]**	**environTo**
Marino	1,1	10-25	$9.7 \cdot 10^{-4}$	100%	Tulum
Reposo	1,0	0-45	0.83	0%	Tulum
Utilización	1,0	0-30	1.0	85%	Tulum

[ss] Esta propiedad se ha incluido en el caso del TULUM debido a que es un producto que se utiliza en entornos acuosos en los que el grado de humedad ambiente es un indicador relevante en una posible corrosión de alguno de sus materiales (Aptdo. 0).

5.3.7 Material

De manera análoga al apartado anterior, a partir de la información creada se conforma la tabla de instancias de la clase Material (Tabla 39 y Tabla 40).

Tabla 39: Instancias de la clase *Material* del TULUM® (1ª parte).

Nombre de la instancia	Propiedades				
	hasDensity	hasPoissonRatio	hasElastic Modulus	hasYieldStrength	hasTensile Strength
ABS_I	1.02	394000	2000	-	30
SBR	9.40	490000	6.1	9.237	13.787
X10Cr13__ mart_410_	7.70	260000	190000	-	-
_50CrV4	7.85	260000	200000	206.807	517.017
_6060_Alloy	2.63	330000	69000	27.574	68.936

Tabla 40: Instancias de la clase *Material* del TULUM® (2ª parte)

Nombre de la instancia	Propiedades			
	hasShearModulus	hasSpecificHeat	hasThermal Conductivity	hasThermalExpansion Coefficient
ABS_I	319	1386	0.226	-
SBR	2.9	-	0.14	$6.7 \cdot 10^{-4}$
X10Cr13__m art_410_	79000	520	37	$1.5 \cdot 10^{-5}$
_50CrV4	79000	500	19	$1.1 \cdot 10^{-5}$
_6060_Alloy	27000	900	200	$2.4 \cdot 10^{-5}$

5.3.8 Evaluación de OntoFaBES aplicado al TULUM®

Una vez incluidas todas las instancias indicadas en los apartados anteriores, se dispone a utilizar el razonador *Racer Pro* con el objetivo de revisar la consistencia de la ontología y clasificar la taxonomía de OntoFaBES a partir de toda la información insertada en la ontología. Dicha actividad sólo la podrá realizar el ingeniero de conocimiento. Y para evaluar detalladamente OntoFaBES se utilizará la métrica *Ontometric* (Aptdo. 2.4.5.1).

Inicialmente, en la herramienta Protégé se revisan los siguientes apartados:

- La consistencia para comprobar que todas las clases y propiedades de OntoFaBES una vez incluidas las instancias del TULUM mantienen su consistencia,
- Se clasifica la taxonomía para comprobar si debido a la introducción de nueva información hay algún cambio en la jerarquía de clases de OntoFaBES lo cual indicaría que la taxonomía de OntoFaBES no es correcta, y
- Se evalúa si alguna instancia puede ser inferida en otra clase mediante el lenguaje OWL, una vez introducida la información del TULUM. Si fuera así, la herramienta KSS 2.0 no habría formalizado correctamente las instancias generando ambigüedades.

Así los resultados obtenidos indican las siguientes conclusiones:

- OntoFaBES es consistente pues después de analizarla no se indica ningún error.
- La taxonomía de OntoFaBES está clasificada de forma adecuada pues mantiene la misma estructura que se planteaba anteriormente sin instancias.
- No hay inferencia de instancias mediante el lenguaje OWL indicando que el conocimiento del proyecto de diseño se ha formalizado satisfactoriamente.

Tabla 41: Métrica básica de OntoFaBES según Protégé

Métrica	Recuento
Axiomas	6536
Axiomas lógicos	6288
Clases	27
Propiedades de objeto	44
Propiedades de datos	53
Individuos	555
Expresividad DL	*ALCHIQ (D)*

La métrica básica de OntoFaBES se muestra en la Tabla 41[tt]. En esta tabla se pueden observar los siguientes datos:

- ***Axiomas***: El número elevado de axiomas que muestra OntoFaBES indica que es una ontología de dominio en el que se ha especificado y concretado tanto las clases y sus restricciones como las propiedades tanto de objeto como de datos. Esto se demuestra revisando la expresividad DL de ésta.
- ***Axiomas lógicos:*** La diferencia existente entre los axiomas, en general, y los axiomas lógicos son aquellos axiomas que no están generados en OWL sino a partir de las reglas en lenguaje SWRL para la inferencia de conocimiento.
- ***Clases***: El reducido número de clases en comparación con los axiomas y los individuos, indica que la taxonomía de dominio está muy acotada pese a que el dominio pueda ser muy variado en individuos. Eso se debe fundamentalmente a la búsqueda de la consistencia con la metaontología DOLCE y podría haber constituido un problema de no haber definido con exactitud las propiedades tanto de objeto y de datos.
- ***Propiedades de objeto***: Las propiedades de objeto que dupliquen las clases muestra la concreta definición de las clases.
- ***Propiedades de datos***: Caso similar a las propiedades de datos, especifica el rango de individuos que sean datos dentro de la ontología eliminando ambigüedades.
- ***Individuos***: El gran número de individuos la gran cantidad de información recopilada sobre el diseño del producto TULUM® indicando el éxito en la formalización de dicho conocimiento en instancias.
- ***Expresividad*** DL: Las siglas ALCHIQ (D) indican lo siguiente:
    - ***ALC:*** Atributo que indica que se permiten los complementos y la intersección de los conceptos, restricciones universales y una cuantificación existencial limitada.
    - ***H:*** Se permite la jerarquía de los roles.
    - ***I:*** Existen propiedades inversas.
    - ***Q:*** Hay restricciones de cardinalidad cualificadas.
    - ***(D):*** Se usan propiedades de tipos de datos.

  Este tipo de expresividad indica que la ontología hace uso de la mayoría de los recursos expresivos para definirse indicando una riqueza semántica en su planteamiento.

5.3.8.1 Evaluación según Ontometric

La métrica de OntoFaBES se puede verificar según Ontometric (Lozano Tello, 2004). Para ello se analizan los datos según las siguientes dimensiones: herramienta, lenguaje, contenido, metodología y costes. Para la evaluación se va a utilizar una escala de 5 puntos con la equivalencia indicada en la Tabla 42.

[tt] La métrica completa de OntoFaBES se puede encontrar en el Anexo 6.

Se mostrarán así los gráficos con el resultado de los factores. Las tablas mostrando los cálculos de cada uno de los ítems que conforman dichos factores se pueden encontrar en el Anexo 6. A continuación, se desglosa la métrica en las diferentes dimensiones:

Tabla 42: Equivalencia de la escala numérica

Valor	Valor numérico
Muy alto	5
Alto	4
Medio	3
Bajo	2
Muy bajo	1

- **Herramienta**: El análisis de este apartado se muestra en el Gráfico 1. Los cálculos correspondientes a dicho gráfico se pueden encontrar en la Tabla 113. Se puede observar cómo la mayoría de los valores se encuentran a partir del nivel medio. Así los factores que tienen un valor más elevado son los relativos a la visualización, edición y aspectos cooperativos. En cambio, los valores menores son aquellos referidos a la traducción y la integración.
- **Lenguaje**: En el Gráfico 2 se muestran los datos resultantes de los diferentes factores incluidos en esta dimensión. Los cálculos correspondientes a dicho gráfico se pueden encontrar en la Tabla 114. Todos los valores menos el referido a los atributos tiene el valor muy alto, cuyo valor es alto.
- **Contenido:** Los factores obtenidos para esta dimensión quedan resumidos en el Gráfico 3. Los datos referidos a los cálculos que conforman el citado gráfico se encuentran en la Tabla 115. Se observa como en la mayoría de los factores predomina el valor alto, exceptuando en el apartado de conceptos cuyo valor es muy alto.
- **Metodología:** Los factores obtenidos para esta dimensión quedan resumidos en el Gráfico 4. Los datos referidos a los cálculos que conforman el citado gráfico se encuentran en la Tabla 116. Se observa como en los factores referidos a la precisión como a la usabilidad de la metodología los valores son altos. Respecto a la madurez, el valor es medio.
- **Costes:** El análisis de este apartado se muestra en el Gráfico 5. Los cálculos correspondientes a dicho gráfico se pueden encontrar en la Tabla 117. Se puede observar cómo la mayoría de los valores son bajos siendo el que adquiere un valor muy bajo es el referido a la utilización de las licencias de la ontología.

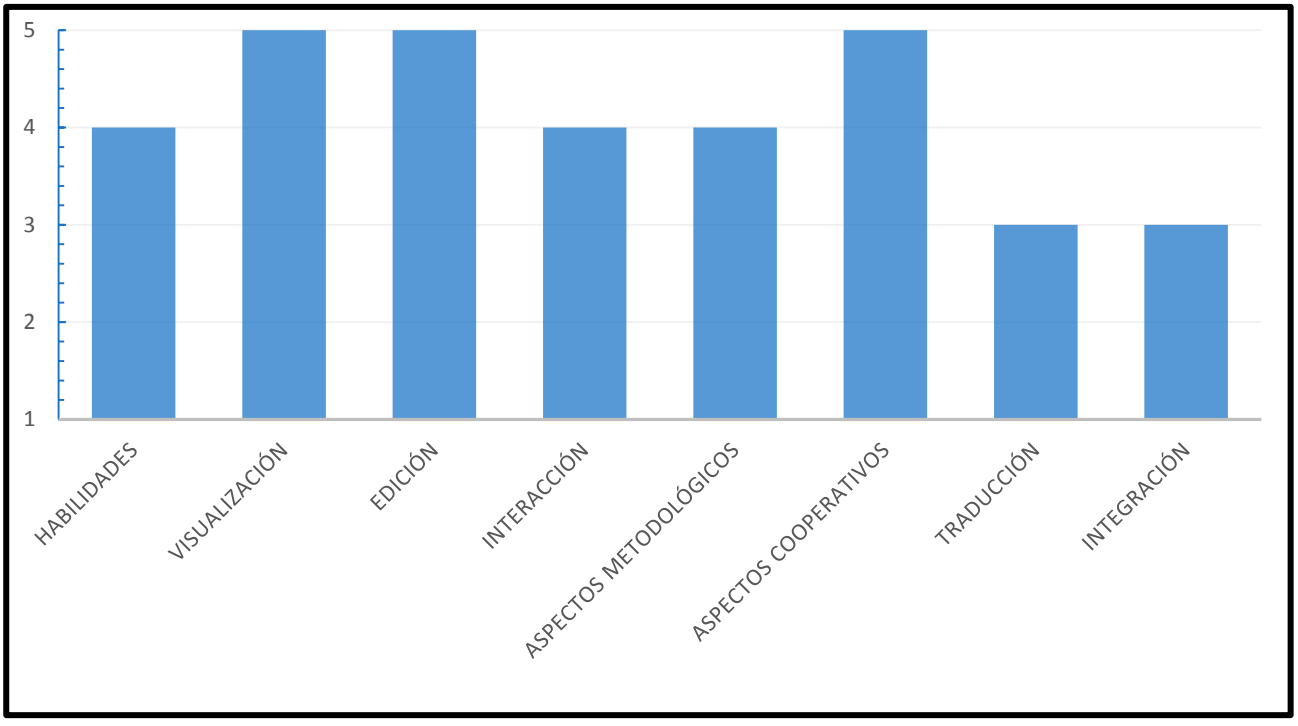


Gráfico 1: Métrica de OntoFaBES según la dimensión "Herramienta" de la métrica Ontometric.

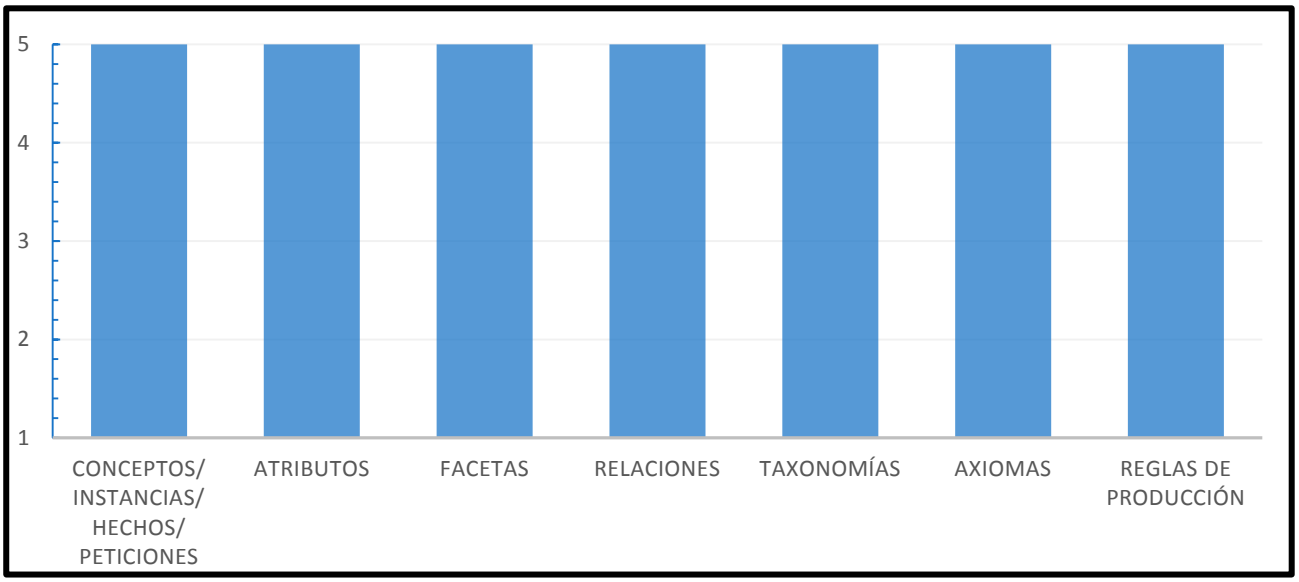


Gráfico 2: Métrica de OntoFaBES según la dimensión "Lenguaje" de la métrica Ontometric.

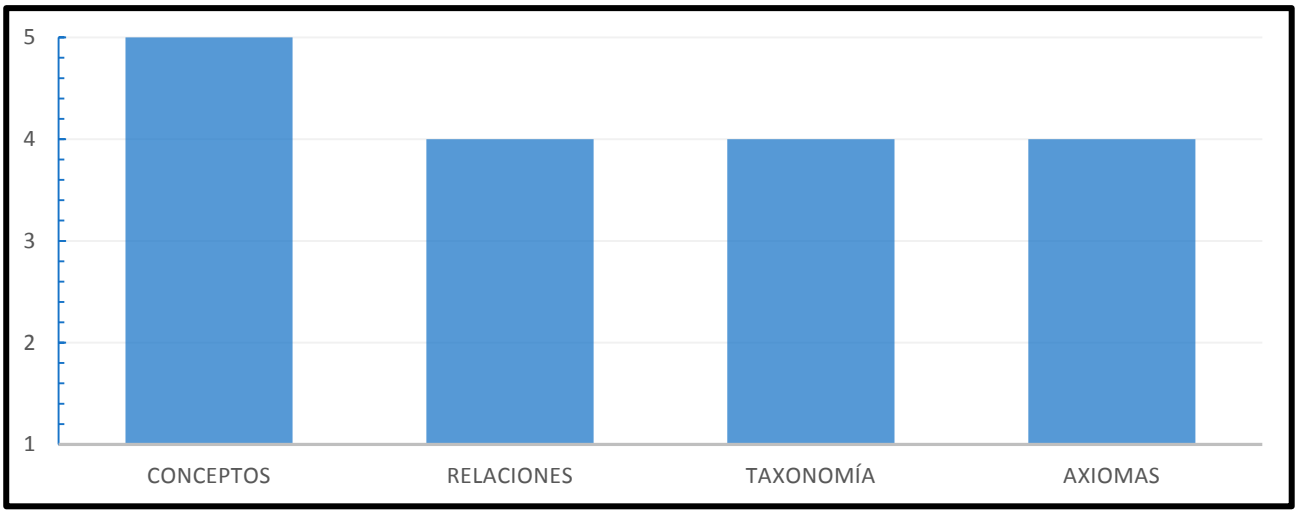


Gráfico 3: Métrica de OntoFaBES según la dimensión "Contenido" de la métrica Ontometric.

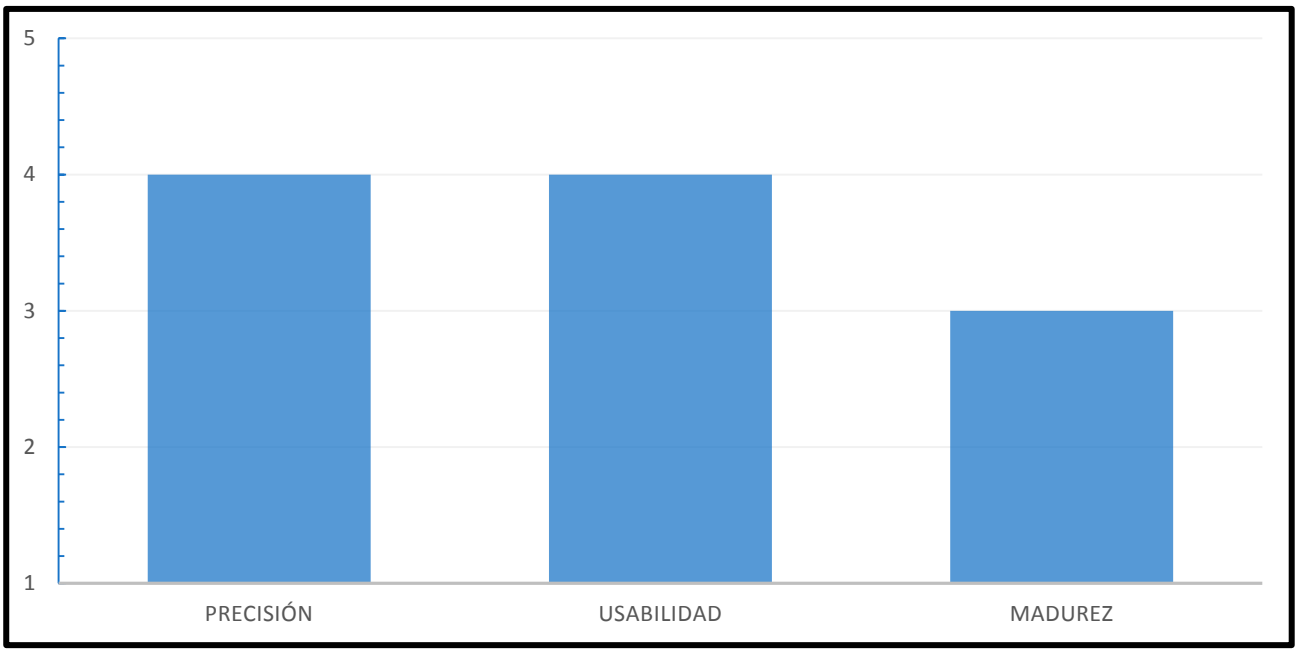


Gráfico 4: Métrica de OntoFaBES según la dimensión "Metodología" de la métrica Ontometric.

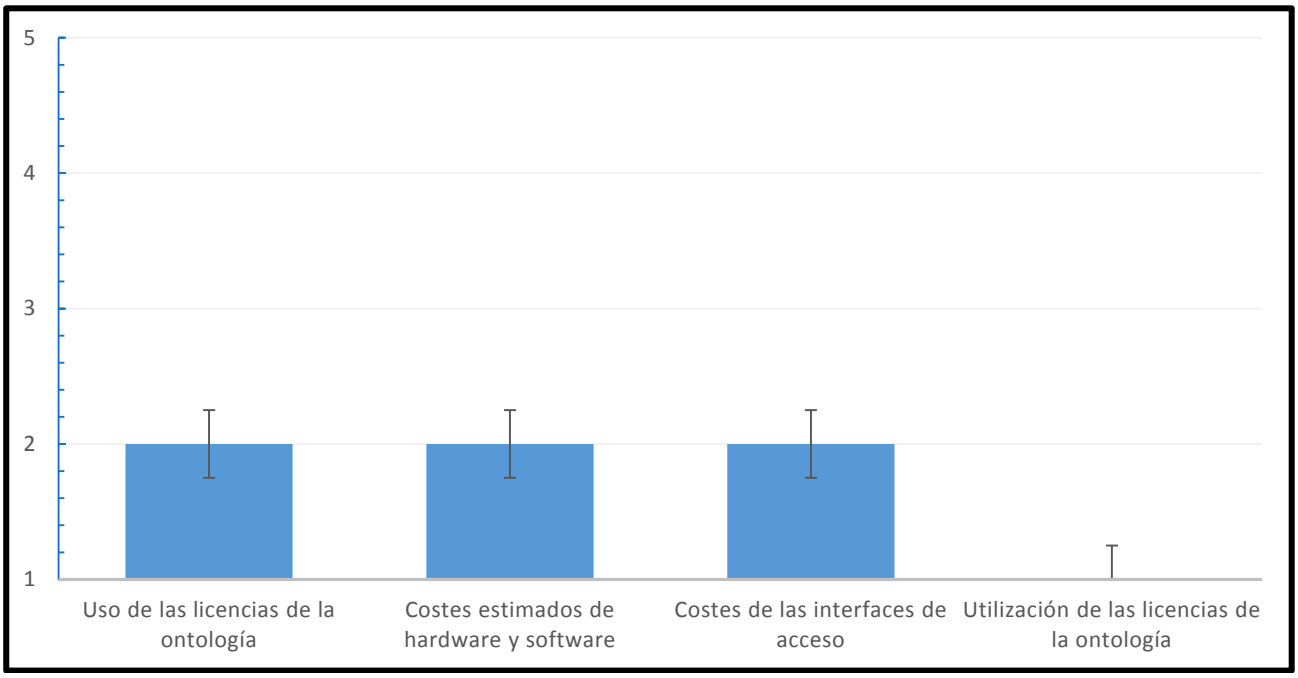


Gráfico 5: Métrica de OntoFaBES según la dimensión "Costes" de la métrica Ontometric.

Finalmente se indican los resultados obtenidos a partir de la inferencia de conocimiento obtenida a través del razonador. Esto sirve como base para la posterior inclusión de las reglas SQWRL, con el fin de plantear diversos casos de inferencia de información.

5.3.9 Inferencia de conocimiento sobre el TULUM®

Una vez recopilado todo el conocimiento en la ontología y verificado que es consistente, a partir de las reglas SQWRL es posible plantear la inferencia de nuevo conocimiento. Con el objetivo de demostrar la capacidad de OntoFaBES, se plantean diferentes casos.

5.3.9.1 Caso 1: Inferencia de estructuras a partir de funciones del TULUM®

En este apartado se plantea la inferencia de una serie de estructuras a partir del conocimiento recopilado en las funciones del TULUM® (Tabla 31). Los resultados de la función "*Alargar*" se muestran en la Tabla 43. El resto de las funciones restantes se puede encontrar en el Anexo 5.

Para ello se utiliza la Regla 1.

Este caso como el siguiente, se plantean para verificar la relación lógica entre las funciones y las estructuras (para este caso a través de la Regla 1) y las restricciones establecidas tanto a las clases como a las propiedades de OntoFaBES. El motivo de ello es que se puede comprobar si los elementos de la estructura realmente posibilitan la realización de dicha función.

A partir de los datos recopilados en la Tabla 43 se puede observar cómo todas las partes como sus ensamblajes sí que cumplen con la premisa al comprobar qué piezas son a través del Anexo 4.

5.3.9.2 Caso 2: Inferencia de funciones a partir de estructuras del TULUM®

En este apartado se plantea la inferencia de la relación de una serie de funciones a partir del conocimiento recopilado en la estructura del TULUM® (Tabla 33). Los resultados de la estructura "*Camera Assembly*" se muestran en la Tabla 44. El resto de las funciones restantes se puede encontrar en el Anexo 5.

Para ello se utiliza la Regla 2.

Este caso como el anterior, se plantean para verificar la relación lógica entre las estructuras y las funciones (para este caso a través de la Regla 2) y las restricciones establecidas tanto a las funciones

como a las subfunciones de OntoFaBES. El motivo de ello es que se puede comprobar si las funciones realmente determinan una serie de elementos de la estructura.

A partir de los datos recopilados en la Tabla 44 se puede observar cómo todas las funciones están conectadas con una serie de piezas que permiten la realización de éstas tal como se puede observar en el Anexo 5.

Tabla 43: Inferencia de una serie de estructuras a partir de la función *Alargar*

Función	Estructura
Alargar	Extendable_tube_assembly_1
	Static_cogged_brake_4
	M4x16_1
	M3x20_flat_head_1
	M4x54_2
	M4x54_1
	Aluminium_tube_25x22_1
	Housing_left_1
	Tube_joining_clip_1
	Aluminium_tube_21x15_B_1
	Accessory_holder_1
	M5x16_3
	Rustproof_regulator_1
	Static_cogged_brake_3
	Camera_tube_assembly_1
	Housing_right_1
	M3x20_flat_head_2
	M4x16_2
	M4x20_1
	M5x16_1
	Telescopic_tubes_assembly
	Camera_assembly_1
	Hinge_assembly_1
	M5x16_2
	M5x30_flat_head_1
	Extendable_tube_assembly_1
	Camera_housing_assembly_1
	Handle_tube_assembly_1
	Aluminium_tube_21x15_A_2
	TULUM

Tabla 44: Inferencia de una serie de funciones a partir de la estructura *Camera Assembly*

Estructura	Función
Camera Assembly	Alargar
	Desplegar
	Enganchar
	Plegar
	Ver

5.3.9.3 Caso 3: Inferencia de estructuras protectoras del TULUM®

En este apartado se plantea la inferencia de una serie de estructuras protectoras a partir del conocimiento recopilado en las funciones y estructura del TULUM® (Tabla 33). Los resultados de la estructura se muestran en la Tabla 46.

Para ello se utiliza la Regla 3.

Este caso, con cierta similitud al caso 1, se indica para mostrar que para la consecución adecuada de una función por parte de un objeto, se pueden establecer una serie de estructuras protectoras que aseguren su realización. Para este caso a través de la Regla 3 se puede comprobar si las funciones realmente determinan una serie de elementos con estas características.

A partir de los datos recopilados en la Tabla 46 se puede observar cómo las piezas de metal más proclives a corroerse están protegidas por otras piezas evitando así su contacto con el entorno marino.

5.3.9.4 Caso 4: Inferencia de los tipos de conexiones en el TULUM®

En este apartado se plantea la inferencia de los tipos de conexiones existentes en el TULUM® y qué piezas relaciona. Para ello se aplica la Regla 4 con las siguientes observaciones:

- En esta regla se define "*type of connection*". En las propiedades del tipo de datos de la clase Constraint (Aptdo. 3.2.8), se define que la clase *hasType* puede tener los siguientes valores: coincidente, concéntrico, perpendicular, paralelo, tangente, distante, ángulo, bloqueado desconocido y simétrico. Por tanto, esos serán los valores para el tipo de conexión.
- La propiedad establecida como *hasTypeOfConnection* se sustituye por las propiedades definidas como subpropiedades de *isConnectedTo* en la clase Estructura (Aptdo. 3.2.5.2): *hasAngleWith, hasDistanceTo, isCoincidentTo, isConcentricTo, isGearTo, isParallelTo, isPerpendicularTo, isSymmetricTo, isTangentTo, isUnknownTo*.

Por tanto para el ejemplo se aplica al caso de una conexión distante se muestra en la Tabla 45 y Tabla 47. De manera análoga se establecen el resto de las conexiones que se hallan en el Anexo 5.

Tabla 45: Aplicación de la Regla 4 al caso de una conexión distante

Definición de la relación Distante
Una NAPO S_j y una NAPO S_k tienen una conexión y definida por un tipo, sí y sólo sí S_j y S_k tienen una relación *isConnectedType_Structure1* e *isConnectedType_Structure2* con un tipo de conexión C_i, respectivamente. Dichos NAPO tienen una relación *hasType* definida como *distante* y se determina entonces que S_j y S_k tienen una relación "*hasDistanceTo*".
Restricciones implicadas
R3-1: Todo NAPO debe tener una conexión. R3-2: Toda conexión debe estar clasificada por un tipo de conexión. R3-3: Toda NAPO debe estar conectada a otra por un tipo de conexión.
Regla SQWRL
Connection_Type ($?C_i$)∧isConnectedType_Structure1 ($?C_i$, $?S_j$)∧isConnectedType_Structure2 ($?C_i$, $?S_k$) ∧ hasType($?C_i$, "*Distance*") → *hasDistanceTo* ($?S_j$, $?S_k$)

Tabla 46: Inferencia de estructuras protectoras del TULUM®.

Estructura	Estructura protectora
Aluminium_tube_21x15_B_1	Rubber_grommet_2
	Rubber_grommet_3
Aluminium_tube_25x22_1	Plug_D25mm_2
	Tube_joining_clip_1
Handle_1	Plug_D25mm_2
M3x20_flat_head_1	Static_cogged_brake_3
M3x20_flat_head_2	Static_cogged_brake_4
M4x16_1	Housing_left_1
	Housing_right_1
M4x16_2	Housing_right_1
M4x20_1	Housing_right_1
M4x54_1	Housing_right_1
M4x54_2	Housing_right_1
M5x16_1	Handle_1
M5x16_2	Handle_1
M5x16_3	Tube_joining_clip_1
M5x30_flat_head_1	Housing_cap_1
Nut_13mm_1	Cogged_brake_1
Nut_7mm_1	Housing_left_1
Nut_7mm_2	Housing_left_1
Nut_7mm_3	Housing_left_1
Nut_7mm_4	Housing_left_1
Nut_7mm_5	Housing_left_1
Nut_7mm_6	Housing_left_1
Nut_7mm_7	Housing_left_1
Nut_7mm_8	Housing_left_1
Nut_7mm_9	Housing_left_1
Nut_8mm_1	Handle_1
Nut_8mm_2	Tube_joining_clip_1
Nut_8mm_3	Handle_1
Spring_5mm_2	Static_cogged_brake_3
	Tightening_screw___post_1
Spring_5mm_3	Static_cogged_brake_4
	Tightening_screw___post_1
TV_camera_housing_1	Housing_right_1
Tightening_screw___post_1	Housing_right_1
	Tightening_screw___cap_1

Tabla 47: Resultados de la inferencia de datos de la Regla 4 para el caso de la relación *distante*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Distante	*hasDistanceTo*	M5x16_1	Handle_assembly_2
		M5x16_2	Handle_assembly_1
		M3x20_flat_head_1	Static_cogged_brake_3
		M5x16_3	Tube_joining_clip_1
		Aluminium_tube_21x15_B_1	Rubber_grommet_2
		Rubber_grommet_3	Aluminium_tube_21x15_B_1
		Spring_5mm_3	Static_cogged_brake_4
		Spring_5mm_2	Static_cogged_brake_3
		Housing_right_1	TV_camera_housing_1
		Tightening_screw___cap_1	Camera_housing_assembly_1
		Handle_1	Tube_joining_clip_1
		Handle_1	Plug_D25mm_2
		M5x16_2	Nut_8mm_3
		M5x16_3	Nut_8mm_2
		Handle_1	Handle_1

5.3.9.5 Caso 5: Inferencia de las estructuras corrosibles

En este apartado se plantea la inferencia de aquellas estructuras que tienen riesgo de corrosión en el TULUM®. Para ello se aplican las Regla 5, Regla 6 y Regla 7, con las observaciones indicadas en la Relación 5 del Aptdo. 3.3.1.

En este caso, se plantea si donde se ubica el TULUM verifica las condiciones de un entorno favorable para la corrosión atmosférica (Regla 5) además de cumplir que alguna material sea propenso a corroerse (Regla 6). Si estas condiciones se correlacionan con las propiedades en alguna pieza, entonces, se indica como resultado (Regla 7).

Los resultados muestran en la Tabla 48 el material propenso a corroerse más aquellas piezas que cumplen las premisas indicadas.

5.3.9.6 Caso 6: Inferencia de la correlación entre acciones del TULUM®

En este apartado se plantea la inferencia de las relaciones entre las acciones del TULUM® (Tabla 31 y Tabla 32). Para ello se utiliza la Regla 1, Regla 8 y Regla 9 que se presentaban en la relación 7 del Aptdo. 3.3.1.

Para este caso, el objetivo es determinar cuáles son las acciones en el TULUM® que tienen una mayor relevancia ya que están correlacionadas con otras. Para ello, se buscan aquellas acciones ejecutadas por aquellas piezas que tienen propiedades que indican su enlazamiento (tal como se indica en la Figura 16).

Los resultados de la inferencia de conocimiento en este caso se muestran en la Tabla 49. Se puede observar que hay tres acciones que cumplen las condiciones planteadas, enlazadas a otras acciones tanto en su precedencia como en su seguimiento.

Tabla 48: Resultados de la inferencia de datos de las estructuras corrosibles

Material corrosible	Estructura corrosible
X10Cr13__mart_410	M3x20_flat_head_1
	M3x20_flat_head_2
	M4x16_1
	M4x16_2
	M4x20_1
	M4x54_1
	M4x54_2
	M5x16_1
	M5x16_2
	M5x16_3
	M5x30_flat_head_1
	Nut_13mm_1
	Nut_7mm_1
	Nut_7mm_2
	Nut_7mm_3
	Nut_7mm_4
	Nut_7mm_5
	Nut_7mm_6
	Nut_7mm_7
	Nut_7mm_8
	Nut_7mm_9
	Nut_8mm_1
	Nut_8mm_2
	Nut_8mm_3
	Tightening_screw___post_1

Tabla 49: Resultados de la inferencia las relaciones entre las acciones del TULUM®.

Action	is Followed By	is Made Up By	is Part Of	is Preceded By
Rotar	Plegar	Frenar	Utilizar	Girar
		Amortiguar		
		Dar_vueltas		
Girar	Rotar	Amortiguar		Alargar
		Mover		
Alargar	Girar	Extender		Desplegar
		Sujetar		

5.3.9.7 Caso 7: Inferencia de los materiales de ensamblajes del TULUM®

En este apartado se plantea la inferencia de diferentes materiales en los ensamblajes del TULUM® (Tabla 33). Para ello se utiliza la Regla 10 que se presentaba en la relación 8 del Aptdo. 3.3.1.

En esta situación, se busca conocer los diferentes materiales que conforman los diferentes ensamblajes. Los resultados se muestran en la Tabla 50 a partir de los que se observa cómo la mayoría de los ensamblajes están formados por piezas de diferentes materiales.

Tabla 50: Resultados de los materiales de los ensamblajes del TULUM®.

Estructura	Materiales conformantes
Camera_assembly_1	ABS_I
	_50CrV4
	_6060_Alloy
	X10Cr13__mart_410_
Camera_housing_assembly_1	_6060_Alloy
	ABS_I
	X10Cr13__mart_410_
Camera_tube_assembly_1	_6060_Alloy
	SBR
Extendable_tube_assembly_1	_6060_Alloy
Handle_assembly_1	ABS_I
Handle_assembly_2	ABS_I
Handle_tube_assembly_1	_6060_Alloy
	ABS_I
	SBR
	X10Cr13__mart_410_
Hinge_assembly_1	_50CrV4
	ABS_I
	X10Cr13__mart_410_
Main_housing_assembly_1	ABS_I
	X10Cr13__mart_410_
Static_brake_assembly_1	_50CrV4
	ABS_I
	X10Cr13__mart_410_
Static_brake_assembly_2	_50CrV4
	ABS_I
	X10Cr13__mart_410_
Telescopic_tubes_assembly_2	_6060_Alloy
	ABS_I

Estructura	Materiales conformantes
	SBR
	X10Cr13__mart_410_
Tightening_screw_assembly_2	ABS_I
	X10Cr13__mart_410_
TULUM	_50CrV4
	_6060_Alloy
	ABS_I
	SBR
	X10Cr13__mart_410_

5.3.9.8 Caso 8: Inferencia del porcentaje de materiales en los ensamblajes del TULUM®

En este apartado se plantea la inferencia del porcentaje de materiales en los ensamblajes del TULUM® (Tabla 33). Los resultados de la estructura *Camera Housing Assembly* se muestran en la Tabla 51.

Para ello se utiliza la Regla 11 que se presentaba en la relación 9 del Aptdo. 3.3.1. Lo que establece dicha regla es la verificación de la masa y del material tanto de los ensamblajes como de las piezas, y si el material es el mismo en diferentes piezas se suma y se divide por el peso total del ensamblaje.

De manera análoga se establecen el resto del cálculo de los porcentajes de los materiales existentes que se pueden hallar en el Anexo 5.

Tabla 51: Resultados del porcentaje de materiales de la *Camera Housing Assembly* del TULUM®.

Ensamblaje	Partes	Material	Porcentaje (%)
Camera_housing_assembly_1	TV_camera_housing_1	_6060_Alloy	33,03
	Housing_left_1	ABS_I	16,99
	Housing_right_1		16,80
	Accessory_holder_1		10,55
	Cogged_brake_1		2,71
	Cogged_brake_2		2,71
	M4x54_1	X10Cr13__mart_410_	1,57
	M4x54_2		1,57
	Nut_13mm_1		1,34
	M4x20_1		0,68
	M4x16_1		0,58
	M4x16_2		0,58
	Nut_7mm_1		0,22
	Nut_7mm_2		0,22
	Nut_7mm_3		0,22
	Nut_7mm_4		0,22
	Nut_7mm_5		0,22

5.3.9.9 Caso 9: Inferencia de las acciones del TULUM® según el A-QB

En este apartado se plantea la inferencia de las acciones del TULUM® (Tabla 31 y Tabla 32) según el A-QB.

En este caso se pueden clasificar las acciones a partir de cualquiera de los tres ejes planteados en el A-QB: *perdurant*, cualidad física o cualidad temporal. Los resultados se muestran en la Tabla 52 clasificados según su *perdurant*.

Para ello se utiliza la Regla 12 que se presentaba en la relación 10 del Aptdo. 3.3.1.

Tabla 52: Resultados de las acciones del TULUM® según el AQ-B

Acción	*Perdurant*	Cualidad Física	Cualidad Temporal
Inspeccionar_casco	3	4	1
Desplegar	3	1	1
Manipular	3	1	1
Proteger	3	1	-1
Atornillar	3	1	0
Plegar	3	1	-1
Utilizar	3	0	0
Ver	2	4	1
Girar	2	1	1
Enganchar	2	1	1
Rotar	2	1	1
Alargar	2	0	1
Reproducir	1	4	-1
Grabar	1	4	1
Amortiguar	1	2	0
Agarrar	1	1	1
Conducir	1	0	0
Dar_vueltas	1	0	0
Mover	1	0	0
Frenar	0	1	-1
Cubrir	0	1	0
Soltar	0	1	-1
Sujetar	0	1	1
Fijar	0	1	-1
Canalizar	0	0	0

5.4 Aplicación del KSS 2.0 al TULUM®

Con el fin de poder comprender la utilización de KSS 2.0 se ha aplicado a un freno dentado (*Static Cogged Brake*), como ensamblaje perteneciente a la estructura de la cámara submarina TULUM® (Figura 94).

5.4.1 Descripción de la pieza

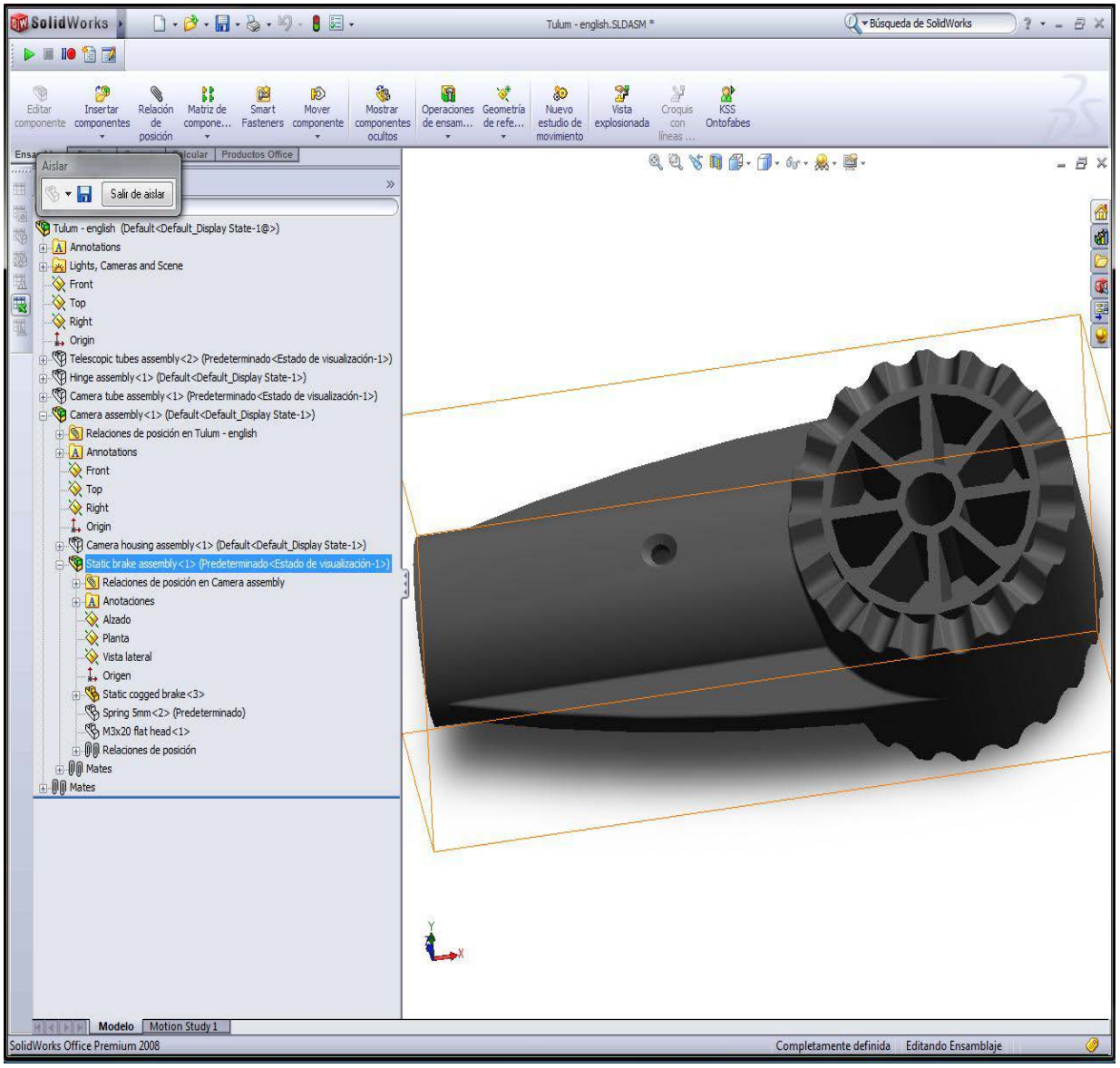


Figura 94: Freno dentado modelado en *Solidworks*

Una vez que el modelado de la pieza está acabado según la consideración del usuario de la aplicación, el diseñador ejecuta la macro KSS para formalizar el conocimiento del diseño que ha realizado. Concretamente, el freno dentado (Figura 95) forma parte del ensamblaje del freno estático (Figura 96) que constituye uno de los elementos de la cámara, constituyente fundamental del TULUM® (Figura 97).

5.4.2 Macro KSS – *Solidworks*

En el apartado del freno dentado se puede observar que tiene las siguientes propiedades (Tabla 53):

Tabla 53: Propiedades físicas del freno dentado

Masa (gr.)	64,054	
Material Info	ABS I	
	Elastic Modulus	2000 N/mm^2
	Poisson Ratio	394000
	Shear Modulus	318,9 N/mm^2
	Density	1,02 g/cm^3
	Thermal Conductivity	0,226 W/mK
	Specific Heat	1386 J/kg K
	Tensile Strength	30 N/mm^2

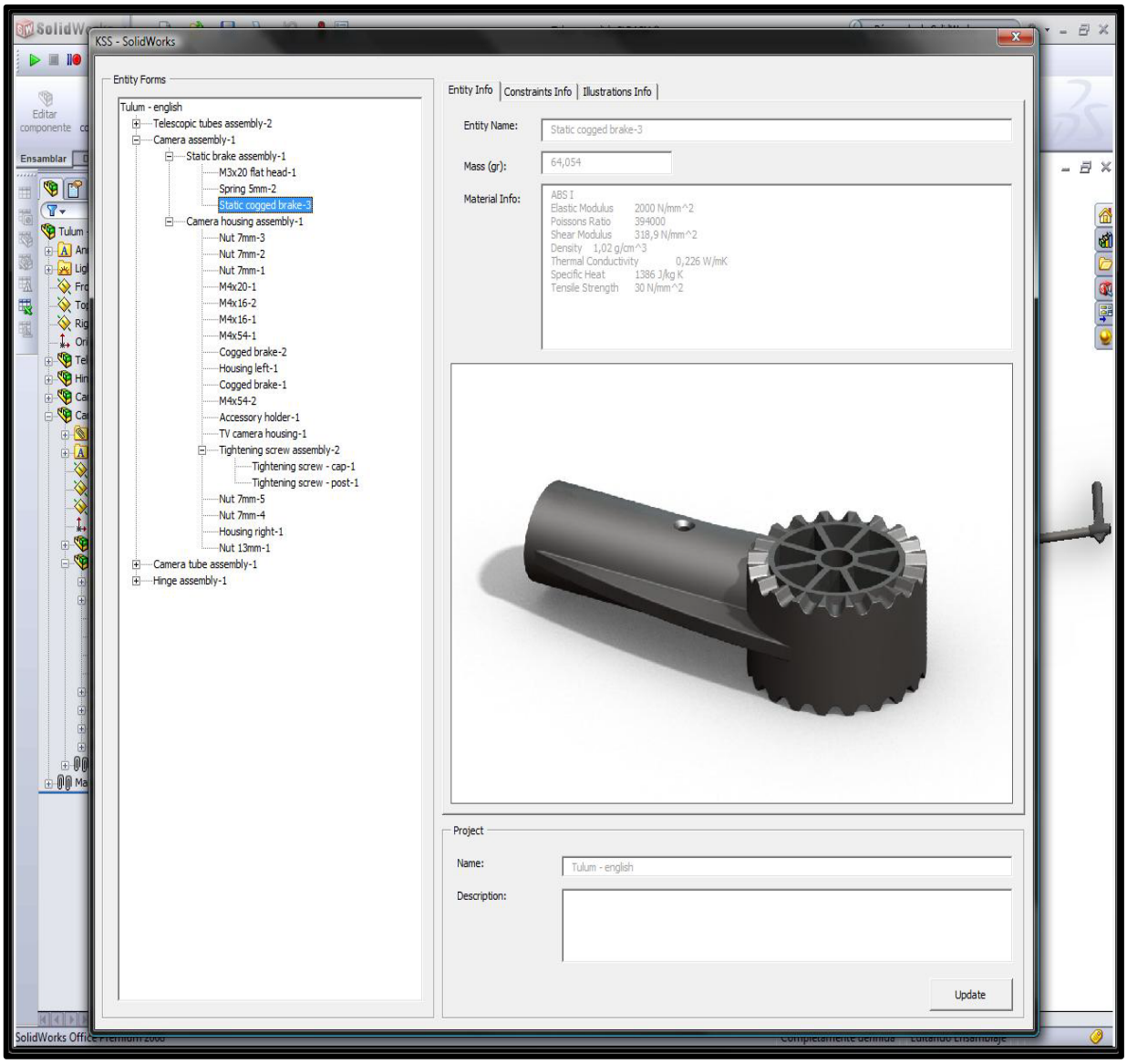


Figura 95: KSS – *Solidworks*: Static cogged brake (Entity Info)

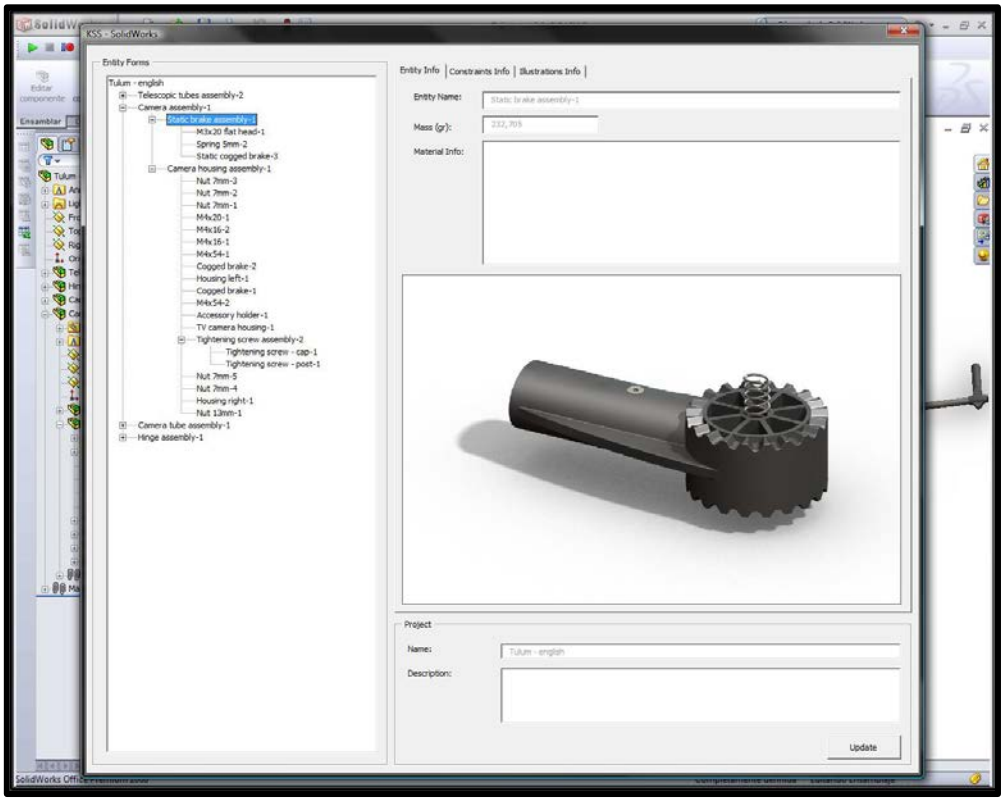


Figura 96: KSS – *Solidworks*: Static Brake Assembly (Entity info)

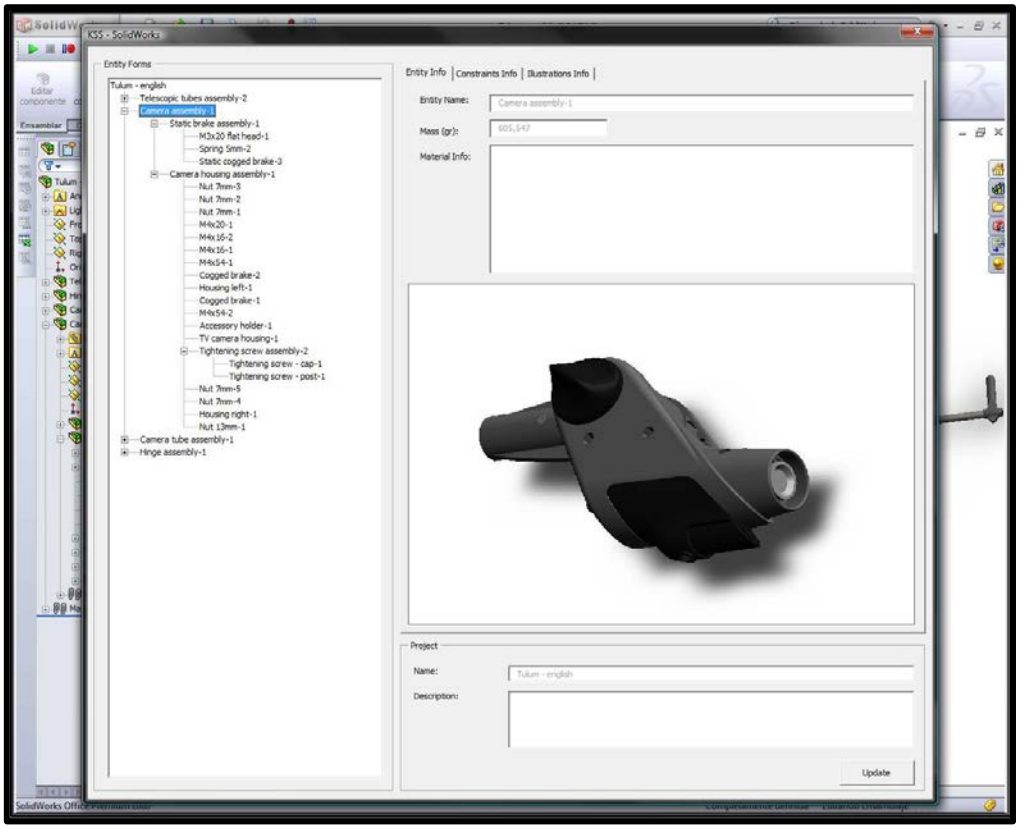


Figura 97: KSS – *Solidworks*: Camera Assembly (Entity info)

El ensamblaje del freno dentado está conformado también por un tornillo y un muelle (Figura 98).

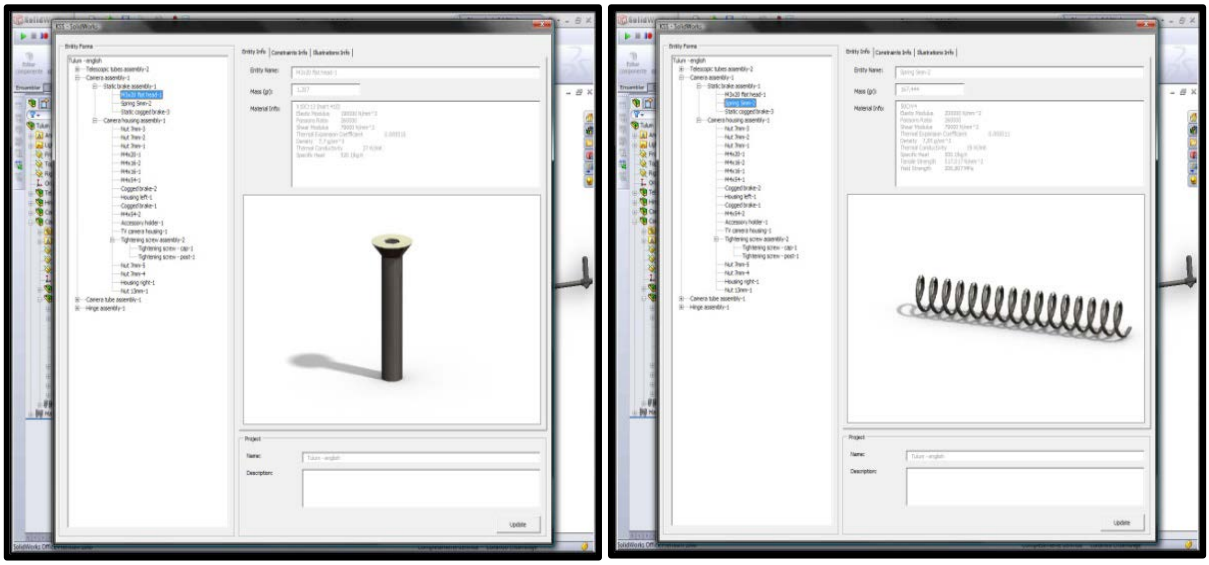


Figura 98: KSS – *Solidworks*: Spring and screw (Entity Info)

Una vez identificadas las distintas piezas que conforman el ensamblaje, el diseñador del producto puede identificar y revisar los Constraints existentes entre las distintas piezas (Figura 99). Si se considera que la información está correcta, se realiza el volcado de información al sistema KSS 2.0 y a OntoFaBES con el nombre TULUM-English. Para este caso, el tiempo de actualización se establece en 140 sec. (Figura 100).

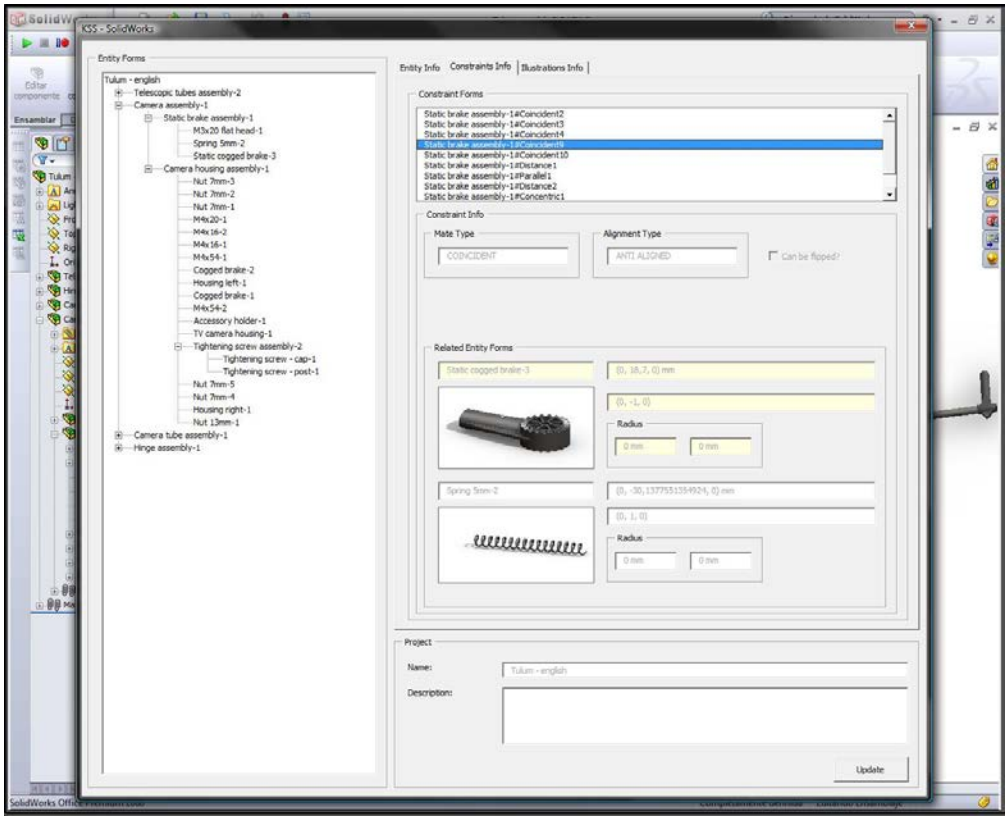


Figura 99: KSS – *Solidworks*: Static Brake Assembly (Constraints Info)

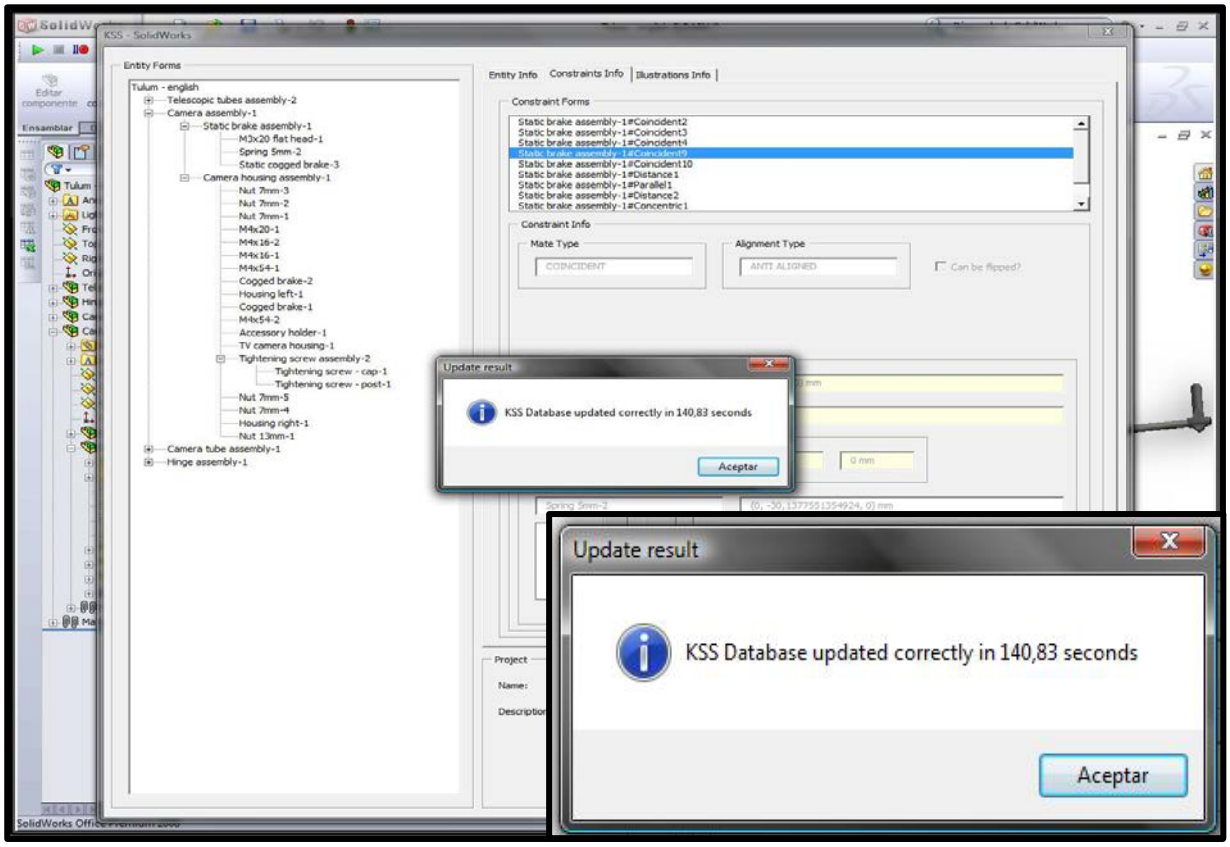


Figura 100: KSS 2.0: Volcado de información en OntoFaBES y KSS 2.0

5.4.3 KSS Web Application

Una vez introducida la información en KSS relativa a las Entities y Constraints del freno dentado, cualquier miembro del proyecto puede cargar el citado proyecto al escribir la dirección web en un navegador. Para este caso, se utiliza Firefox. Tal como se ha explicado se carga el proyecto Tulum-english (Figura 101).

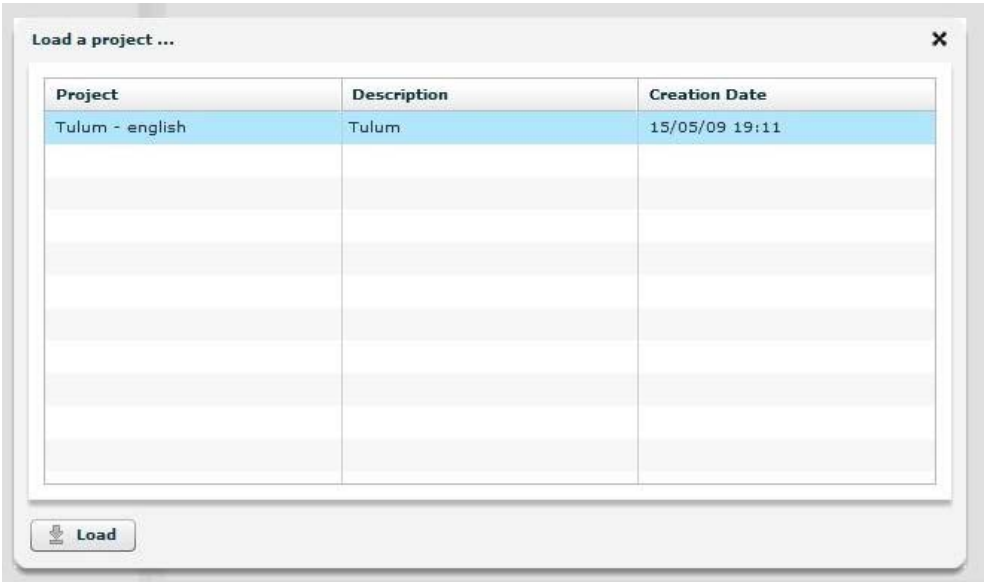


Figura 101: KSS 2.0: Carga del proyecto TULUM®

Y a través del KSS se pueden modificar los formularios de Entities (Figura 102), observar cómo se han rellenado automáticamente los ConstraintsForms (Figura 103) y visualizar las imágenes almacenadas en el Illustration Form (Figura 104) o analizar la información del Entity referido al freno dentado (Figura 105) que se ha obtenido directamente de *Solidworks* tal como se indica en el apartado 4.3.3.3 a partir del trabajo realizado por el diseñador.

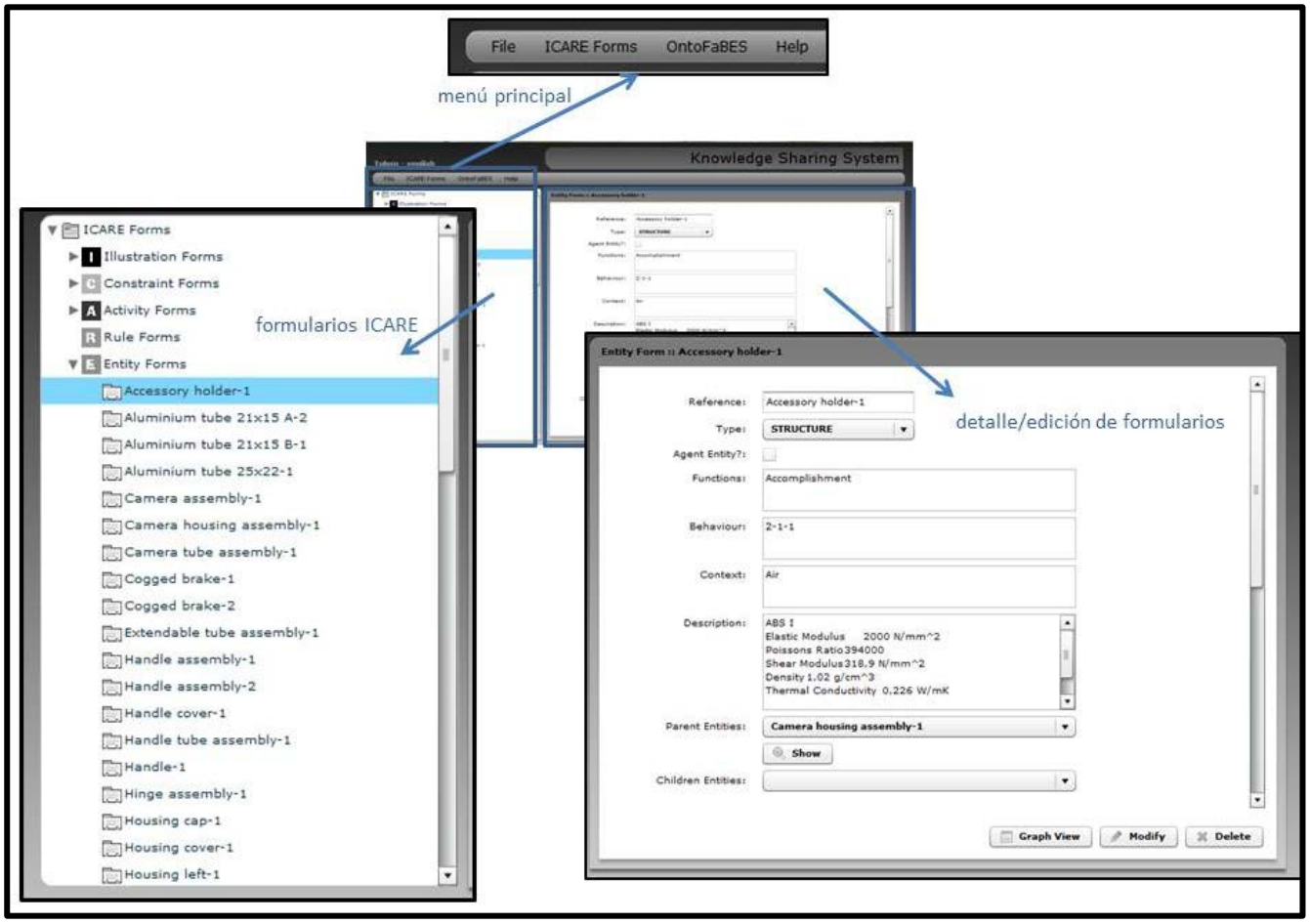


Figura 102: KSS 2.0: Accessory holder-1 (Entity Info)

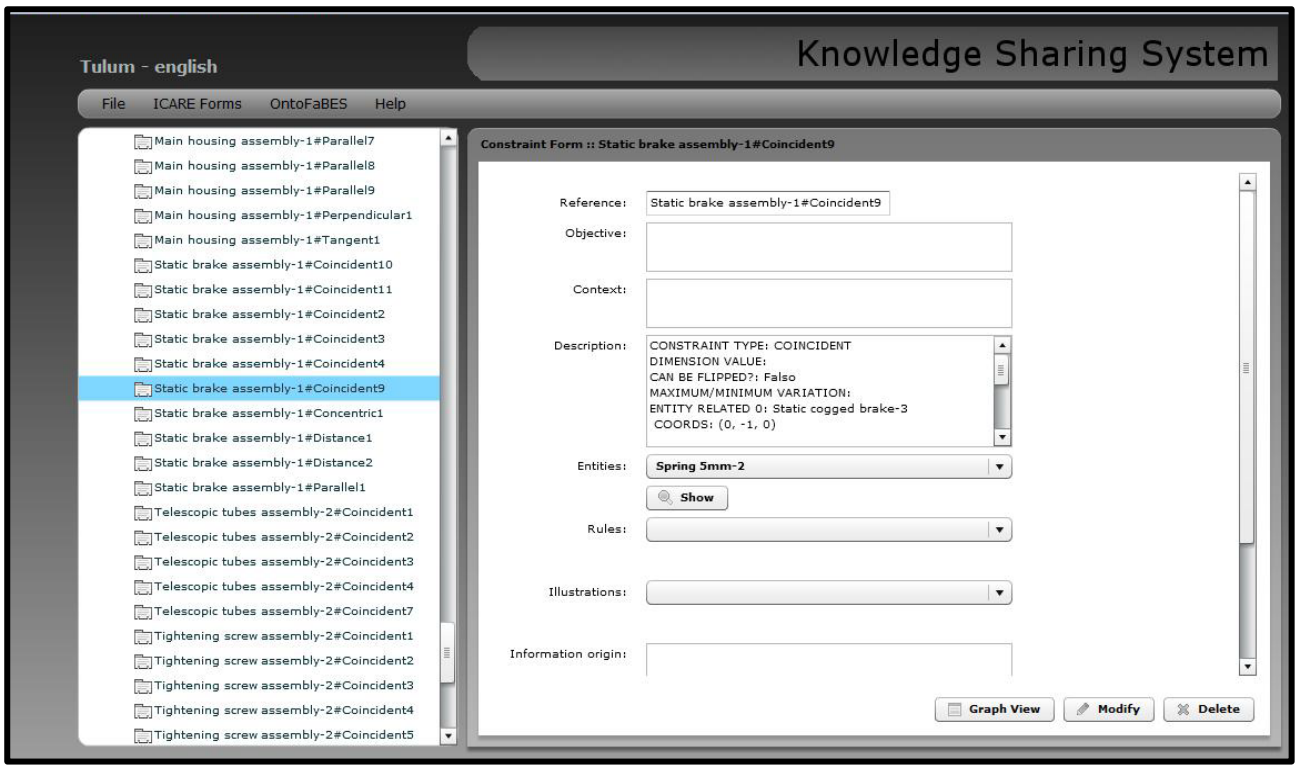


Figura 103: KSS 2.0: Static Brake Assembly (Constraint Info)

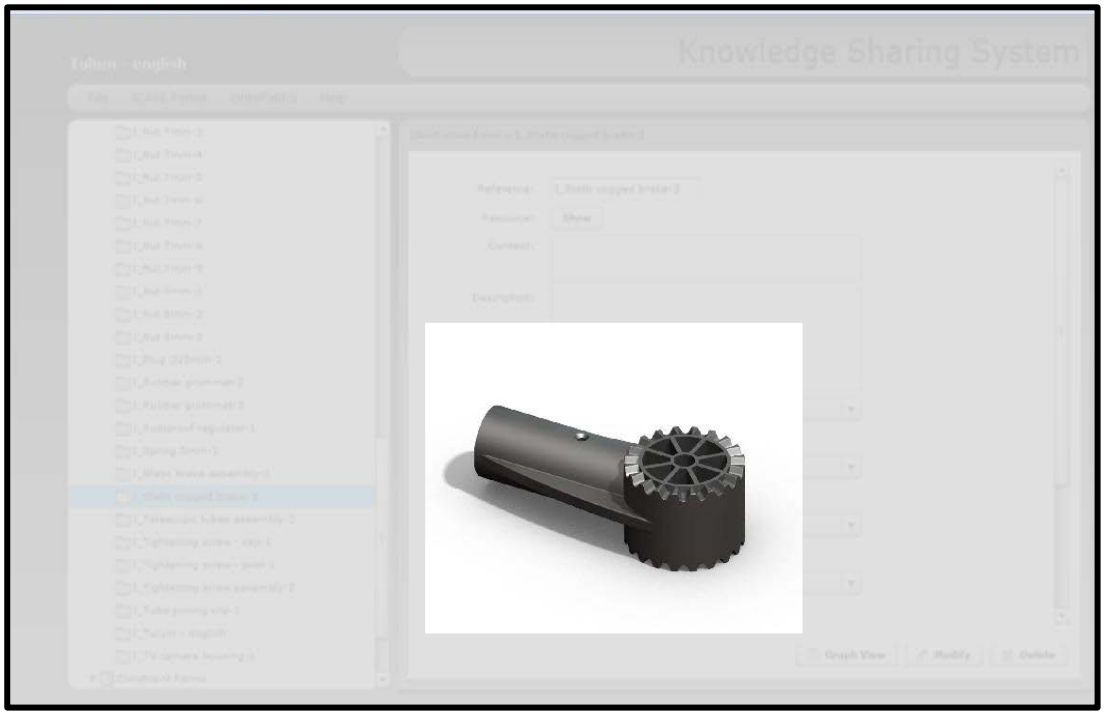


Figura 104: KSS 2.0: Static Cogged Brake (Illustrations Info)

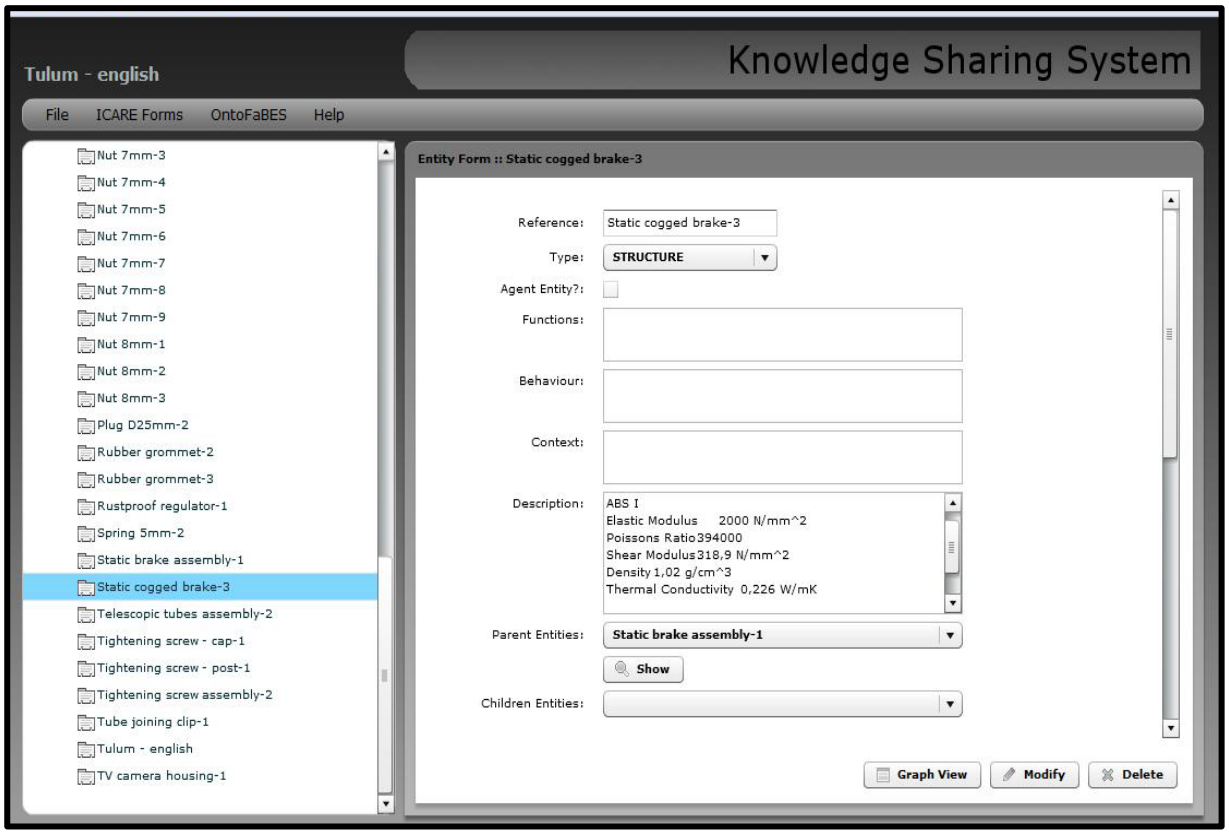


Figura 105: KSS 2.0: Static Cogged Brake (Entity Info)[uu]

[uu] En esta figura los campos Functions, Behaviour y Context están vacíos ya que se muestra la información que se ha extraído directamente del modelo de diseño a través del programa *Solidworks*. El conocimiento disponible a dichos campos puede rellenarlo el usuario de KSS Web App a través de internet o el ingeniero de conocimiento a través de la ontología OntoFaBES directamente mediante el programa Protégé.

El miembro del proyecto puede observar toda la información visualizándola gráficamente a través de las relaciones existentes entre los formularios o a través de OntoFaBES (Figura 106).

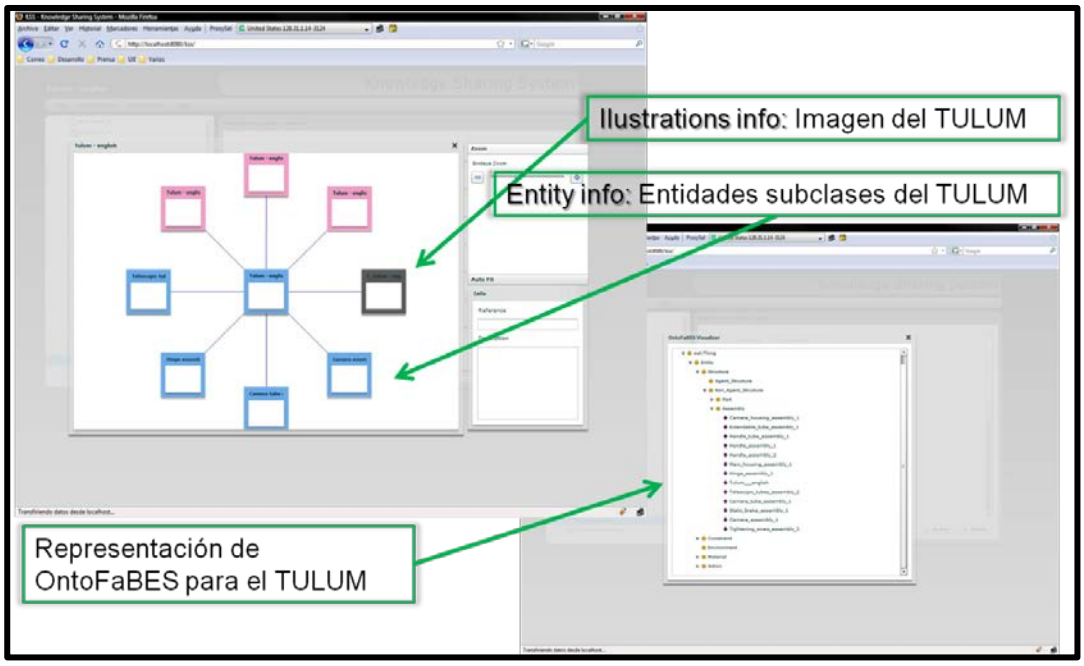


Figura 106: KSS 2.0: Visualización de las relaciones entre entidades MOKA y árbol de OntoFaBES.

5.4.4 OntoFaBES 2.0

Finalmente, el ingeniero de conocimiento puede observar en OntoFaBES cómo toda la información del freno dentado se ha persistido, incluyendo las modificaciones realizadas en KSS 2.0 a través del software Protégé (Figura 107). Dicho conocimiento queda resumido en la Tabla 54 donde se indica el uso del individuo *Static Cogged Brake* (Figura 108).

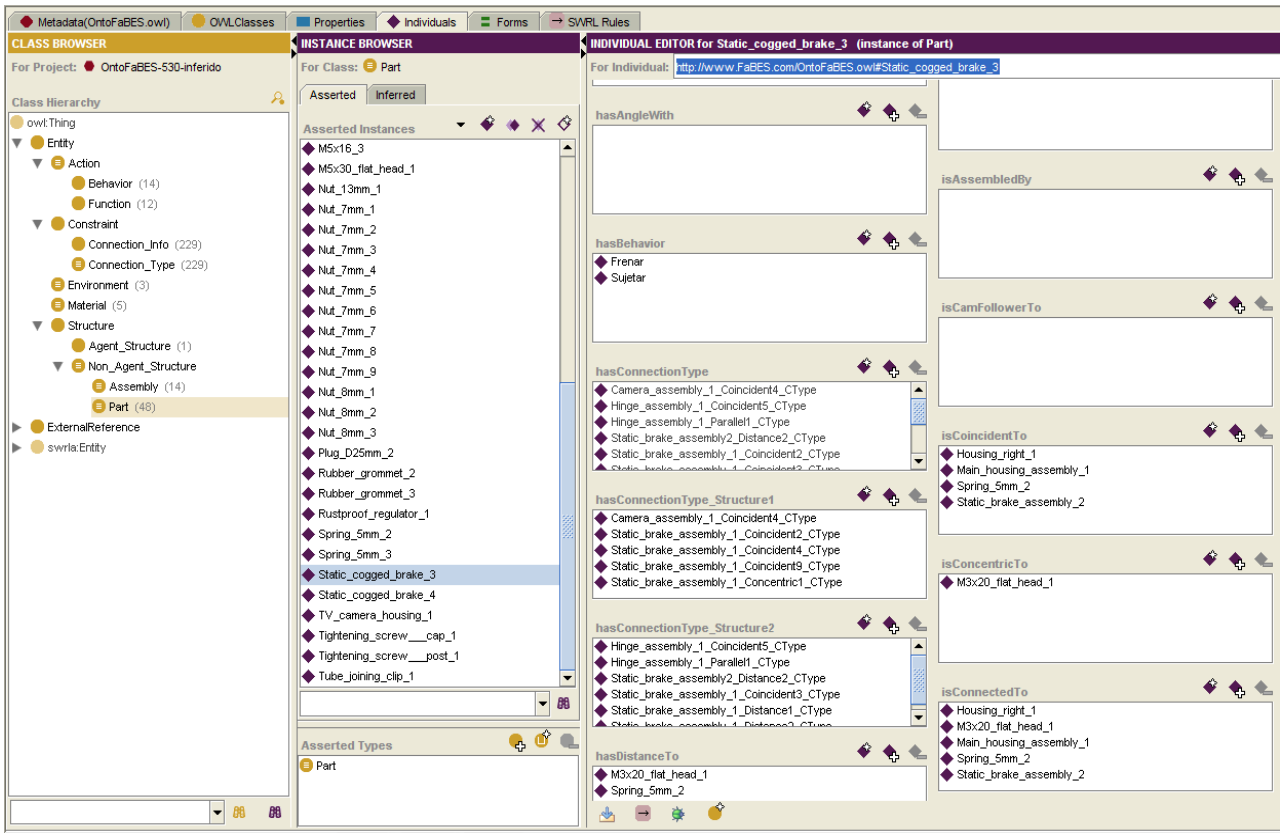


Figura 107: KSS – OntoFaBES: Static cogged brake

Tabla 54: Uso del individuo *Static Cogged Brake* en OntoFaBES.

PROPIEDAD	INDIVIDUO
hasBehavior	Sujetar
hasBehavior	Frenar
hasConnectionType_Structure1	Camera_assembly_1_Coincident4_CType
hasConnectionType_Structure1	Static_brake_assembly_1_Coincident4_CType
hasConnectionType_Structure1	Static_brake_assembly_1_Coincident2_CType
hasConnectionType_Structure1	Static_brake_assembly_1_Coincident9_CType
hasConnectionType_Structure1	Static_brake_assembly_1_Concentric1_CType
hasConnectionType_Structure2	Static_brake_assembly2_Distance2_CType
hasConnectionType_Structure2	Hinge_assembly_1_Parallel1_CType
hasConnectionType_Structure2	Hinge_assembly_1_Coincident5_CType
hasConnectionType_Structure2	Static_brake_assembly_1_Distance2_CType
hasConnectionType_Structure2	Static_brake_assembly_1_Distance1_CType
hasConnectionType_Structure2	Static_brake_assembly_1_Coincident3_CType
hasDistanceTo	M3x20_flat_head_1
hasDistanceTo	Spring_5mm_2
hasMaterial	ABS_I
isCoincidentTo	Static_brake_assembly_2
isCoincidentTo	Main_housing_assembly_1
isCoincidentTo	Spring_5mm_2
isCoincidentTo	Housing_right_1
isConcentricTo	M3x20_flat_head_1
isConnectedTo	Static_brake_assembly_2
isConnectedTo	Housing_right_1
isConnectedTo	Main_housing_assembly_1
isConnectedTo	M3x20_flat_head_1
isConnectedTo	Spring_5mm_2
isParallelTo	Main_housing_assembly_1
isPartOf	Static_brake_assembly_1

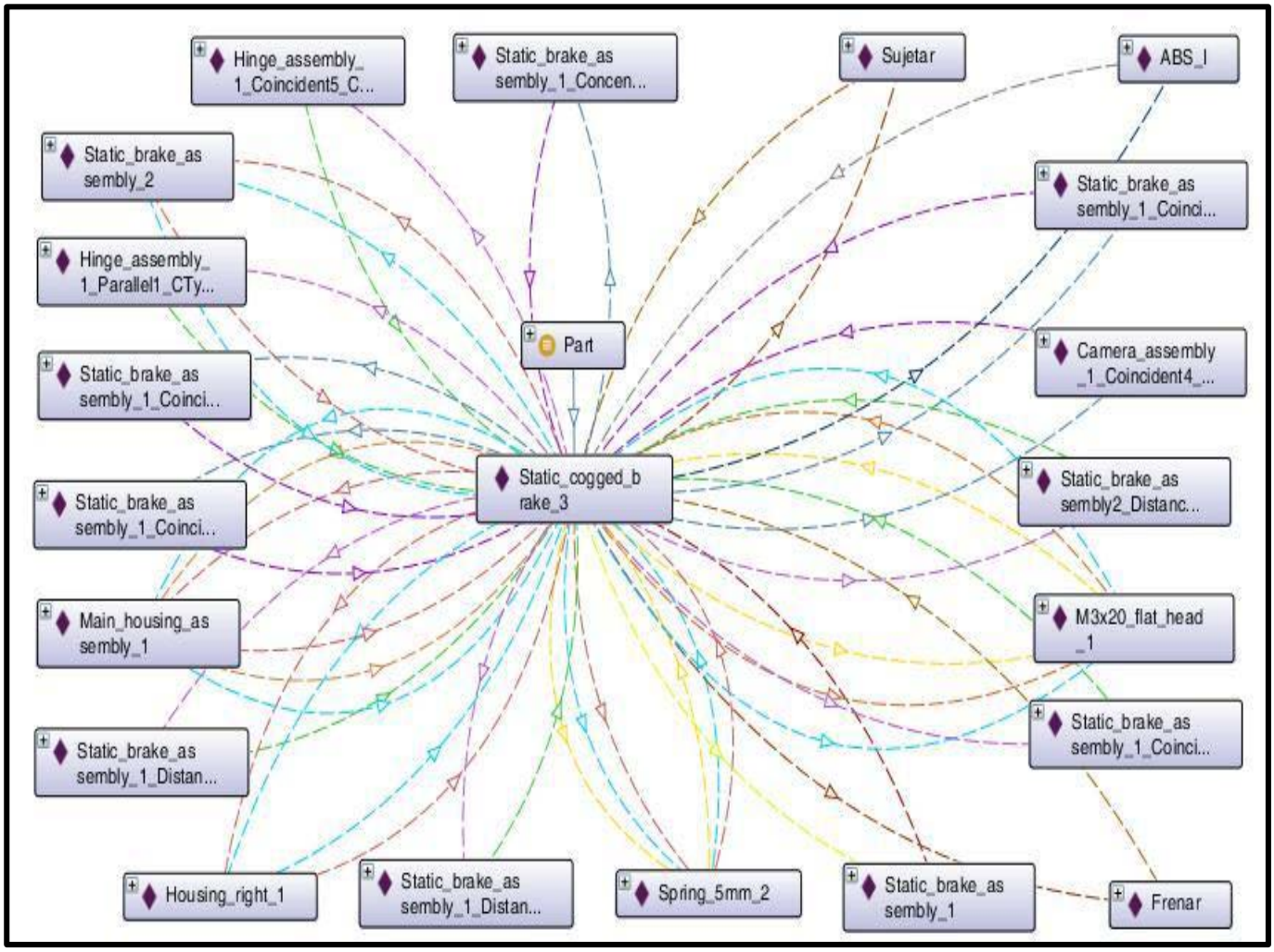


Figura 108: Uso del individuo *Static Cogged Brake* en OntoFaBES.

En la Tabla 55 se muestran las propiedades inferidas a través de las reglas 1, 3 y 4.

Tabla 55: Propiedades inferidas del individuo *Static Cogged Brake* en OntoFaBES.

PROPIEDAD	INDIVIDUO
hasFunction	Alargar
	Enganchar
	Inspeccionar casco
Estructura protectora	M3x20_flat_head
	Spring_5mm
hasDistanceTo	M3x20_flat_head
	Spring_5mm
isCoincidentTo	Main_housing_assembly
	Static_brake_assembly
	Spring_5mm
isConcentricTo	M3x20_flat_head
isParallelTo	Main_housing_assembly

5.4.5 Posibles mejoras del TULUM®

A partir de la inferencia de conocimiento de las reglas mostradas en el apartado 5.3.9 se pueden determinar una serie de mejoras para un futuro rediseño del TULUM®:

- En la Tabla 48 hay una serie de piezas expuestas al entorno marino directamente conformadas por el acero *X10Cr13 mart 410*. Se podría sustituir dicho material por otro con características mecánicas y de precio similares que no fuera sensible a un proceso corrosivo.
- En la Tabla 49 se muestran tres acciones fundamentales para el diseño del TULUM®: rotar, girar, alargar. Dichas funciones pese a no ser las tres funciones primarias, se establecen como elementos bisagra fundamentales para el éxito del diseño. Se recomendaría prestar atención a las piezas y ensamblajes que acometen dichas acciones para optimizar el proceso de rediseño.
- En la Tabla 50 se indican una serie de ensamblajes que están conformados por diferentes materiales. Y en el Anexo 5 se muestran los porcentajes de cada uno de ellos. Se podría estudiar dichos ensamblajes para evaluar la posibilidad de modificar alguno de éstos y así simplificar el proceso de montaje o desmontaje e incluso fabricación.

5.5 Discusión y conclusiones

Tal como se ha comentado en el apartado 2.4.5 las aproximaciones de las evaluaciones de ontologías se encuentran dentro de las siguientes categorías (Fortuna, 2011):

a) Aquellas basadas en la comparación de la ontología con una ontología estándar de referencia, (comparación a nivel léxico y conceptual);

OntoFaBES en su construcción se basa en una metaontología como es DOLCE, buscando una rigurosidad lógica y la universalidad de las definiciones tal como reclaman Gómez-Pérez (2004) y Fernández-López (2010) en sus estudios al respecto de la ingeniería ontológica para que tanto el fundamento teórico basado en el modelo FBS, y en la metodología MOKA se apliquen coherentemente.

b) Basadas en el uso de la ontología dentro de una aplicación para la posterior evaluación de resultados;

OntoFaBES se utiliza dentro del sistema KSS 2.0 para clasificar la información proveniente del programa CAD además de permitir la inferencia de nuevo conocimiento haciendo uso de un razonador como es *Racer* Pro.

c) Basadas en comparaciones con una fuente de datos (una colección de documentos) acerca del dominio que cubre la ontología; y,

OntoFaBES se ha construido a partir de un detallado estudio de la ingeniería del diseño, metodologías y revisión de la filosofía relativa al diseño funcional (capítulos 2.1 y 2.5) con el objetivo de lograr un conocimiento experto en el ámbito. Motivo por el cual para la creación de OntoFaBES se generó un marco nuevo de diseño denominado FaBES, que se ha comprobado su éxito en 5 ejemplos diferentes (capítulos 3.1.9 y 5.2). Y luego se ha comprobado su fiabilidad aplicándolo a un caso industrial (apartado 5.3).

d) Basadas en criterios de expertos en ontologías.

Para la creación de OntoFaBES se ha consultado a diferentes expertos además de consultar una amplia bibliografía al respecto (Apartado 3.2).

Luego cabe añadir un aspecto de discusión respecto a la aplicación de KSS 2.0 al caso práctico mostrado y es el referido al beneficio de la utilización de KSS 2.0 respecto a otros modelos de gestión del conocimiento. A ese respecto, primero cabe indicar que las características de la aplicación KSS 2.0 son muy novedosas por lo que no se ha encontrado aplicaciones similares en características al enlazar un modelo de gestión del conocimiento a una ontología de dominio de diseño funcional de productos con las cuales evaluar comparativamente su eficiencia. Sin embargo, sí que se puede explicar las ventajas con respecto a MOKA al haber aplicado KSS 2.0 sobre el TULUM®:

- Uno de los problemas iniciales de los KBE y específicamente MOKA es que para rellenar la información relativa a los formularios ICARE se hacían una serie de entrevistas y se databa a mano la información recopilada por un ingeniero del conocimiento que posteriormente se encargaba de

formalizar mediante un modelo formal (Stokes, 2001). En cambio, KSS 2.0 agiliza en gran medida ese proceso ya que, por un lado, tanto la información proveniente de los formularios *Illustrations* como los *Constraints* se obtienen automáticamente de la macro KSS-*Solidworks*. Y, por otro, la información de los restantes formularios se incluye directamente en el sistema a través de la KSS Web App, siendo verificada por la ontología OntoFaBES si la información no ha sido introducida correctamente.

- Otro de los problemas es que el modelo formal de MOKA y otros KBE es muy complejo y requiere de una curva de aprendizaje alta además de ser privado, lo cual imposibilitaba su aplicación en PYMEs (Fasth, 2000; Penoyer, 2000). Eso y también debido a otro factor, pues para su correcta aplicación hacía falta mucho personal dedicado a actualizar el conocimiento disponible. En el caso del KSS 2.0, Protégé y todas las herramientas informáticas excepto el programa CAD son de código libre, y exceptuando el ámbito de las ontologías, la curva de aprendizaje de funcionamiento del sistema no es tan exigente. Eso permitiría su utilización por parte de PYMEs con un coste de inversión más reducido.
- Y otra de las ventajas es que la información sobre el proyecto en KSS 2.0 se encuentra permanentemente online, lo cual facilitaría el trabajo deslocalizado, ahorrando costes al respecto.

Finalmente cabe discutir al respecto de los resultados obtenidos en el apdo. 5.3.8.1 El primer aspecto a destacar es que la métrica en general obtiene unos valores muy elevados en todas las dimensiones menos en la del coste por diferentes motivos:

- En este proyecto, para la creación del sistema KSS 2.0, ha sido vital que OntoFaBES fuera el núcleo además de poder facilitar el uso de los formularios ICARE a través de una interfaz ágil y disponible, como ha sido Internet. Luego comprendiendo que la utilización directa de una ontología es un proceso de aprendizaje costoso, se planteó que pudiese ser editada teniendo en cuenta las habilidades de los posibles usuarios, trabajando así las prestaciones del entorno como los aspectos de visualización, edición e interacción. Así, los valores referidos a la traducción e interpretación quedan en un valor medio ya que OntoFaBES aún no ha sido probada con otros programas CAE o metodologías de ingeniería del conocimiento para mejorar su fiabilidad, supervisión de las traducciones o lograr una fusión de OntoFaBES con otras ontologías de forma semi-automática.
- Respecto al lenguaje utilizado, se ha buscado desde el principio que los términos de los formularios ICARE tuvieran su traducción a la ontología OntoFaBES igual que aquellos términos obtenidos desde el programa CAE, en este caso, Solidworks. Además se ha utilizado para ello OWL, lenguaje ontológico que se ha convertido en el genérico para la realización de ontologías, por lo tanto es coherente que los valores hayan resultado muy altos en todos los casos.
- En referencia al contenido, los valores son muy elevados ya que desde los inicios de este trabajo de tesis, se ha tenido en cuenta todo el sistema que se estaba generando. Es decir, desde el planteamiento del marco teórico FaBES, que conforma la taxonomía, se ha tenido en cuenta que fuese riguroso lógicamente para facilitar la adaptación a OntoFaBES. Luego, se ha estudiado detenidamente para que dicha ontología fuera el núcleo del sistema de compartición KSS 2.0, conociendo y evaluando la jerarquía de conceptos, las relaciones, acciones y axiomas (que permiten desarrollar las leyes de inferencia).
- En referencia a la metodología, los valores son altos, con excepción de la madurez de ésta, ya que es la primera vez que se adaptan los formularios ICARE de la metodología MOKA a un sistema de compartición de conocimiento online mediante una ontología y a través de la obtención automática de instancias a través de un programa CAE. Para mejorar en este aspecto, se debería comprobar la utilización de OntoFaBES mediante otras metodologías.
- Finalmente, en referencia a los costes asociados es bastante congruente que los valores sean bajos ya que desde el principio del proyecto, se ha buscado generar una herramienta de bajo coste orientada para PYMEs. Eso ha generado que se plantee una herramienta que hiciera uso de programas de código libre sin necesidad de tener que pagar licencias, o con el menor coste posible asociado y mayor uso posteriormente por parte de sus usuarios. Para poder confirmar este aspecto en un futuro se podría aplicar KSS 2.0 en casos reales verificando así dicha hipótesis.

En conclusión, OntoFaBES y el sistema KSS 2.0 logran perfilar la gran parte de las necesidades planteadas inicialmente por la ingeniería ontológica verificándolo a través de un caso real como el del TULUM®.

Capítulo 6. Conclusiones y trabajos futuros

6.1 Conclusiones

En esta tesis se ha presentado KSS 2.0 un sistema de reutilización, gestión e inferencia del conocimiento adquirido del diseño de productos a través del marco de diseño funcional FBS. Para ello se ha adaptado dicho modelo al ámbito de la ingeniería ontológica a través de la ontología OntoFaBES. Finalmente para comprobar su funcionalidad se ha aplicado a un caso real: una cámara submarina. Las conclusiones principales de la presente tesis se enumeran a continuación.

Los primeros dos capítulos han servido para definir el ámbito o marco de actuación de la presente tesis. En el capítulo 2 se ha descrito la situación actual en los campos del diseño funcional, la ingeniería basada en el conocimiento, el concepto de ontologías y su aplicación en el campo de la ingeniería del diseño a través de una revisión del estado del arte sobre las citadas materias hasta el momento.

En el capítulo 3 se ha presentado AQ-B, FaBES y OntoFaBES. AQ-B es una adaptación del modelo B-Cube adaptándose al ámbito de las acciones, para poder ser formalizado lógicamente. FaBES surge como una extensión y refinamiento del modelo B-FES utilizando como fundamento el marco formal proporcionado por DOLCE y parte de las investigaciones realizadas a raíz del A-QB. Para facilitar su comprensión se aplica a cuatro casos prácticos. Y finalmente, en dicho capítulo se presenta OntoFaBES, adaptación del marco FBS para facilitar su modelado como una ontología, que adquiere su marco formal con FaBES, se programa usando la herramienta Protégé, escrita en el lenguaje OWL y utilizando el lenguaje SWRL para inferir las reglas de conocimiento.

Del capítulo 4, en donde se presenta el sistema KSS 2.0, cabe destacar cómo se logra una adaptación del modelo MOKA de metodología de gestión del conocimiento para sistemas KBE con la ontología OntoFaBES, los programas CAD, concretamente *Solidworks*, y estando disponible en una plataforma web 2.0 accesible a través de Internet. Un sistema que desarrollado de esta forma permite el flujo de información formalizado desde el diseñador pasando por el ingeniero de conocimiento hasta cualquier miembro de proyecto en el proceso de diseño conceptual (apartado 4.2). Esto permitiría la utilización de KSS 2.0 tanto en procesos de diseño de producto en los que se quiere formalizar el conocimiento de todo el proceso estando disponible para cualquier miembro del proyecto como en casos de (re)diseño de productos cuando el diseño original del producto aporta poca información bibliográfica y se desea extraer el mayor conocimiento disponible.

En el capítulo 5, a partir del ejemplo real de una cámara submarina TULUM®, se evalúa la consistencia de KSS 2.0 y por tanto de OntoFaBES. Para poder entender la aplicación informática primero se va a determinar el análisis de funciones y comportamientos haciendo uso del marco FaBES y del modelo A-QB. Posteriormente, se describe el proceso de obtención de información del diseño del TULUM® utilizando el programa de CAD *Solidworks*® 2009, su posterior formalización a través de la macro KSS-*Solidworks*, edición en la aplicación web KSS 2.0 y conversión a la ontología OntoFaBES donde se aplican una serie de reglas para inferir información proveniente del TULUM®.

Finalmente, se ha demostrado la utilidad del KSS 2.0 como herramienta para la reutilización, gestión e inferencia del conocimiento adquirido del diseño de productos a través del marco de diseño funcional FBS a través de su descripción teórica y presentación en el caso práctico del TULUM®.

6.1.1 Validación de las hipótesis

El desarrollo del sistema KSS, el marco FaBES, el modelo A-QB, la ontología OntoFaBES y su aplicación en el TULUM® ha permitido validar las hipótesis planteadas en el capítulo primero. A continuación se justifica individualmente la validación de dichas hipótesis:

Hipótesis 1: *El marco FaBES es compatible con las reglas lógicas de una ontología siendo una especificación formal y explícita de una conceptualización compartida sobre el ámbito del diseño conceptual.*

En esta tesis se demuestra que el marco FaBES cumple las exigencias marcadas por una ontología, concretamente OntoFaBES, al ser el núcleo formal en el cual ésta se basa explicitando el conocimiento del diseño conceptual, tal como se muestra en el ejemplo de la cámara submarina.

Hipótesis 2: *Una ontología desarrollada en lenguaje OWL y SWRL puede explicitar e inferir conocimiento del diseño conceptual de un producto a partir del marco FBS y la metodología MOKA.*

La ontología OntoFaBES desarrollada en lenguaje OWL y SWRL explicita el conocimiento del diseño conceptual de un producto a partir de la información recibida de *Solidworks*. Para ello hace uso del marco FaBES permitiendo rellenar los formularios ICARE de la metodología MOKA.

Hipótesis 3: *Se puede disponer de un sistema online y semi-automático de compartición e inferencia de conocimiento del diseño CAD de un producto que utilice la metodología MOKA e ingeniería ontológica para optimizar el rediseño de productos.*

La plataforma KSS 2.0 está conformada para el rediseño de productos ya que se formaliza la información por parte del programa CAD de forma automática en formularios ICARE de MOKA donde queda toda la información explicitada y representada para la inferencia de conocimiento por parte de FaBES. Además de permitir acceder a dicha información de forma deslocalizada en Internet gracias a estar basadas en RIAs.

A través de KSS 2.0 se pueden realizar múltiples proyectos anclados a diferentes ontologías de tarea asociadas a partir de la ontología de dominio OntoFaBES. Eso permite el almacenamiento y reutilización de dicho conocimiento. Además, como OntoFaBES está basado en el marco FaBES haciendo uso además de reglas SWRL se puede representar e inferir nuevo conocimiento en base al marco FBS.

6.1.2 Cumplimiento de los objetivos

De igual forma que las hipótesis, los objetivos generales quedan justificados de la manera que se detalla a continuación:

Objetivo 1: *La mejora, desarrollo e implementación de una ontología para el diseño de objetos basada en el marco de diseño funcional FaBES.*

Se ha generado OntoFaBES como una ontología lógicamente consistente. Eso implica que la citada ontología cumple los requisitos planteados en su construcción. Para ello se ha adaptado el marco FaBES al usuario y al diseñador e introduciendo una nueva interpretación del sentido de acción, acorde a las investigaciones realizadas en la actualidad. Para ello se ha adaptado el modelo A-QB para que fuera consistente con la estructura de OntoFaBES. Igualmente, se ha intentado conseguir un marco de diseño funcional de fácil aplicación para su aprendizaje. Además, a partir de OntoFaBES se han podido generar una serie de reglas en lenguaje SWRL que permiten inferir nuevo conocimiento sobre el diseño de productos.

Objetivo 2: *Desarrollar e implementar un sistema de compartición e inferencia de conocimiento basado en el marco FaBES a través de la ingeniería ontológica.*

El sistema KSS 2.0 ha conseguido embeber la funcionalidad completa de la ontología OntoFaBES automatizando la obtención de información proveniente de sistemas CAD, concretamente *Solidworks*, y facilitando la tarea de inclusión de información vía Web 2.0 así como el análisis de información vía Internet de cualquier modificación realizada en el proyecto de un diseño. Eso permite que toda la información del diseño quede formalizada automáticamente, explicitando todo el conocimiento disponible sobre el diseño. Igualmente se puede observar la consistencia del diseño a partir de la ontología.

Objetivo 3: *Aplicar el sistema de compartición e inferencia de conocimiento a un caso práctico a nivel industrial.*

Se ha aplicado favorablemente al caso de una cámara submarina tal como se ha indicado en el apartado 5.1. Para ello primero se ha determinado el análisis de funciones y comportamientos haciendo uso del marco FaBES y del modelo A-QB. Posteriormente, se ha descrito el proceso de obtención de información del diseño del producto utilizando el programa de CAD *Solidworks*® 2009, su posterior formalización a través de la macro KSS-*Solidworks*, edición en la aplicación web KSS 2.0 y conversión a la ontología OntoFaBES. Finalmente, se han mostrado los resultados obtenidos al aplicar las reglas de inferencia definidas en OntoFaBES.

6.2 Futuros trabajos y líneas de investigación

Conforme la investigación iba avanzando, fueron surgiendo diversas cuestiones de interés científico que merecen nuevas investigaciones. A continuación se exponen las líneas de investigación derivadas de la realización de esta tesis, clasificadas en diferentes ámbitos.

6.2.1 FaBES

El esquema FaBES ofrecido incide en una serie de características para facilitar su incorporación en una ontología. Se ha optado por esta evolución hacia un esquema simple y amplio con el objetivo de que haya una gran capacidad de adaptación y que sea comprensible en su utilización en el ámbito del diseño.

Con la experiencia adquirida en esta tesis, se plantea la necesidad de seguir profundizando en la correlación entre funciones y comportamientos. Para ello, también podría ser interesante enlazarlo con enfoques diversos como la teoría de diseño C–K (Hatchuel, 2009, 2013), la teoría del *affordance* (Gibson, 1977; Maier, 2009a; Maier, 2009b; Kannengieser, 2012); el esquema FES (Función- efecto- solución) para su aplicación en la innovación de productos en diseño conceptual (He, 2012); y trabajos como el de Kitamura (2006, 2013) que disfrutan de un gran reconocimiento y aplicación industrial.

También se pueden plantear nuevas corrientes de aplicación como el campo de la biomimética, es decir, la utilización de analogías biológicas para resolver problemas de diseño tanto de forma cuantitativa como cualitativa (Cheong, 2012). En la misma línea, Kitamura (2013) a través de dos modelos ontológicos pretende encontrar la interoperabilidad de las diferentes definiciones de función en artefactos, órganos biológicos y elementos naturales con el fin de lograr una visión compartida.

Además, cabe una mayor profundización en las críticas que están realizando por parte del entorno de la filosofía del diseño (Vermaas, 2012; Garbacz, 2005) respecto al modelo FBS donde analizan la relación formal de entre las subfunciones y funciones. Con un enfoque teórico, Crilly (2008) plantea que la investigación en este campo se puede mejorar considerando también el proceso de comunicación entre las intenciones de los diseñadores y las interpretaciones de los consumidores. Alineado con ello, considerando la interacción del ser humano con el objeto diseñado, como en el caso de Sun (2013), se aboga por la introducción del comportamiento del usuario y del producto durante la fase de diseño o Yu (2013) quienes a partir del marco FBS exploran los patrones de comportamiento de los diseñadores en entornos paramétricos. Cuestiones que tienen más relevancia ya que el FaBES se aplica al entorno de las ontologías donde la consistencia formal es imprescindible.

Por otro lado, se puede evaluar el grado de adaptación del esquema FaBES aplicándolo a otros ejemplos ya disponibles en la literatura que puedan ser tratados por estudiantes universitarios como el caso de un sistema de cambios de un automóvil (Álvarez Cabrera, 2012), una impresora o un vehículo autónomo (Tomiyama, 2012), un vehículo anfibio (He, 2012), un robot de cocina (Li, 2012).

6.2.2 OntoFaBES

Como ontología de dominio que es OntoFaBES, uno de los trabajos futuros consiste en realizar comparaciones con otras ontologías del dominio de la ingeniería de diseño para contrastar su eficiencia y eficacia. Una posibilidad que surge es realizar una comparación con nuevos sistemas con similar objetivo:

- *Problem map (P-map)*, ontología basada en el marco FBS para la formulación de problemas que actualmente se encuentra en una fase teórica (Dinar; 2012);
- el marco ontológico Propósito – function – espacio de trabajo – estructura – comportamiento (PFWSB - *Purpose-function-working space-structure-behavior*) para la representación del conocimiento y su propio flujo de conocimiento basado en el modelado del proceso de diseño (Zhang, 2012);
- la ontología de diseño basada en la teoría de diseño C-K en combinación con una teoría de conjuntos (Hatchuel, 2013);
- el modelo ontológico de funciones ya citado en el apartado anterior (Kitamura, 2013).

- El modelo Emisión basado en la información del sistema (IBIS - *Issue-based Information System*), ontología para la representación del conocimiento de productos (Zhang, 2013) en base al diseño racional (Lee, 1997).

De la misma manera, tal como se ha citado en los anteriores casos, sería conveniente la comprobación del funcionamiento modelando otros diseños de productos, para lo cual se podría trabajar como posible proyecto con estudiantes de grado.

Respecto a la definición de las diferentes clases de la ontología, cabe un análisis más profundo en el apartado de restricciones relativas a las conexiones entre las estructuras y sus propiedades intensivas de los materiales (resistencia, coste e impacto ambiental entre otros). Dicho campo de investigación, que está más relacionado con la ingeniería mecánica, salía fuera de los márgenes de estudio de este documento y por tanto queda abierto para análisis posteriores, tomando como base trabajos como el de Kim (2006).

En el apartado de los casos de inferencia, el tratamiento realizado para el caso de la corrosión atmosférica (Regla 5), para próximos estudios se podría analizar en mayor profundidad los distintos contaminantes implicados, relacionándolo con ontologías relativas al análisis de ciclo de vida (ACV), como Bertin (2012) que propone un nuevo enfoque para modelar inventarios de ciclo de vida (LCI – Life Cycle Inventories) utilizando una ontología y las relaciones entre grupos semánticos de procesos.

Para un futuro cabe la posibilidad de ampliar a otros ámbitos del diseño como son el diseño para desensamblaje (DfD - Design for Disassembly); fin de ciclo de vida (EOL - End Of Life) o técnicas como TRIZ. De hecho, los casos tratados de inferencia de conocimiento relativos al porcentaje de material o la corrosión atmosférica enlazarían fácilmente con la aplicación de cálculos de ACV, lo cual facilitaría al diseñador que utilizara la herramienta a poder elegir de inicio materiales más eco-sostenibles.

Otro de las cuestiones a abordar en relación a esta ontología es la realización de una comparativa entre Protégé y Hozo haciendo uso de un mismo ejemplo para mejorar la estructura de OntoFaBES y estudiar las características de Hozo. De la misma manera, se plantea comparar con otros modelos ontológicos como pueden ser Kaon.

Una cuestión que también queda abierta enlazando con el modelo A-QB es la aplicación de técnicas de análisis de textos como *Fluent Editor 2* (Kaplanski, 2013) para que los requerimientos de modificación de los elementos en el proceso de rediseño puedan modificar automáticamente la ontología. Raskin (2013) plantea en su estudio las bases para poder resolver las dificultades en la comprensión del lenguaje natural (NLU – Natural Language understanding).

6.2.3 A-QB

El modelo A-QB es una evolución del modelo B-Cube que aún tiene un largo recorrido de mejora, habiendo ya resuelto ciertas deficiencias e inconsistencias en su formulación para poder ser adaptado a una ontología como es el caso de OntoFaBES.

Se ha demostrado que en las denominaciones de las distintas acciones, éstas se ven condicionadas en su definición no sólo por la función que las organiza sino también por la pieza en concreto que realiza dicho comportamiento (Vermaas, 2012). Cabe plantearse entonces si el comportamiento debería tener una denominación mayor o si, realmente se puede observar a partir de ahí diferentes configuraciones viendo que comparten el mismo punto en el A-QB pese que el nombre sea distinto. Para poder confirmar este planteamiento, se podría comparar con otros modelos ya citados anteriormente para verificar dicha hipótesis (Vermaas, 2012; Crilly, 2013; Kitamura, 2013).

En esta tesis doctoral este modelo se ha aplicado a una serie de ejemplos concretos. Para poder validar su funcionamiento se podría ampliar a un mayor número de objetos para poder estudiar en profundidad la correlación entre los diversos ejes que lo conforman.

Otro de los aspectos en los que se debe realizar una mayor profundización es el tratamiento de las magnitudes físico-químicas (fuerza, velocidad, calor, aceleración entre otras) en referencia al eje de cualidades físicas.

Finalmente, uno de los futuros trabajos es referente a la aplicación del modelo A-QB al caso del diseño axiomático. Al respecto, cabe una mayor profundización en este campo para poder conocer el valor de la información en diferentes diseños y sobre la utilización de las funciones y comportamientos. Un estudio que puede mostrar unos indicios al respecto es el realizado por Torres (2013) en el que se integra el diseño axiomático para un sistema CAD.

6.2.4 KSS 2.0

El sistema KSS 2.0 ha sido probado en esta tesis doctoral satisfactoriamente demostrando su validez para un ejemplo concreto y está en relación a la tendencia actual de herramientas de desarrollo de forma colaborativa (Chandrasegaran, 2013). Para futuras investigaciones se podría generar una base de datos con diferentes modelos y así conocer su funcionamiento en otros contextos. Igualmente, se podría comparar con otros modelos que, basados también en el esquema FBS, pretenden integrar el proceso de diseño conceptual de productos desde una perspectiva amplia, considerando la adición de las diferentes visiones de las partes involucradas en el proceso, incluyendo vendedores, ingenieros, gestores e incluso maquinaria (Álvarez Cabrera, 2012).

De la misma manera, sería interesante que esta herramienta fuera testada por estudiantes de diseño para conocer su parecer al respecto de usabilidad y rendimiento. Si fuera satisfactorio el resultado, en un futuro podría plantearse su puesta en marcha en una PYME para poder obtener datos de campo con casos reales y poder evaluar concretamente el grado de mejora que establece la herramienta en entornos distribuidos. Dicha investigación por si misma daría pie a una línea muy amplia de investigación.

En el ámbito más teórico, se podría evaluar la posibilidad de integrar la investigación realizada por Di Maio (2013), donde se introduce el término de objetos de conocimiento (KOs -Knowledge Objects), como elementos que permiten integrar aspectos sociales (procesos, políticas) como técnicos (artefactos, algoritmos), sirviendo de base para la construcción de una arquitectura que facilite un sistema de representación del conocimiento compartido. También, Yao (2013) utiliza de forma similar las ontologías para mejorar la gestión del conocimiento en el diseño de productos, concretamente con el ejemplo de un altavoz.

Unos de los aspectos en los que se proseguirá el estudio será adaptar la plataforma Web Protégé 2.0 para que la propia ontología pueda editarse directamente desde un entorno web. Wen (2013) establece un modelo para favorecer la colaboración online en entornos multiculturales que podría favorecer la utilización del KSS 2.0 de manera global. De la misma forma, se prevé que el KSS 2.0 pueda realizar las inferencias de conocimiento y la clasificación desde el entorno web ya que actualmente dicho proceso era realizado por el ingeniero de conocimiento.

Otro de los ámbitos a los que se podría aplicar, es la introducción de esta herramienta a otros entornos como pudiera ser la gestión de proyectos, o procesos tales como servicios, tal como van Berk (2012) plantea para mejorar la comunicación entre las partes interesadas de un proyecto. En este sentido Chandrasegaran (2013) indican una serie de tendencias para el futuro en el ámbito de del diseño de productos que se podrían también introducir como es el caso de la aplicación de conceptos biológicos (diseños “bio-inspirados”), *crowdsourcing*, o que se permita evaluar la sostenibilidad del producto generado (Garraín, 2009).

KSS 2.0 se encarga de extraer el conocimiento disponible sobre el diseño de un producto en la herramienta CAD, concretamente *Solidworks*, para poder ser procesado online posteriormente a través de la ontología OntoFaBES. Sin embargo, cabe una mayor profundización para que las soluciones que brinda OntoFaBES estén disponibles también en el programa CAD como aplicación para que el diseñador pueda conocer sin necesidad de entrar en un entorno completamente externo.

Relacionado con este apartado, KSS 2.0 extrae la información de *Solidworks*, como ejemplo de programa CAE. En este sentido, una línea futura de investigación lógica sería aplicar KSS 2.0 a otras plataformas como *Pro/ENGINEER*, *NX*, *Solid Edge* o *CATIA*, con el fin de que estuviera disponible independientemente de la aplicación utilizada, lo cual sería consistente al objetivo de reducir costes. En este sentido Tessier (2013) plantea una ontología para lograr una interoperabilidad de datos entre diferentes plataformas CAD que podría servir de ayuda.

En el apartado de programación y diseño web, KSS 2.0 fue desarrollado en base a unos estándares de programación que están en constante evolución (Richards, 2009). Por tanto, unos de los trabajos que se podrían desarrollar sería la adaptación de la plataforma al protocolo HTML 5 con una posible aplicación para dispositivos móviles.

6.3 Publicaciones derivadas de la tesis

6.3.1 Revistas

- **Cebrián Tarrasón, D.** (2011): "La creatividad 2.0: Una posible realidad en torno a la web 2.0". Creatividad y Sociedad, 2011/3.

6.3.2 Capítulos de libro

1. **Cebrián-Tarrasón, D.**, Vidal, R. (2008): "Ontologies and FBS framework", Selected Proceedings from XII Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
2. Abad-Kelly J., **Cebrián-Tarrasón, D.**, and Chulvi V (2008): "An Ontology-based approach to integrating life cycle analysis and computer aided design". Selected Proceedings from XII Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
3. **Cebrián-Tarrasón, D.** and Vidal, R. (2008): "How an ontology can infer knowledge to be used in product conceptual design". *Computer-Aided Innovation* - IFIP International Federation for Information Processing, pp. 57-68.

6.3.3 Congresos nacionales e internacionales

1. **Cebrián-Tarrasón, D**. (2012): "Arquitectura del *Knowledge Sharing System* 2.0", XVI Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
2. **Cebrián-Tarrasón, D**., Vidal R., Salmerón J. L., Bertolín J. A. & Negre P. (2011): "Análisis de las empresa de base tecnológica de un parque científico y tecnológico basado en técnicas de redes sociales", XV Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
3. **Cebrián-Tarrasón, D.** (2010): "¿La Web 2.0 potencia la creatividad?", XIV Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
4. **Cebrián-Tarrasón, D.**, Garraín, D. & Vidal, R. (2009): "Justificación taxonómica del factor paisaje en el uso del suelo" XIII Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
5. **Cebrián-Tarrasón, D.** & Vidal R. (2009):" KSS 2.0: Knowledge Sharing System aplicado a la Web 2.0", XIII Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
6. **Cebrián-Tarrasón, D**., Muñoz C., Royo R. & Vidal R. (2009):" New technologies applied to teaching Engineering Design: Ontologies applied in the Moodle platform", INTED 2009 Proceedings, pp. 5121-5128.
7. Sorli M., Vidal R., **Cebrián-Tarrasón D.**, Sopelana A. & Chulvi V. (2009): Product Eco-Innovative design based on the knowledge. Product Eco-Innovative Design based on the Knowledge. 16th CIRP Engineering in the Sustainability Age. LCE 2009.
8. Garraín, D., Vidal, R. Franco, V., Muñoz, C. & **Cebrián-Tarrasón, D.** (2009):**"**Environmental considerations on land use for design and development of biopolymers", 5th International Conference on Industrial Ecology "Transitions Towards Sustainability", 21-24 Jun 2009, Lisboa (Portugal).
9. **Cebrián-Tarrasón, D.** & Vidal, R. (2008): "Las ontologías y el marco FBS", XII Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
10. **Cebrián-Tarrasón D.**, Lopez-Montero J. A. & Vidal R. (2008): "OntoFaBeS: Ontology design based in FBS framework", *CIRP Design Conference 2008*, University of Twente.
11. Chulvi V., Vidal R. & **Cebrián-Tarrasón D.** (2008): "B-Cube", XII Congreso Internacional de Ingeniería de Proyectos (AEIPRO), Zaragoza.
12. Chulvi V., **Cebrián-Tarrasón D.**, Garraín D., Abad-Kelly J., Vidal R. & Royo M. (2008): "Knowledge Based Design Education" International Conference on Engineering and Product Design Education, EPDE 08, 4 & 5 September, pp. 710-715, Barcelona Spain.

13. **Cebrián-Tarrasón D.**, Muñoz C., Chulvi V. & Vidal R. (2007): "Nuevo enfoque en el diseño inteligente de implantes craneales personalizados a través de KBE". XI Congreso Internacional de Ingeniería de Proyectos (AEIPRO).
14. Chulvi V., Sancho A., **Cebrián-Tarrasón D.**, Gimenez R., Muñoz C. & Vidal R., (2007): *Knowledge-Based Engineering in cranioplasty implant design*, 16th International Conference on Engineering Design, Paris, France.
15. Garraín D., Vidal R., Martínez P., Franco V. & **Cebrián-Tarrasón D.** (2007): *LCA of biodegradable multilayer film from biopolymers*, 3rd International Conference on Life Cycle Management, Zurich.

6.3.4 Documentos científico-técnicos

- **Cebrián-Tarrasón, D.**: "OntoFaBES: Modelado de una ontología basada en el marco FBS", Diploma de Estudios Avanzados, Dpto. Ingeniería Mecánica y Construcción. Castellón (España): Publicacions Universitat Jaume I, 2008.

6.4 Reconocimientos

- Accésit del Premio *Jaume Blasco* a la comunicación más innovadora por la comunicación "Arquitectura del *Knowledge Sharing System* 2.0" del XVI Congreso Internacional de Ingeniería de Proyectos (AEIPRO).

Referencias y bibliografía consultada

Ahmed, S. (2006): "A methodology for creating ontologies for Engineering Design.", Journal of Computing and Information Science in Engineering, 7 (2), pp. 132-140.

Ahmed, S. & Storga, M. (2007): “Engineering Design Ontologies - Contrasting an Empirical and a Theoretical Approach”, International Conference on Engineering Design, ICED’07.

Alesso, H.P., Smith C.F. (2004): "Developing Semantic Web Services", AK Peters Ltd.

Alvarez Cabrera A. A., Komoto H., van Beek T.J. & Tomiyama T. (2012): “Architecture-centric design approach for multi-disciplinary product development”, Mechatronics Conference 2012, Linz.

Ambler A. P. & Popplestone R. J. (1975): "Inferring positions of bodies from specified spatial relationships", Artificial Intelligence, 6, pp. 157-174.

Ammar-Khodja S., Perry, N., Bernard, A, (2008): “Processing knowledge to support knowledge-based engineering systems specification”, *Concurrent Engineering*, 16(1), pp. 89-101.

Andreasen, M. M. & Howard T. J. (2011): “The Future of Design Methodology”, Chapter 2: “Is Engineering Design Disappearing from Design Research?” .Springer-Verlag London Limited.

Antoniou G., Bikakis A., Dimaresis N., Genetzakis M., Georgalis G., Governatori G., Karouzaki E., Kazepis N., Kosmadakis D., Kritsotakis M., Lilis G., Papadogiannakis A., Pediaditis P., Terzakis C., Theodosaki R. & Zeginis D., (2008): "Proof Explanation for a Nonmonotonic Semantic Web Rules Language", Data & Knowledge Engineering, 64, pp. 662-687.

Armstrong D. (1997): “The World of States of Affairs”, Cambridge University Press, Cambridge.

Berners-Lee T., Hendler J., Lassila O. (2001): "The Semantic Web", Scientific American, May 2001, pp. 28-37.

Bertin B., Scuturici M., Pinon J.M., Risler E. (2012): “Semantic Modelling of Dependency Relations between Life Cycle Analysis Processes”, Lecture Notes in Computer Science, 7453, pp. 109-124.

Borgo S., Carrara M., Vermaas P. E. & Garbacz P. (2006): "Behavior of a Technical Artifact: An Ontological Perspective in Engineering", Frontiers in Artificial Intelligence and Applications, 150 (Formal Ontology in Information Systems), pp. 214–225.

Borgo S., Carrara M., Garbacz P. & Vermaas P. E. (2009): " A formal ontological perspective on the behaviors and functions of technical artifacts", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 23, pp. 3–21.

Brimble, R. & Sellini, F. (2000): "The MOKA modeling language", *Knowledge Engineering and Knowledge Management.* Methods, Models, and Tools: 12th International Conference, EKAW 2000, 1937, pp. 49.

Bryant-Arnold C.R., Stone R. B., Greer J. L., McAdams D.A., Kurtoglu T. & Campbell M. I. (2007) "A Function-Based Component Ontology for Systems Design", *International Conference on Engineering Design, ICED’07,* Cite des Sciences et de l'Industrie, Paris, France, 2007.

Bytheway, C. W. (1971): “The creative aspects of FAST diagramming”, Proc. Soc. Am. Value Eng. Conf., pp. 301-12.

Camelo D. (2007): "Modelado y desarrollo de un modelo computacional de síntesis interactivo y multirrelacional para guiar la actividad de diseño en la fase conceptual", Dpto. de Ingeniería Mecánica y Construcción, Universitat Jaume I, Castellón, Spain.

Campbell M., Cagan J. & Kotovsky K. (2003): "The a-Design Approach to Managing Automated Design Synthesis", Research in Engineering Design, 14, pp. 12 - 24.

Cascini G., Fantoni G., Montagna F. (2013): “Situating needs and requirements in the FBS framework”, Design Studies, In Press.

Chandrasegaran S. K., Ramani K., Sriram R. D., Horváth I., Bernard A., Harik R. F. & Gao W. (2013): “The evolution, challenges, and future of knowledge representation in product design systems”, Computer-Aided Design, 45 (13), pp. 204-228.

Chandrasekaran B. & Josephson J. R. (2000): "Function in device representation". Engineering with Computers, 16(3/4), pp. 162–177.

Chandrasekaran B. (2005): "Representing Function: Relating Functional Representation and Functional Modeling Research Streams", AIEDAM Artificial Intelligence for Engineering Design, 19 (2), pp. 65-74.

Chao-Chen Gu, Jie Hu, Ying-Hong Peng & Sheng Li (2012): FCBS model for functional knowledge representation in conceptual design, Journal of Engineering Design, 23(8), pp. 577-596.

Chapman C.B. & Pinfold M. (1999): “Design engineering – a need to rethink the solution using knowledge based engineering”, Advanced Technology Centre, Warnick Manufacturing Group, University of Warwick, Coventry. UK.

Cheong H., Hallihan G., & Shu L.H. (2012): “Understanding Analogical Reasoning in Biomimetic Design: An Inductive Approach” Design Computing and Cognition DCC’12, College Station, TX, June 7-9.

Christophe F., Bernard A. & Coatanéa E. (2010): “RFBS: A model for knowledge representation of conceptual design”. CIRP Annals - Manufacturing Technology 59(1), pp. 155–158.

Chui M., Miller A. & Roberts R.P. (2009): “Six Ways to make Web 2.0 work”. The McKinsey Quarterly.

Chulvi V. & Vidal R. (2013): “B-Cube, behavioural modelling of technical artefacts”, Computers in Industry, 64, pp. 68-79.

Costadoat R., Mathieu L. & Falgarone H. (2010): Design method taking into account geometric variations management along the design process. CIRP Design, Nantes - France.

Crilly N. (2012): “Function propagation through nested systems”, Design Studies, 34 (2013), pp. 216-242.

Crilly, N., Good, D., Matravers, D., & Clarkson, P. J. (2008): “Design as communication: Exploring the validity and utility of relating intention to interpretation”. Design Studies, 29 (5), pp. 425-457.

Cuenca Grau, B.; Horrocks, I.; Kazakov, Y.; & Sattler, U. (2008): “Modular reuse of ontologies: Theory and practice”. J. Artif. Intelligence Res. (JAIR) 31, pp. 273-318.

Cuenca Grau, B., Jimenez-Ruiz, E., Kharlamov, E. & Zheleznyakov, D. (2012): “Ontology evolution under semantic constraints”. In: KR, Rome, Italy.

Curbera F., Duftler M., Khalaf R., Nagy W., Mukhi N. & Weerawarana S. (2002),“Unraveling the Web Services Web an Introduction to SOAP, WSDL, and UDDI”, IEEE Internet Computing, 6(2), pp. 86-93.

Daconta M.C., Obrst L. J. & Smith K. T., (2003): “The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management”. Editado por John Wiley and Sons Inc.

D’Aquin M., Motta E., Sabou M., Angeletou S., Gridinoc L., Lopez V. & Guidi D. (2008): “Toward a New Generation of Semantic Web Applications”, IEEE Intelligent Systems, IEEE Computer Society, pp. 20-28.

DedaSys LLC, (2012):“Programming language popularity”. http://langpop.com. Se obtiene en 28/07/2013.

Deng Y. M. (1999): "A Computerized Design Environment for Functional Modeling of Mechanical Products", Fifth ACM symposium on Solid modeling and applications, ACM Press, Ann Arbor, Michigan, pp. 1-12.

Díaz Ortuño, P. M (2003): “Problemática y tendencias en la arquitectura de metadatos web”. En: Anales de documentación, 6, pp. 35–58.

Di Maio P. (2013): “Knowledge objects as shared system representation”, Knowledge Management Research & Practice, 11, pp. 23-31.

Dinar M., Maclellan C., Danielescu A. & Shah J. (2012): “Beyond Function-Behavior-Structure”, Design Computing and Cognition DCC’12, College Station, TX, June 7-9.

Du Y., Cao H., Chen X. & Wang B. (2013): "Reuse-oriented redesign method of used products based on axiomatic design theory and QFD". Journal of Cleaner Production, Volume 39, January 2013, Pages 79-86.

Dutra M., Ghodous P., Kuhn O. & Tri N. M. (2010): "A Generic and Synchronous Ontology-based Architecture for Collaborative Design". Concurrent Engineering; 18; 65, pp. 328-341.

El-Hadik S. (2009): "Integrating Adobe Flex with IBM WebSphere Portal". http://www.ibm.com/developerworks/websphere/library/techarticles/0911_el-hadik/0911_el-hadik.html WebSphere Portal wiki. Se obtiene en 26/07/12.

Erden, M., Komoto, H., van Beek, T., D'Amelio, V., Echavarria, E., & Tomiyama, T. (2008): "A Review of Function Modeling: Approaches and Applications", AI EDAM, 22 (2), pp. 147-169.

Escardino, A. (2001): "La innovación tecnológica en la industria cerámica de Castellón". Bol. Soc. Esp. Ceram. V., 40, pp. 43-51.

Fasth T., (2000): "Knowledge Based Engineering for SMEs", Master's Thesis, Tekniska Universitet.

Fernández-López M., Corcho O. (2010): "Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web", Springer Publishing Company.

Fleury, M. & Reverbel, F (2003):"The JBoss Extensible Server", Proceedings of IFIP/ACM Middleware, LNCS 2672, pp. 344-373.

Fortuna B., Mladenic D. & Grobelnik M. (2011): "User modeling combining access logs, page content and semantics". 1st International Workshop on Usage Analysis and the Web of Data (USEWOD2011) based at the 20th International World Wide Web Conference (WWW 2011), Hyderabad, India.

Fraternali P., Rossi G. & Sanchez-Figueroa F. (2010): "Rich Internet Applications," IEEE Internet Computing, pp. 9-12.

Galle, P. (2008), 'Candidate worldviews for design theory', Design Studies, 29 (3), pp. 267-303.

Galle, P. (2009), 'The ontology of Gero's FBS model of designing', Design Studies, 30(4), pp. 321-339.

Garbacz, P. (2005): "Towards a standard taxonomy of artifact functions", Terminology and Content Development, GTW, pp. 251-264.

Garbacz, P. (2006): "Towards a Standard Taxonomy of Artifact Functions", Applied Ontology, 1 (3), pp. 221-236.

Garraín D., Vidal R., Abad-Kelly J., París A. (2009): "ACV libre: La utilización del ELCD en la fase de diseño", XIII Congreso Internacional de Ingeniería de Proyectos (AEIPRO).

Georgatos, F. (2002): "How applicable is Python as first computer language for teaching programming in a pre-university educational environment, from a teacher's point of view?" Master Thesis, AMSTEL Institute, Faculty of Science, Universiteit of Amsterdam.

Gero J. S. (1990): "Design Prototypes: A Knowledge Representation Schema for Design", AI magazine, 11 (4), pp. 26 - 36.

Gero J. S. & Kannengiesser U. (2002): "The Situated Function-Behavior-Structure-Framework", Artificial Intelligence in Design'02, pp. 89-104.

Gero J. S. & Kannengiesser U. (2004): "Modelling Expertise of Temporary Design Teams", Journal of Design Research, 4 (3), pp. 42-56.

Gibson, J. J. (1977): "The theory of affordances", in RE Shaw and J Bransford, Perceiving, Acting and Knowing, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 67-82.

Gobin B. A. & Subramanian K. (2010): "An OWL ontology for CommonKADS template knowledge models". International Journal of Human and Social Sciences, 5, pp. 256–261.

Golbreich C., Bierlaire O., Dameron O. & Gibaud B. (2005): "What reasoning support for ontology and rules? The brain case study" 8th International Protégé Conference, Protégé with Rules Workshop, Madrid, Spain, SMI-2005-1079.

Gómez-Pérez A., Fernández-López M. & Corcho O. (2004): "Theoretical Foundations of Ontologies - Chapter 1", Springer, ed., Ontological Engineering with Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web, London, pp. 1- 44.

Greenfield, J. & Short, K. (2004): "Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools", Wiley, Indianapolis.

Gruber T.R. (1993): Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Knowledge Systems Laboratory, Stanford University., Padova, Italy.

Hatchuel, A. & Weil, B. (2009): "C-K design theory: An advanced formulation". Research in Engineering Design, 19, pp. 181-192.

Hatchuel A., Weil B. & Le Masson P. (2013): "Towards an ontology of design: lessons from C–K design theory and Forcing", Research in Engineering Design, 24 (2), pp. 147-163.

He B. & Feng P. (2012): "Guiding conceptual design through functional space exploration", The International Journal of Advanced Manufacturing Technology, September 2012.

Hendler, J. & Berners-Lee, T. (2010): From the semantic web to social machines: A research challenge for AI on the World Wide Web. , *Artif.Intell.*174 (2), pp. 156 - 161.

Hirtz J., Stone R., McAdams D., Szykman S. & Wood K. (2002): "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts", Research in Engineering Design, 13, pp. 65-82.

Hubka V. and Eder W. E. (1988): "Theory of Technical Systems", Springer-Verlag, Berlin.

Ingarden R. (1965): "Der Streit um die Existenz der Welt", Max Niemeyer, Tubingen.

Kannengiesser U. & Gero J. S. (2012): "A Process Framework of Affordances in Design" Design Studies, 28(1), pp. 50-62.

Kaplanski P. (2011): "Controlled English Interface for knowledge bases", Studia Informatica, 32, 3 (84).

Kemsley S., (2010): Enterprise 2.0 Meets Business Process Management, Handbook on Business Process Management 1, International Handbooks Information System, Part III, pp. 565-574.

Khaidzir K. A. M. & Lawson B. (2012): "The cognitive construct of design conversation", Research in Engineering Design, October 2012.

Kim K.-Y., Manley D. G. & Yang H. (2006): "Ontology-Based Assembly Design and Information Sharing for Collaborative Product Development", Computer-Aided Design, 38 (12), pp. 1233-1250.

Kitamura Y. & Mizoguchi R., (2003): "Ontology-Based Description of Functional Design Knowledge and Its Use in a Functional Way Server", Expert Systems with Applications, 24 (2), pp. 153-166.

Kitamura Y. & Mizoguchi R. (2004): "Ontology-Based Systematization of Functional Knowledge", Journal of Engineering Design, 15 (4), pp. 327-351.

Kitamura Y. & Mizoguchi R. (2013): "Characterizing functions based on phase- and evolution-oriented models", Applied Ontology, In Press.

Kitamura Y., Washio N., Koji Y., Sasajima M., Takafuji S. & Mizoguchi R. (2006), "An Ontology-Based Annotation Framework for Representing the Functionality of Engineering Devices", *ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Pennsylvania, USA.

Klein Meyer J. S., Cabannes G., Lafon P., Troussier N., Roucoules L. & Gidel T. (2007): "Product Modeling for Design Alternatives Selection Using Optimization and Robustness Analysis.", 16th International Conference on Engineering Design, Paris, France.

Krima, S.; Barbau, R.; Fiorentini, X.; Sudarsan, R. & Sriram, R.D., (2009): OntoSTEP: OWL-DL Ontology for STEP, NIST Internal Report, NISTIR 7561.

Labarga A., Valentin F., Anderson M. & Lopez R. (2007):"Web Services at the European Bioinformatics Institute*", Nucleic Acids Res*, 35, pp. W6–W11.

J. Lee (1997): “Design rationale systems: understanding the issues”, IEEE Expert, 12, pp. 78–85.

Li, B., Chen, Y., Zhang, J., Hu, Y., (2012): “Modeling and representation of a computer-aided conceptual design system”, Journal of Mechanical Science and Technology, 26 (11), pp. 3515-3524.

Linden, G., Kraemer K., & J. Dedrick, (2007): “Who Captures Value in a Global Innovation System? The Case of Apple’s iPod” is in Personal Computer Industry Center Working Paper UC-Irvine.

Liu Y. & Bason A.H. (2007): "An Ontology Based Approach to a Flexible Aid for Mechanical Conceptual Design." Proceedings of the 16th International Conference on Engineering Design ICED 07, Paris, France.

Lovett, P., Ingram, A. & Bancroft, C. (2000): "Knowledge-Based Engineering for SMEs-a methodology", *Journal of Materials Processing Technology*, pp. 384 - 389.

Lozano Tello, A. & Gomez-Perez A. (2004): “Ontometric: A Method to Choose the Appropriate Ontology”, Journal of Database Management, 15 (2), pp. 1-18.

Maier, J. R. A. & Fadel, G. M. (2009a): “Affordance based design: A relational theory for design”, Research in Engineering Design 20(1), pp. 13-27.

Maier, J. R. A. & Fadel, G. M. (2009b): “Affordance-based design methods for innovative design, redesign and reverse engineering”, Research in Engineering Design 20(4), pp. 225-239.

Marquardt, W., Morbach, J., Wiesner, A. & Yang, A.D. (2010): “OntoCAPE - A Re-Usable Ontology for Chemical Process Engineering”, Springer, Berlin.

Masolo C., Borgo S., Gangemi A., Guarino N. & Oltramari A., (2003): “Wonderweb Deliverable D18”, Dissertation/Thesis, Laboratory for Applied Ontology - ISTC-CNR.

Mathisen, R. O. (2008): “Adoption of open source software in the software industry.” Fall 2008, Norwegian University of Science and Technology.

Matsokis A. & Kiritsis D. (2010): “An Ontology-based Approach for Product Lifecycle Management”, Computers in Industry, 61(8), pp. 787-797.

Mizoguchi R. (2003): "Tutorial on Ontological Engineering - Part 1: Introduction to Ontological Engineering", Ohm Sha & Springer, ed., *New Generation Computing*, 4, pp. 365-384.

Montecchi T. & Russo D. (2011): “FBOS: Function/Behaviour–Oriented Search”, TRIZ Future 2011, Dublin, Ireland.

Montiel-Ponsoda, E., Aguado de Cea G., Gomez-Perez A. & Peters W. (2011): “Enriching Ontologies with Multilingual Information”, *Natural Language Engineering*, Ed Cambridge University Press, 17, pp. 283-309.

Morbach, J. (2009): “A Reusable Ontology for Computer-Aided Process Engineering”. Dissertation, RWTH Aachen University.

Moreno-Sánchez, J. & García, L. A., (2007): “Desarrollo de un sistema para la gestión del conocimiento en el diseño de productos industriales”, Métodos Informáticos Avanzados. Castellón (España): Publicacions Universitat Jaume I.

Motik, B., Patel-Schneider, P. F. & Parsia, B. (eds.) (2009): “OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax”. W3C Recommendation 27 October 2009. http://www.w3.org/TR/owl2-syntax/

MySQL AB (2012): MySQL Reference Manual. MySQL AB: Cupertino, CA, USA. http://dev.mysql.com/doc/refman/5.6/en/index.html, Obtenido en 10/07/2012.

Navibot SR8855 (2012): “Robot Navibot SR8855 de Samsung “Fecha de Consulta: 12/02/2011 http://www.samsung.com/es/consumer/home-appliances/vacuumcleaner/robot/VCR8855L3B/XEE/index.idx?pagetype=prd_detail

Nieh, J., Yang, S. J. & Novik, N. (2000):“A comparison of thin-client computing architectures”. Tech. Rep. CUCS-022-00. Department of Computer Science, Columbia University.

Oberle D., Ankolekar A., Hitzler P., Cimiano P., Schmidt C., Weiten M., Loos B., Porzel R., Zorn H.-P., Micelli M., Sintek M., Kiesel M., Mougouie B., Vembu S., Baumann S., Romanelli M., Buitelaar P., Engel R., Sonntag R., Reithinger N., Burkhardt F., Zhou J. (2007): "DOLCE ergo SUMO: On Foundational and Domain Models in SWIntO (SmartWeb Integrated Ontology)", Journal of Web Semantics, Science, Services and Agents on the World Wide Web, 5, pp. 156–174.

O'Connor M. J., Knublauch H., Tu S. W. & Musen M.A. (2005): "Writing rules for the Semantic Web using SWRL and Jess" 8th International Protégé Conference, Protégé with Rules Workshop, Madrid, Spain, SMI-2005-1079.

Oreilly, T. (2007):"What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software". *Communications & Strategies*, No. 1, p. 17.

Orfali R., Harkey D. & Edwards J. (2009): "Client Server Survival Guide - Third Edition", John Wiley & Sons Inc.

Pahl G. & Beitz W. (1984): "Engineering Design", Design Council.

Penoyer, J., Burnett, G., Fawcett, D. & Liou, S. (2000): "Knowledge based product life cycle systems: Principles of integration of KBE and C3P", *Computer-Aided Design*, 32, pp. 311-320.

Pilgrim M. (2010): "HTML5: Up and Running", O'Reilly Media.

Pourmohamadi M. & Gero J. S. (2011): "Linkographer: An analysis tool to study design protocols based on FBS coding scheme", 18th International Conference on Engineering Design (ICED'11).

Preciado J. C., Linaje M., Comai S. & Sanchez-Figueroa F. (2007): "Design Rich Internet Applications with Web Engineering Methodologies", WSE 2007: 9th IEEE International Workshop on Web Site Evolution, pp. 23-30.

Protégé (2012): Protégé 3.5, Stanford University http://protégé.standford.edu.

Rashed M. G., & Ahsan R. (2012): "Python in Computational Science: Applications and Possibilities". International Journal of Computer Applications, 46(20), pp. 26-30.

Raskin V., Taylor J. M. & Hempelmann C. F. (2013): "Meaning- and ontology-based technologies for high-precision language an information-processing computational systems", Advanced Engineering Informatics, 27, pp. 4–12.

Real Academia Española. (S. f.). Función [artículo enmendado]. En Diccionario de la lengua española (avance de la 23ª ed.). Recuperado de http://lema.rae.es/drae/?val=FUNCI%C3%93N

Richards D. (2009): A social software/Web 2.0 approach to collaborative knowledge engineering Information Sciences, 179, pp. 2515–2523.

Rockwell J., Krishnamurty S., Grosse I., & Wileden J. (2010): "A Semantic Information Model for Capturing and Communicating Design Decisions," *Journal of Computing and Information Science in Engineering (JCISE)*, Special Issue on Knowledge-Based Design. 10 (3), pp. 256-263.

Roth M. (2006):"A Java EE framework for managing biometric data based on BioLANCC". Master's thesis is from University of Zurich. Dept. of Informatics.

Roy U. & Bharadwaj B. (2002): "Design with Part Behaviors: Behavior Model, Representation and Applications", Computer-Aided Design, 34 (9), pp. 613-636.

Sánchez Moreno, J. & García, L. A. (2007): "Desarrollo de un sistema para la gestión del conocimiento en el diseño de productos industriales", Métodos Informáticos Avanzados. Castellón (España): Publicacions Universitat Jaume I.

Sasajima M., Kitamura Y., Ikeda M. & Mizoguchi R. (1995): "FBRL: A Function and Behavior Representation Language", Proceedings of IJCAI, pp. 1830-1836.

Sen C., Summers J. D., & Mocko G. M. (2011): "A protocol to formalise function verbs to support conservation-based model checking", Journal of Engineering Design, pp. 765-788.

Sharma R., Stearns B., Ng T. & Dietzen S. (2002):“J2EE Connector Architecture and Enterprise Application Integration”, Addison Wesley.

Sicilia M. A., Rodríguez, D., García-Barriocanal, E. & Sánchez-Alonso, S. (2012): “Empirical Findings on Ontology Metrics”, Expert Systems Applied, Volume 39, Issue 8, pp. 6706–6711.

Singh I., Stearns B. & Johnson M. (2005):“Designing enterprise applications with the J2EE platform”, 4th Edition, Addison Wesley.

Skarka, W. (2007): "Application of MOKA methodology in generative model creation using CATIA", Engineering Applications of Artificial Intelligence, Volume 20, Issue 5, pp. 677-690.

Srinivasan V., Chakrabarti A. & Lindemann U. (2012): “A framework for describing functions in design”, Proceedings of the 12th International Design Conference DESIGN'12, Dubrovnik, Croatia.

Stokes, M. (2001):“Managing Engineering Knowledge: MOKA Methodology and Tools for Knowledge based Engineering Application”.

Storga M., Andreasen M.M. & Marjanovic D. (2005): "Towards a Formal Design Model Based on a Genetic Design Model System." Proceedings of the 15th International Conference on Engineering Design ICED 05, Melbourne, Australia.

Studer R., Benjamins V.R. & Fennel D., (1998): “Knowledge engineering: principles and methods, Data and Knowledge Engineering”, 25, pp. 161–197.

Su T. M., Qiu X. P. & Yu Y. L. (2009): “An Ontology-based Collaborative Design System”. Luo, Y. (ed.) CDVE 2009. LNCS, 5738, pp. 69-76.

Suárez-Figueroa M. C., García-Castro R., Villazón B. & Gómez-Pérez A. (2011): “Essentials in Ontology Engineering: Methodologies, Languages and Tools”, Ontological Engineering, Ontology Engineering Group, Universidad Politécnica de Madrid.

Suh, N. P. (2001): “Axiomatic Design: Advances and Applications”. New York: Oxford University Press.

Suh, N. P. (1995): “Designing in of Quality through Axiomatic Design”, IEEE Transactions on Reliability, 44, 256.

Suh, N. P. (1990): "The Principles of Design", Oxford University Press, New York.

Sun H., Houssin R., Gardoni M. & de Bauvrond F. (2013): “Integration of user behaviour and product behaviour during the design phase: Software for behavioural design approach”, International Journal of Industrial Ergonomics, 43, pp. 100-114.

Szykman S., Racz J. & Sriram R. (1999): "The Representation of Function in Computer-Based Design", Design Engineering Technical Conferences, ASME, Las Vegas, Nevada.

Taft S. T., Bloch J., Bocchino R., Burckhardt S., Chafi H., Cox R., Gaster B., Steele G. & Ungar D. (2011): “Multicore, manycore, and cloud computing: is a new programming language paradigm required?”, Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion, pp. 165 – 170.

Takeda H. (1994): Towards Multi-Aspect Design Support Systems, Technical Report NAIST-IS-TR94006, Nara Institute of Science and Technology, Nara, Japan.

Tang H.-H., Lee Y. Y., & Gero J. S. (2011): “Comparing collaborative collocated and distributed design processes in digital and traditional sketching environments: A protocol study using the function behavior structure coding scheme,” Design Studies, 32 (1), pp. 1–29.

Tessier S. & Wang Y. (2013): “Ontology-based feature mapping and verification between CAD systems”, Advanced Engineering Informatics, 27, pp. 76–92.

Tomiyama T., van Beek T. J., Alvarez Cabrera A. A., Komoto H. & D’Amelio V. (2013): “Making function modeling and reasoning practically usable”, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM), Special issue on Functional Descriptions in Engineering (accepted).

Tor S. B., Britton G. A., Zhang W. Y. & Deng Y. M., (2000): "Design Automation of Two-Stage Collapsible Core Using Design Prototype", International Journal of Computer Integrated Manufacturing, 13 (1), pp. 31-39.

Tor S. B., Britton G. A., Zhang W. Y. & Deng Y. M., (2002): "Guiding Functional Design of Mechanical Products through Rule-Based Causal Behavioral Reasoning", International Journal of Production Research, 40 (3), pp. 667-682.

Torres V. H., Ríos J., Vizán A. & Pérez J. M. (2013): "Approach to integrate product conceptual design information into a computer-aided design system", Concurrent Engineering: Research and Applications, 21(1), pp. 27–38.

Umeda Y., Takeda H., Tomiyama T. & Yoshikawa H. (1990): "Function, Behavior, and Structure", Gero J., ed., Applications of Artificial Intelligence in Engineering V, 1 Springer, Berlin, pp. 177-194.

Van Rennsen A., Vermaas P.E. & Zwart S.D. (2007): "A Taxonomy of Functions in Gellish English." Proceedings of the 16th International Conference on Engineering Design ICED '07, Paris, France.

Van Beek T. J. & Tomiyama T. (2012): "Structured workflow approach to support evolvability", Advanced Engineering Informatics, 26, pp. 487–501.

Vázquez, A. (1991): "Corrosión y Protección Metálicas", S. Feliu y M. C. Andrade (Coordinadores), Ed. Consejo de Investigaciones Científicas, Madrid, España.

Vermaas P. E. & Dorst K., (2007): "On the Conceptual Framework of John Gero's FBS-Model and the Prescriptive Aims of Design Methodology", Design Studies, 28 (2), pp. 133-157.

Vermaas, P. (2012): "On the formal impossibility of analyzing subfunctions as parts of functions in design methodology", Research in Engineering Design, Springer London, 0934-9839, pp. 1-14.

Vidal, R. & Mulet, E. (2006): "Thinking about computer systems to support design synthesis", *Communications of the ACM*, 49, pp. 100-104.

Visser, W. (2009):"Design: one, but in different forms". Design Studies, 30(3), 187-223.

Wang G., Sun B., Lu J., and Yu G., (2004): "RPE query processing and optimization techniques for XML databases". *J. Comput. Sci. Technol*., 19(2), pp. 224–237.

Wen K., Tan S., Wang J., Li R. & Gao Y. (2013): "A model based transformation paradigm for cross-language collaborations", Advanced Engineering Informatics, 27, pp. 27–37.

World Wide Web Consortium (1992): "World Wide Web: Summary" http://www.w3.org/summary.html .

World Wide Web Consortium (2004): "OWL: Web Ontology Language Overview," http://www.w3.org/TR/owl-features .

Yang D., Dong M. & Miao R., (2008): "Development of a product configuration system with an ontology-based approach," Computer-Aided Design, 40 (8), pp. 863-878.

Yao Y. G., Lin L. F., Wang F. & Zhang . W. Y. (2013): "Multi-perspective modeling: managing heterogeneous manufacturing knowledge based on ontologies and topic maps", International Journal of Production Research.

Yu R., Gu N. & Ostwald M. (2012): "Using situated FBS ontology to explore designers' patterns of behavior in parametric environments", ITcon Vol. 17, Special Issue CAAD and Innovation, pp. 271-282.

Zebra Imaging (2012): Zebra Imaging, http://www.zebraimaging.com/. Obtenido en 20/07/2012.

Zhang Y., Luo X., Li J. & Buis J. J. (2013): "A semantic representation model for design rationale of products", Advanced Engineering Informatics, 27, pp. 13–26.

Zhang W. Y., Tor, S. B. & Britton, G. A. (2002): "A Heuristic State-Space Approach to the Functional Design of Mechanical Systems", The International Journal of Advanced Manufacturing Technology, 19 (4), pp. 235-244.

Zhang Z., Liu Z., Chen Y. & Xie Y. (2012): “Knowledge flow in engineering design: An ontological framework”, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 227(4), pp. 760–770.

Glosario

***Abstracto**:* Entidad del DOLCE que no tiene cualidad/atributo. Ej. El valor del dinero.

Adobe Flash Player: Aplicación en forma de reproductor multimedia actualmente distribuido por Adobe Systems.

Adobe Flex: Término que agrupa una serie de tecnologías para dar soporte al despliegue y desarrollo de RIAs, basadas en su plataforma propietaria Flash.

Acción: Según la definición de la R.A.E. "*Efecto que causa un agente sobre algo*". En el A-QB, se acomete por un diseño, dividiéndose en función o comportamiento. Es función si es un *perdurant* eventual, dependiente del tiempo y dirigida por un APO, y comportamiento en el caso contrario, si es un *perdurant* estático, independiente del tiempo, y regido por un NAPO. Y según la ontología OntoFaBES, queda definida como un *perdurant* que, entre otras cosas, se ejecuta por un objeto físico, agente o no y está adscrito como mínimo a una estructura. Además tiene un valor en el eje de cualidad física, tiene un valor en el eje de *perdurant* y un valor en el eje de cualidad temporal.

Accomplishment* (cumplimiento*): Según el modelo A-QB, acción cuya función no es un acto instantáneo y no es acumulativo. Ej. Dar una conferencia.

Achievement* (logro*): Según el modelo A-QB, acción cuya función no es un acto instantáneo y no acumulativo. Ej. Caer un rayo.

***Activity* (Actividad):** Según la metodología MOKA, es un formulario ICARE que describe los procesos de diseño requeridos para un producto en cuestión.

***Affordance* (permiso):** Capacidad de que un instrumento pueda permitir realizar una acción a su usuario. Pueden ser de tres tipos: reflexivas, reactivas y reflectadas.

***Amount-of-matter* (cantidad de materia):** Entidad del DOLCE, es *endurant* sin unidad y mereológicamente invariante. P. ej. Oro, plata, madera o carne.

Aplicación web: Básicamente consiste en al menos dos partes, llamadas capas: un servidor, en la que los principales componentes se están ejecutando y un cliente, donde se muestra la interfaz de usuario, generalmente en un navegador web, cuyo usuario no posee control sobre los recursos, sino que es el servidor el encargado de manejarlos.

Applet: Término en inglés referido al componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

API (*Application Programming Interface*): Interfaz implementada debido a la interacción entre dos programas informáticos de la misma manera que una interfaz del usuario facilita la interacción entre humanos y ordenadores.

APO (*Agentive Physical Object*): Entidad del DOLCE que es objeto físico para el cual se puede adscribir intenciones, creencias o deseos. P. ej. Un ser humano.

A-QB (*Action "Cube"*): Evolución del B-Cube se adapta que al ámbito de las acciones del diseño de un producto, sean éstas funciones o comportamientos, para poder ser utilizado en una ontología basada en el marco FaBES.

Arquitectura: En el ámbito informático, es la estructura organizativa de un sistema de software compuesta por diferentes bloques constituyentes, sus propiedades externas y las relaciones entre sí y con el entorno. Está diseñada para que la estructura del sistema permita las funcionalidades deseadas a través del criterio de integración, evolución y mantenimiento de éste. La **arquitectura cliente/servidor** consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta y se especializa para aquellos sistemas operativos que se encuentran distribuidos a través de una red de ordenadores.

Axiomas formales: En el ámbito de las ontologías, sirven para modelar sentencias que son siempre verdad. Se utilizan normalmente para representar aquel conocimiento que no puede ser formalmente definido por los otros componentes.

Back-end: Término que se relaciona con el final de un proceso. En este texto se relaciona con la parte final de la arquitectura para el servidor Java EE.

Biblioteca de programación: Conjunto de subprogramas utilizados para desarrollar software.

B-Cube: Modelo para la representación de los comportamientos del diseño de un producto según las bases del marco FBS.

B-FES (*Behavior- Function Environment Structure*): Marco de modelado funcional basado en el esquema FBS desarrollado por Tor (2000, 2002).

Bytecode: En informática, código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

CAE *(Computer Aided Engineering)*: Es el conjunto de programas informáticos que permiten analizar y simular los diseños de ingeniería realizados con el ordenador, o creados de otro modo e introducidos en éste, para valorar sus características, propiedades, viabilidad y rentabilidad.

Capa: En las arquitecturas cliente/servidor, es un esquema abstracto de aplicaciones distribuidas genéricas que se corresponden con las funciones típicas en un sistema. La **capa EIS** (*Enterprise Information System*) se encarga del software utilizado y que incluye sistemas de infraestructura empresarial.

CAx (*Computer Aided technologies*): Término amplio que se refiere al uso de tecnologías por ordenador para ayudar en el diseño, análisis, y fabricación de productos.

Clase: En el ámbito de ontologías, también llamada concepto, en un sentido amplio de la palabra, describe conceptos de un dominio de conocimiento determinado usualmente organizada en taxonomías a través de las cuales se pueden aplicar mecanismos de inherencia. Una **clase disjunta** es aquella que no puede tener ninguna instancia en común con otra clase.

Cliente: Es una aplicación informática o un ordenador que consume un servicio remoto en otro ordenador, conocido como servidor, normalmente a través de una red de telecomunicaciones. Un **cliente ligero** es un ordenador o programa informático en una arquitectura de red cliente-servidor que depende primariamente del servidor central para las tareas de procesamiento, y principalmente se enfoca en transportar la entrada y la salida de datos entre el usuario y el servidor remoto. En contraste, un **cliente pesado** procesa la mayor cantidad de información posible y pasa solamente los datos para las comunicaciones y el almacenamiento al servidor.

Componente Java EE: Unidad autónoma de software funcional que se ensambla en una aplicación Java EE con sus clases y archivos relacionados y que se comunica con otros componentes

Comportamiento: Eje sobre el que se estructura una serie de metodologías de diseño permitiendo conectar las descripciones de la estructura física de objetos a las descripciones de sus funciones técnicas. Según el marco FaBES, se establece como la acción que relaciona la estructura de un diseño, su función y es realizada por un NAPO. Según la ontología OntoFaBES, se considera un *Perdurant*, cuya acción es realizada por un NAPO, objeto físico no agente, que, entre otras cosas, está ligado a la capa de funciones y a la capa estructural, y que también o necesitan otro comportamiento para ejecutarse o son necesarios para que se ejecute otro comportamiento.

Conocimiento: Conjunto de datos que modelan de forma estructurada la experiencia que se tiene sobre un cierto dominio o que surgen de interpretar los datos básicos.

***Constraint* (Restricción):** Según la metodología MOKA, es un formulario ICARE que está relacionado con las entidades y se emplea para capturar y almacenar conocimiento sobre limitaciones del producto o de algún paso del diseño del producto.

DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*): Metaontología que trata de englobar las diferentes categorías ontológicas subrayando el lenguaje natural y el sentido común humano.

***Endurant**:* Entidad perteneciente a la metaontología DOLCE que está presente en todo momento. Ej. Un trozo de papel. Se divide en *endurant*s físicos (*Physical Endurant*), no-físicos (*Non-physical Endurant*) y en conjuntos donde están unidos ambos: suma arbitraria (*Arbitrary Sum*).

***Energy* (energía)**: Según DOLCE, cualidad física relativa al estado energético en que se encuentra el elemento físico.

***Entity* (entidad):** Formulario de la metodología MOKA que describe objetos del dominio de conocimiento del producto que se está diseñando como pueden ser componentes, ensamblajes, partes y características.

Empresa 2.0: Es aquella que, por analogía con la web 2.0, toma la referencia de ésta en lo que se refiere al uso de herramientas de software social.

Entorno: Según el marco FaBES, se establece como una región dónde sólo cualidades físicas pueden ubicarse. Según la ontología OntoFaBES, se considera una región física que, entre otras cosas, condiciona a la estructura con una serie de restricciones basadas en un rango entre dos temperaturas, presión y volumen específico.

Especificación: En informática, una especificación es una definición formal o semiformal de un sistema informático ya sea un programa informático, un componente de un software, una librería o cualquier otro tipo de programario.

Estructura: Cada una de las partes diferenciadas, aunque vinculadas, en que puede ser dividida a efectos de su diseño. Según el marco FaBES, se establece como la representación física de un diseño. Según la ontología OntoFaBES, se considera una entidad que tiene como subclases a una estructura agente y una no agente.

Experto: En ingeniería del conocimiento, aquella persona que suministra la principal fuente de conocimiento mediante especificaciones sobre el diseño del producto.

FaBES: Marco de diseño funcional basado en el esquema FBS trasladable al ámbito ontológico. Surge como una extensión y refinamiento del modelo B-FES con el fin de poder relacionar el apartado funcional con el estructural de manera racional.

FAST: Técnica de sistema de análisis de funciones para un objeto que se pueden estructurar en función objetivo, función básica y cuatro tipos de funciones de apoyo: asegurar fiabilidad, aumentar el valor, asegurar comodidad de uso y complacer los sentidos.

FBS (*Function Behavior Structure*): Marco donde modelar y representar la funcionalidad de un sistema.

Flash: Nombre reducido de la marca comercial oficial Adobe Flash Professional. Se trata de una aplicación de creación y manipulación de gráficos vectoriales con posibilidades de manejo de código.

Formalización del conocimiento: La formalización del conocimiento consiste en la transformación de la información en conocimiento, interpretando el conjunto de datos para que explícito y gestionable por un sistema informático.

Función: Relación de entrada-salida prevista de un sistema cuyo propósito es realizar una tarea. Según el marco FaBES, consiste en la acción inherente al diseño del cual está siendo procesado, y que está intencionada o deseada por un agente inteligente. Según la ontología OntoFaBES, se considera un *Perdurant*, entidad que está presente temporalmente, cuya acción, entre otras cosas, es realizada por un APO, objeto agente físico, que está ligada a un comportamiento y que necesita otra función para ejecutarse o es necesaria para que se ejecute otra función.

Gestión del conocimiento: Consiste en el proceso de integrar la información sobre un dominio concreto, extraer el sentido de la información que se encuentra incompleta y actualizarla.

GUI (*Graphical User Interface*): Programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

Hibernate: Software gratuito de código abierto regula el mapeo de clases Java a tablas de bases de datos y en un paso posterior, los tipos de datos Java de los tipos de datos SQL.

Host (*anfitrión*): Todo equipo informático que posee una dirección IP y que se encuentra interconectado con uno o más equipos.

HTML (*Hypertext Markup Language*): Lenguaje de marcación de hipertexto creado para la elaboración de páginas web.

HTTP (*Hypertext Transfer Protocol*): Uno de los protocolos de Internet que puede ser utilizado para fines de comunicación entre *hosts.*

ICARE (*Illustration, Constraint, Activity, Rule, Entity*): En la metodología MOKA, consiste en una serie de formularios enlazados pertenecientes al modelo informal que crean un marco para almacenar las unidades de conocimiento.

***Illustration* (Ilustración):** Según la metodología MOKA, es un formulario ICARE que se emplea para completar la información sobre los otros formularios, incluir histórico o referencias a diseños similares, pruebas, explicaciones complejas, entre otros.

Inferencia de conocimiento: Es la obtención de nuevo conocimiento que a priori se desconoce desde información previa incluida en el dominio de trabajo. Para ello es necesario un lenguaje que permita el tratamiento del conocimiento.

Información: Conjunto de datos básicos sobre un dominio, sin interpretar, que se usan como entrada de un sistema.

Ingeniería Basada en el Conocimiento: Desarrollo de una estructura, a partir de la cual se puede implementar un diseño automático, haciendo uso del conocimiento experto sobre el ciclo de vida de un producto.

Ingeniería Ontológica: En informática y ciencias de la información es un campo nuevo, que estudia los métodos y metodologías para la construcción de ontologías.

Ingeniero del conocimiento: Ingeniero que organiza de forma apropiada la captura del conocimiento sobre un dominio así como propone las actividades para su formalización.

Instancia: En el ámbito de las ontologías representan objetos determinados en el dominio en el cual se trabaja. También recibe el nombre de individuo.

Invariancia mereológica: Propiedad de un elemento que indica que todos los componentes de dicho elemento se mantienen constantes de manera indefinida.

Java: Lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

Java SE: Plataforma Java, Edición Estándar (*Java Platform*, *Standard Edition*), (antes J2SE) para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.

Java EE: Plataforma Java, Edición Empresa (*Java Platform, Enterprise Edition*) (antes J2EE) orientada a entornos distribuidos empresariales o de Internet para desarrollar y ejecutar software de aplicaciones con arquitectura de *n* capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

Java ME: Plataforma Java, Edición Micro (*Java Platform, Micro Edition*), (antes J2ME) orientada a entornos de limitados recursos, como teléfonos móviles o PDAs (Asistente digital personal, del inglés *Personal Digital Assistant*).

JBoss: Es un servidor de aplicaciones Java EE de código abierto implementado en Java conocido actualmente como WildFly.

Jess: En el ámbito de los lenguajes de programación es un motor de reglas para la plataforma Java.

KBE (*Knowledge Based Engineering*): Ver *Ingeniería Basada en el Conocimiento*.

Knowledge (saber): Según el modelo B-Cube, cualidad abstracta con control consciente en la que no existen normas prefijadas.

KSS (*Knowledge Sharing System*): Arquitectura para la captura y adquisición del conocimiento de los procesos de diseño de producto que permite la integración del diseñador en la KBE.

Lenguaje natural: En la filosofía del lenguaje, es el lenguaje hablado o escrito por humanos para propósitos generales de comunicación.

Lógica de negocio: En informática, término técnico utilizado generalmente para describir los algoritmos funcionales que gestionan el intercambio de información entre una base de datos y la interfaz del usuario.

Magnitude (Magnitud): Según DOLCE, cualidad física que indica la magnitud física que queda afectada por el elemento físico.

Máquina virtual: Programa informático que emula a una computadora y puede ejecutar programas como si fuese una computadora real.

Marco FBS: Marco utilizado fundamentalmente en el ámbito del diseño industrial para modelar y representar la funcionalidad de un sistema.

Material: Según la ontología OntoFaBES, es la materia de la que está hecha una parte de la estructura.

MCRDR (*Multiple Classification Ripple Down Rules*): Es una técnica de adquisición del conocimiento que preserva los beneficios y la estrategia esencial de reglas combinadas en cadena (*Ripple Down Rules*) para el manejo de clasificaciones múltiples.

Metaontologías: Describen conceptos muy generales (p.ej. sustancia, tangible, intangible) y proveen de nociones generales sobre las cuales se enlazan todos los términos básicos en las ontologías existentes.

Middleware: Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos.

Modelo de objetos: En informática es la colección de objetos o clases por las cuales un programa puede examinar y manipular algunas partes específicas de su mundo.

MOKA (*Methodology and tools Oriented to Knowledge Acquisition*): Metodología que describe en términos de reglas, procesos, técnicas de modelado y definiciones, las etapas necesarias para la especificación de sistemas de KBE.

MySQL: Es un RDBMS que se ejecuta en un servidor permitiendo un acceso multiusuario a un número determinado de bases de datos. Archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo.

NAPO (*Non-Agentive Physical Object*): Entidad del DOLCE que es un objeto físico y no es APO. Ej. La estructura física de un objeto.

Naturaleza monotónica: En lógica, implica que agregar una información a una teoría nunca produce una reducción de su conjunto de consecuencias. Así la lógica de primer orden es de naturaleza monotónica.

Nivel: En un sistema hace referencia a la posición relativa de determinados conjuntos de elementos en su disposición en diferentes planos de organización de un sistema. En la metodología MOKA el **nivel informal** es relativamente simple y orientado para representar y formalizar el conocimiento en un lenguaje que sea entendido por los expertos sin ser especialistas en lenguajes formales. El **nivel formal** tiene como objetivo representar y almacenar el conocimiento de una forma codificada con el fin de conectarlo a nivel informático.

Objeto físico: Entidad del DOLCE que es un *endurant* con unidad. Ej. Una persona, un coche o un reloj.

OntoFaBES: Ontología que adapta el marco FBS para facilitar su modelado. Adquiere su marco formal de FaBES, tomando como referencia las nociones definidas en la metaontología DOLCE, desarrollada usando la

herramienta Protégé, escrita en el lenguaje OWL y utilizando el lenguaje SWRL para inferir las reglas de conocimiento.

Ontología: Especificación explícita de una conceptualización compartida, la cual se puede basar en una taxonomía o en axiomas que tengan en cuenta el modelado de un sistema basado en ciertas descripciones funcionales. Considerada también como una descripción explícita y formal de clases en un dominio de discurso, con propiedades de cada clase describiendo varias características y atributos de la clase, y con restricciones sobre dichas propiedades. Se pueden considerar tres tipos fundamentales:

- **Ontologías de dominio:** Son reutilizables en un dominio específico dado (ingeniería, fabricación, diseño, entre otros).
- **Ontologías de tarea:** Describen el vocabulario relacionado con una tarea específica o actividad (p.ej. diagnóstico, planificación).
- **Ontologías de aplicaciones** o dependientes de las aplicaciones: Contienen todas las definiciones necesarias para modelar el conocimiento requerido para una aplicación particular.

OntoRFB (*Ontological Reconciled Functional Basis*): Taxonomía de funciones de artefactos, basada en el modelo RFB y la metaontología DOLCE y analizada desde una perspectiva de la lógica filosófica con el objetivo de depurar las deficiencias encontradas.

ORM (*Object/Relational Mapping*): Técnica que mapea una representación de datos a partir de un modelo de objetos para obtener de esa manera un modelo de datos relacional.

OWA (*Open World Assumption*): Hipótesis proveniente del campo de la lógica formal que indica que el valor real de una sentencia es independiente de si cualquier observador o agente *sabe* que es cierto.

OWL (*Ontology Web Language*): Lenguaje de marcado para publicar y compartir datos usando ontologías en Internet.

OWL 2: Es una extensión y revisión de OWL que añade nueva funcionalidad con respecto a OWL, alguna de las nuevas características incrementan su estructura sintáctica mientras otras ofrecen nueva expresividad.

Plataforma Java: Nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java, u otros lenguajes que compilen a *bytecode*, y un conjunto de herramientas de desarrollo.

Perdurant: Entidad perteneciente a la metaontología DOLCE que está presente durante un tiempo determinado. Ej. Una acción. Se dividen en eventuales, entidades dependientes del tiempo, y estacionarios, independientes del tiempo.

Persistencia: En almacenamiento de la información/conocimiento, consiste en el proceso de guardar información/conocimiento en un modelo de datos para poder restaurarse en un momento posterior con todas sus propiedades y relaciones.

Plug-in: Aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como *plug-in* (del inglés "enchufable"), add-on (agregado), conector o extensión.

Process (*proceso*): Según el modelo A-QB, acción cuyo comportamiento es acumulativo, no tiene una finalización natural y se puede descomponer en acciones diferentes. Ej. Correr.

Protégé: Es un editor libre de código abierto de ontologías además de un sistema de adquisición de conocimiento.

PyME (Pequeña y Mediana Empresa): Es una empresa con características distintivas, y tiene dimensiones con ciertos límites ocupacionales y financieros prefijados por los Estados o regiones.

Quality (*cualidad*): Entidad perteneciente a la metaontología DOLCE que se puede percibir o medir. Ej. Una medida, un color. Se divide en cualidad temporal, física o abstracta.

Quale: Describe la posición de un atributo individual dentro de un espacio conceptual. Para su comprensión cabe distinguirse entre un *atributo* (p.ej. el color de un tablero), y su *valor* (p.ej. una tonalidad particular de marrón).

Racer Pro: En el ámbito de ontologías, es un razonador semántico desarrollado por la empresa *Racer Systems*.

Razonador semántico: En el ámbito de ontologías, es un tipo de software que permite inferir consecuencias desde un conjunto de hechos afirmados o axiomas.

RDBMS (*Relational Database Management System*): Es un sistema de gestión de bases de datos relacionales que sigue un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos.

RDF *(Resource Description Framework):* Es una familia de especificaciones de la W3C originalmente diseñado como un modelo de datos para metadatos. Ha llegado a ser usado como un método general para la descripción conceptual o modelado de la información que se implementa en los recursos web, utilizando una variedad de notaciones de sintaxis y formatos de serialización de datos.

Red cliente/servidor: Red de comunicaciones en la que todos los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados.

Red Hat: Es la compañía responsable de la creación y mantenimiento de una distribución del sistema operativo GNU/Linux que lleva el mismo nombre: Red Hat Enterprise Linux, y de otra más, Fedora.

Región física: Según DOLCE, es una región en la cual sólo pueden adherirse las cualidades físicas directamente. P. ej. Peso, módulo de Young o conductividad térmica, entre otras.

Regla de Horn: Regla basada en la cláusula de Horn, es decir, un tipo de regla con una serie de premisas, y un único consecuente. Ej.: "A es hija de B si A es mujer y B es padre de A".

Relaciones: En el ámbito de las ontologías representan un tipo de asociación entre conceptos de un dominio. Se definen formalmente como cualquier subconjunto de un producto de n conjuntos.

RFB (*Reconcilied Functional Basis*): Modelado funcional del diseño de un objeto basado en funciones y flujos perfeccionado por Szykman (1999) y Hirtz (2002).

RIA (*Rich Internet Application*): Aplicaciones web que utilizan un navegador web estandarizado para ejecutarse y por medio de complementos o mediante una máquina virtual se agregan las características adicionales disponibles.

***Rule* (regla):** Según el modelo B-Cube, cualidad abstracta con control consciente que sigue unas normas prefijadas. Según la metodología MOKA, es un formulario ICARE que tiene varios propósitos, desde generar la salida de la entrada como filtro o diagnóstico o para seleccionar las actividades determinadas a través de un proceso.

Sandbox: En el contexto de desarrollo de software o desarrollo web, entorno de pruebas que aísla los cambios en el código, fruto de la experimentación, del propio entorno de producción o entorno de edición.

Servicio web: Es una parte de un programa informático que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

Servidor: En informática, es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan hacer uso de ellos. En Internet, un **servidor web o servidor HTTP** es un ordenador que usa el protocolo http para enviar páginas web al ordenador de un usuario cuando el usuario las solicita. Y en el entorno de programación Java, un **servidor de aplicaciones** es el componente que permite el carácter distribuido de una aplicación con sus servicios.

Signal (señal): Según DOLCE, cualidad física relativa al efecto del elemento físico sobre una señal.

SaaS (*Software as a Service*): Modelo de distribución de software donde el soporte lógico y los datos que maneja se alojan en servidores de una compañía de tecnologías de información y comunicación, a los que se accede con un navegador web desde un cliente, a través de Internet.

SoA (State-of-Affairs): Es una entidad objetiva o no-mental representada por una frase. Proveniente de la filosofía, se puede traducir como *situación actual* y se define como la combinación de circunstancias aplicadas en una sociedad o grupo en un tiempo particular.

SOAP (*Simple Object Access Protocol*): Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

***Solidworks*:** Programa de diseño asistido por ordenador para modelado mecánico desarrollado por Dassault Systèmes (Francia), para el sistema operativo Microsoft Windows.

Skill (destreza): Según el modelo B-Cube, cualidad abstracta sin control consciente.

Spatial Location: Según DOLCE, cualidad física que indica la posición que ocupa el elemento físico en el espacio.

SQWRL (*Semantic Query-Enhanced Web Rule Language*): Lenguaje de reglas mejorado para la búsqueda semántica para poder aplicar SWRL en Protégé en base a Jess.

State (*estado*): Según el modelo A-QB, acción cuyo comportamiento es acumulativo, no tiene una finalización natural y no se puede descomponer en acciones diferentes. Ej. Estar sentado.

STEP (*Standard for the Exchange of Product model data*): Es un estándar internacional para la representación e intercambio de información de productos industriales.

SWRL (*Semantic Web Rule Language*): Es un lenguaje propuesto para la web semántica que puede ser usado para expresar reglas y su lógica combinándose con el lenguaje OWL.

Transmisión del conocimiento: Consiste en la comunicación del conocimiento adquirido sobre un dominio de un sistema a otro.

***Topological Connectedness* (Conexión topológica)**: Según DOLCE, cualidad física que indica el tipo de conexión topológica que cumple un elemento físico.

TULUM®: Instrumento que sirve para inspeccionar el casco de veleros sin necesidad de sacarlo del agua.

Tupla: Secuencia ordenada de objetos, esto es, una lista con un número limitado de objetos. Las tuplas se emplean para describir objetos que son capaces de ser descompuestos en un cierto número de componentes.

UDDI (*Universal Description, Discovery and Integration*): Es una plataforma independiente basada en XML que registra que negocios mundiales pueden enlazarse en Internet y además es un mecanismo para registrar y localizar aplicaciones de servicios web.

URI (*Uniform Resource Identifier*): En informática, cadena de caracteres corta que identifica inequívocamente un recurso (servicio, página, documento, entre otros) normalmente accesible en una red o sistema.

VBa (*Visual Basic for applications*): Lenguaje de macros de Microsoft *Visual Basic* utilizado en la interfaz de programación de aplicaciones de *Solidworks*.

XML (*eXtensible Markup Language*): Lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

Web Semántica: ver *Web 3.0*.

Web 2.0: Es un término que hace referencia a una segunda evolución de la Web, basada fundamentalmente en dos aspectos: web como plataforma de acceso a los distintos servicios y en la participación activa de los usuarios como fuente de contenido e información.

Web 3.0: Es una expresión que se utiliza para describir la evolución del uso y la interacción de las personas en internet a través de diferentes formas entre las que se incluyen la transformación de la red en una base de datos, un movimiento social hacia crear contenidos accesibles por múltiples aplicaciones sin navegador, el empuje de las tecnologías de inteligencia artificial, la web semántica, la Web Geoespacial o la Web 3D.

WSDL (*Web Services Description Language*): Formato XML que se utiliza para describir servicios Web.

Anexo 1: Imágenes de gráficos de comportamientos-estructura.

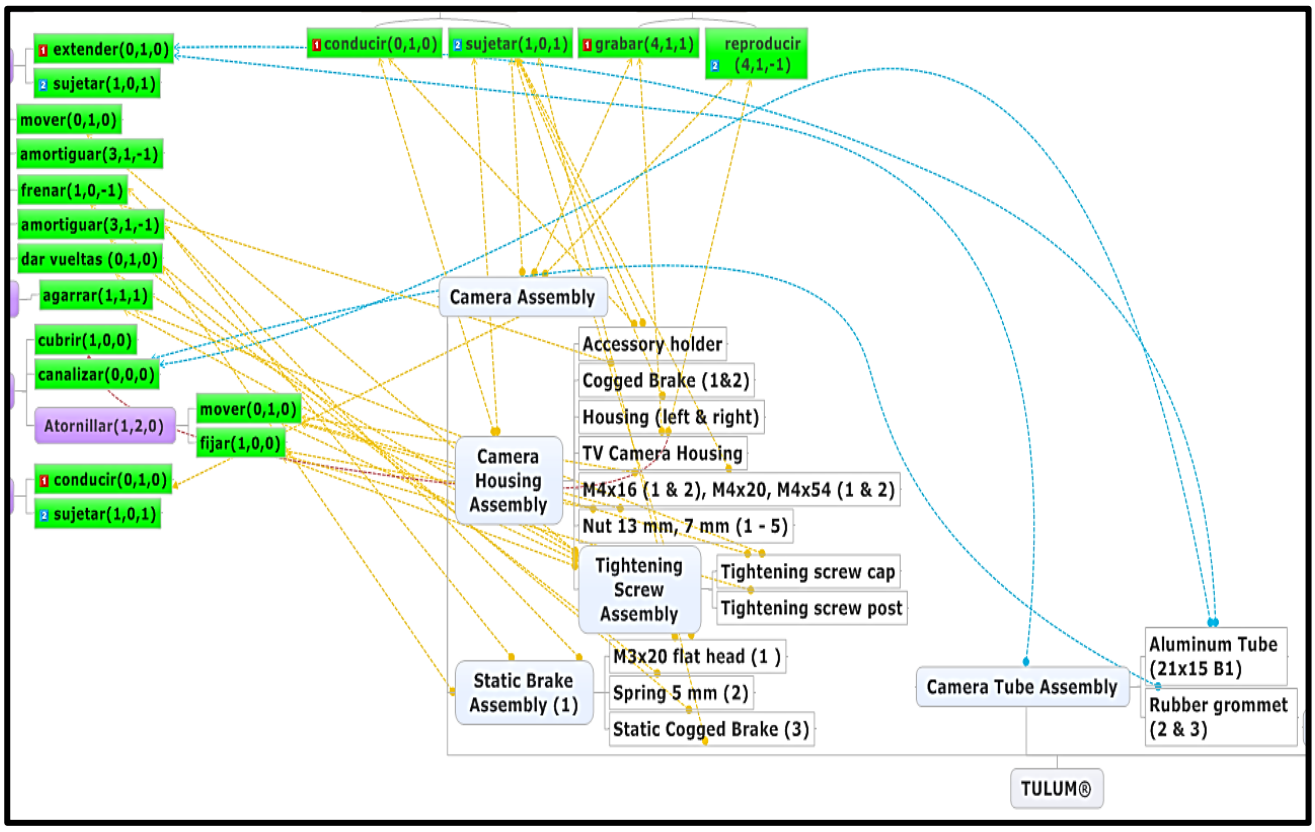


Figura 109: Camera Assembly y Camera Tube Assembly – Relación de comportamientos y estructura

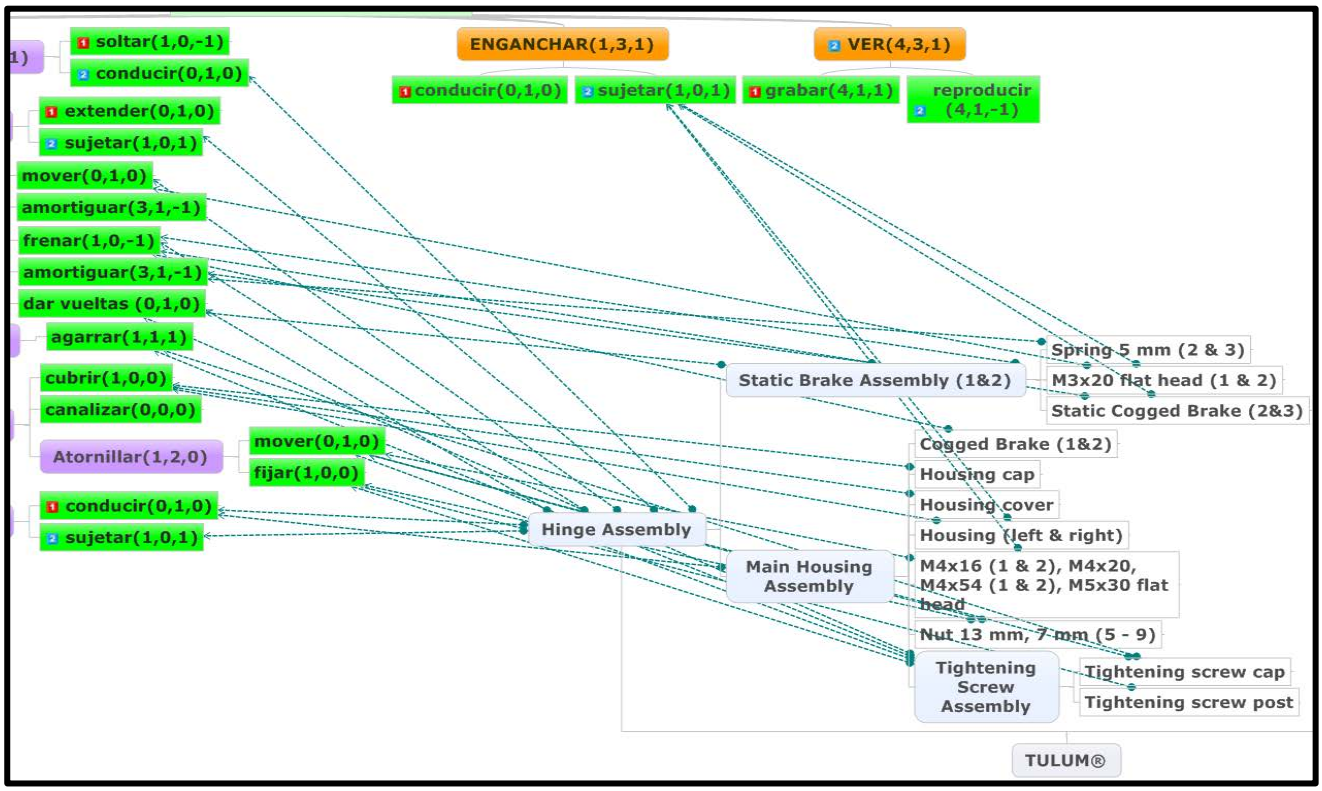


Figura 110: *Hinge Assembly* – Relación de comportamientos y estructura

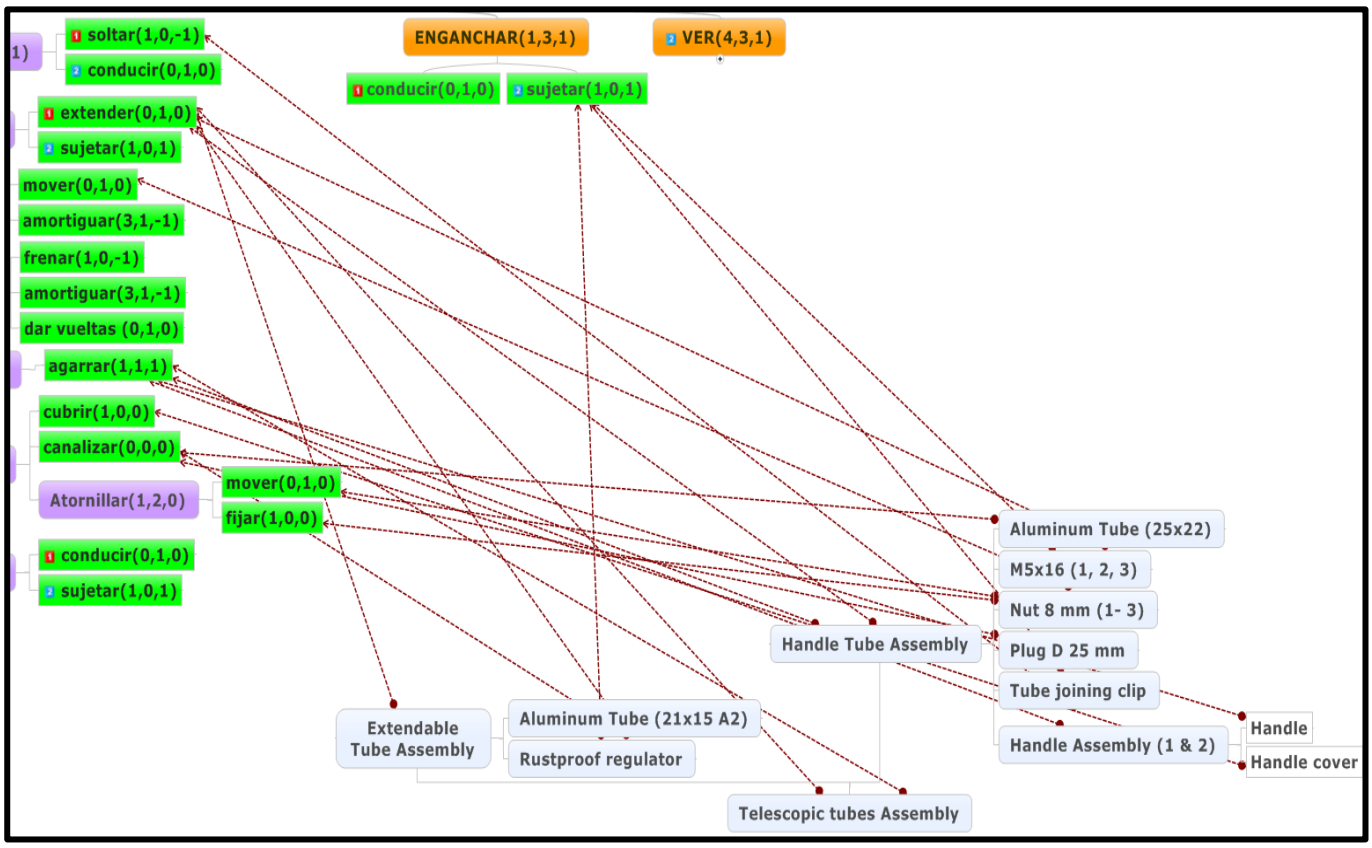


Figura 111: Telescopic Tubes Assembly – Relación de comportamientos y estructura

Anexo 2: Clase *Constraint* de OntoFaBES aplicada al TULUM®.

A2-1. INFORMACIÓN SOBRE LA CONEXIÓN DEL TULUM®[vv]

Tabla 56: Relación de los tipos de restricciones con los diferentes ensamblajes del TULUM®

ESTRUCTURA	Parallel	Distance	Lock	Coincident	Tangent	Concentric	Perpendicular
Camera Assembly	1	-	-	5	-	-	-
Camera Housing Assembly	11	2	1	41	-	-	-
Camera Tube Assembly	-	-	-	7	-	-	-
Extendable Tube Assembly	-	-	-	5	1	1	-
Handle Assembly (1 & 2)	-	-	-	12	-	-	-
Handle Tube Assembly	-	8	-	24	-	-	-
Hinge Assembly	1	-	-	6	-	-	-
Main Housing Assembly	11	-	1	44	1	1	1
Static Brake Assembly (1 & 2)	2	4	-	12	-	2	-
Telescopic tubes Assembly	-	-	-	5	-	-	-
Tightening Screw Assembly ®	-	-	-	6	-	-	-
TULUM	-	-	-	10	2	-	-

[vv] *Is Connected Info With* no se representa al ser la propiedad que relaciona la información sobre la conexión con el tipo de conexión que se establece. Sólo se representarán las variables utilizadas en los diferentes tipos de conexiones.

A2-2. TIPO DE CONEXIONES DEL TULUM®[ww]

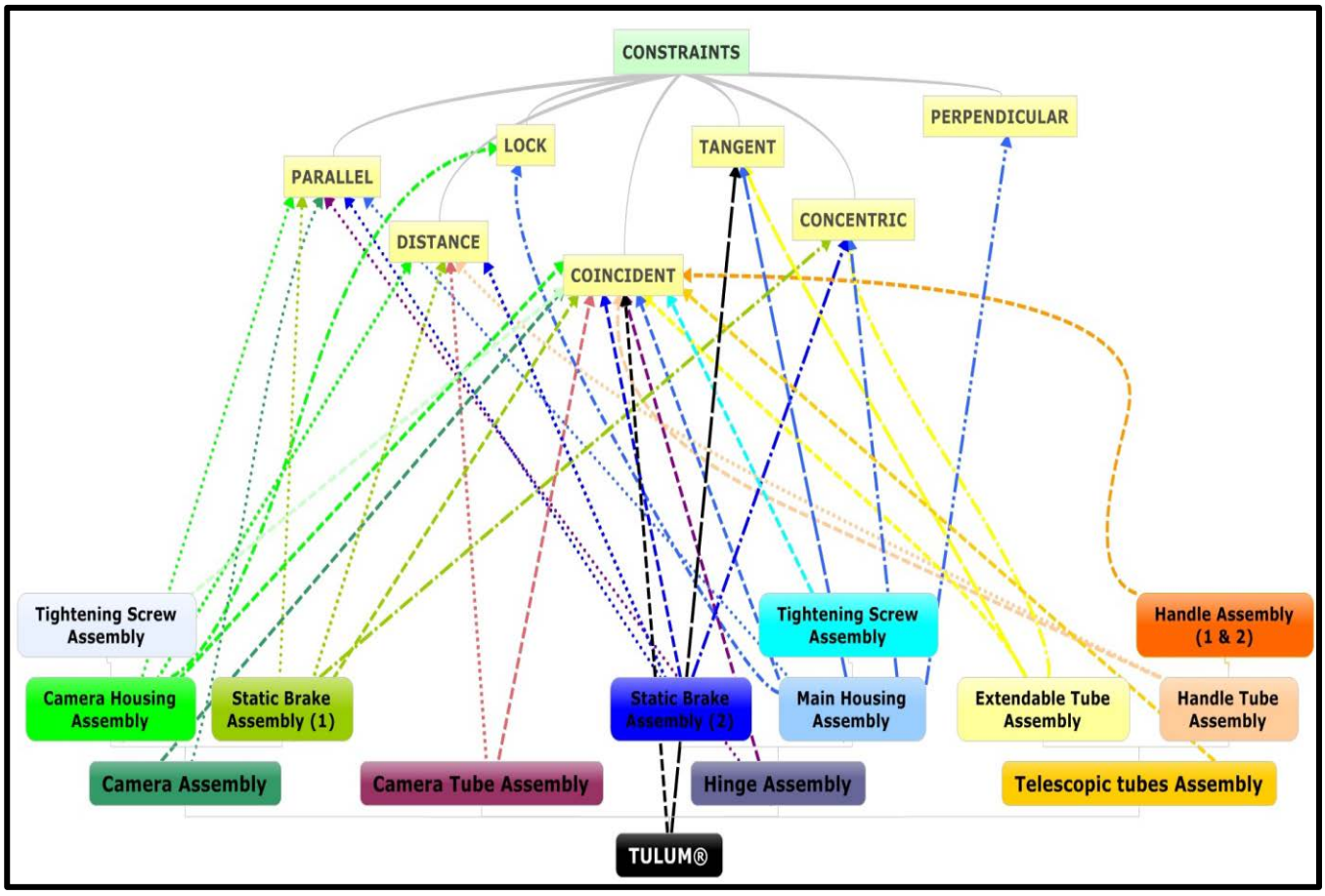


Figura 112: Relación de las restricciones de los ensamblajes del TULUM®

A2-3. CAMERA ASSEMBLY

A2-3.1. TIPO DE CONEXIÓN

Tabla 57: Tipos de conexiones del *Camera Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_1	Camera Housing Assembly 1	Camera Assembly 1	false	Aligned	Coincident
Coincident_2	Camera Housing Assembly 1	Camera Assembly 1	false	Aligned	Coincident
Coincident_3	Camera Housing Assembly 1	Camera Assembly 1	false	Aligned	Coincident
Coincident_4	Static cogged brake 3 Static cogged brake 4	Housing_right_1	false	Aligned	Coincident
Coincident_5	Tightening screw post 1	Spring_5mm_2 Spring_5mm_3	false	Anti-Aligned	Coincident
Parallel_1	TV camera housing 1	Static Brake Assembly 2	false	Aligned	Parallel

[ww] ***Is Connected Type_Structure 1*** no se representa al ser el ensamblaje obvio que se está tratando.

A2-3.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 58: Localización interna de las conexiones del *Camera Assembly*.

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_4	(91,7019590868492,-60, -3,42744199446595E-16)	-
Coincident_5	(91,7019590868491, -60, 35)	(91,7019590868492,-60, -30,1377551354924)
Parallel_1	(98,0469035484279, 4,81481078550641E-31, -1,51667018309149E-30)	(91,7019590868492, -60, -3,42744199446595E-16)

Tabla 59: Localización externa de las conexiones del Camera Assembly.

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_1	(0,0,1)	(0,0,1)
Coincident_2	(0,1,0)	(0,1,0)
Coincident_3	(1,0,0)	(1,0,0)
Coincident_4	(8,98681367883089E-18, -2,16636898317977E-17, 1)	(0,0,1)
Coincident_5	(-1,38777878078145E-17, 6,93889390390726E-18, -1)	(8,98681367883089E-18, -2,16636898317977E-17, 1)
Parallel_1	(4,93038065763133E-32, 1, 2,77555756156289E-16)	(0, 1, 2,16636898317977E-17)

A2-4. CAMERA HOUSING ASSEMBLY

A2-4.1. TIPO DE CONEXIÓN

Tabla 60: Tipos de conexiones del *Camera Housing Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Tipo de alineamiento*	*Subclase de conexión*	*Reverso*
Coincident_25	Housing_left_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_26	Housing_right_1	Cogged_brake_1	Aligned	Coincident	false
Coincident_27	Housing_right_1	Cogged_brake_1	Anti-Aligned	Coincident	false
Coincident_29	Housing_left_1	Cogged_brake_2	Anti-Aligned	Coincident	false
Coincident_31	Housing_left_1	Cogged_brake_2	Aligned	Coincident	false
Coincident_34	Housing_left_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_35	Housing_left_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_36	TV_camera_housing_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_37	TV_camera_housing_1	Camera_housing_assembly_1	Aligned	Coincident	false

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Tipo de alineamiento*	*Subclase de conexión*	*Reverso*
Coincident_3	Housing_right_1	Camera_housing_assembly_1	Closest	Coincident	false
Coincident_41	Accessory_holder_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_42	Accessory_holder_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_43	Accessory_holder_1	Camera Housing Assembly 1	Aligned	Coincident	false
Coincident_45	M4x16_1	Housing_right_1	Anti-Aligned	Coincident	false
Coincident_46	Housing_right_1	M4x16_1	Aligned	Coincident	false
Coincident_4	Housing_right_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_52	M4x16_2	Housing_right_1	Anti-Aligned	Coincident	false
Coincident_53	Housing_right_1	M4x16_2	Aligned	Coincident	false
Coincident_55	M4x20_1	Housing_right_1	Anti-Aligned	Coincident	false
Coincident_57	Housing_right_1	M4x20_1	Anti-Aligned	Coincident	false
Coincident_58	Housing_right_1	M4x20_1	Anti-Aligned	Coincident	false
Coincident_5	Housing_right_1	Camera_housing_assembly_1	Aligned	Coincident	false
Coincident_60	Housing_left_1	Nut_7mm_2	Anti-Aligned	Coincident	false
Coincident_61	M4x16_1	Nut_7mm_2	Aligned	Coincident	false
Coincident_62	Nut_7mm_1	Housing_left_1	Anti-Aligned	Coincident	false
Coincident_64	M4x16_2	Nut_7mm_1	Aligned	Coincident	false
Coincident_68	Cogged_brake_1	Nut_13mm_1	Anti-Aligned	Coincident	false
Coincident_69	Cogged_brake_1	Nut_13mm_1	Aligned	Coincident	false
Coincident_70	Housing_left_1	Nut_7mm_3	Anti-Aligned	Coincident	false
Coincident_71	M4x20_1	Nut_7mm_3	Aligned	Coincident	false
Coincident_72	M4x20_1	Nut_7mm_3	Aligned	Coincident	false
Coincident_73	M4x54_1	Camera_housing_assembly_1	Anti-Aligned	Coincident	false
Coincident_79	Nut_7mm_4	Housing_left_1	Anti-Aligned	Coincident	false
Coincident_80	M4x54_1	Nut_7mm_4	Aligned	Coincident	false
Coincident_83	Nut_7mm_5	Camera_housing_assembly_1	Anti-Aligned	Coincident	false
Coincident_85	Housing_right_1	M4x54_1	Anti-Aligned	Coincident	false
Coincident_86	M4x54_2	Housing_right_1	Anti-Aligned	Coincident	false

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Tipo de alineamiento*	*Subclase de conexión*	*Reverso*
Coincident_87	M4x54_2	Camera_housing_assembly_1	Anti-Aligned	Coincident	false
Coincident_88	Housing_right_1	Tightening_screw___post_1	Aligned	Coincident	false
Coincident_90	Tightening_screw___post_1	Nut_13mm_1	Anti-Aligned	Coincident	false
Coincident_91	Housing_left_1	Nut_7mm_5	Anti-Aligned	Coincident	false
Distance1	Housing_right_1	TV_camera_housing_1	Aligned	Distance	true
Distance2	Tightening_screw___cap_1	Camera_housing_assembly_1	Aligned	Distance	true
Lock1	Nut_7mm_2	Nut_7mm_1	Closest	Lock	false
Parallel10	Housing_left_1	Nut_7mm_5	Aligned	Parallel	false
Parallel11	Housing_right_1	M4x54_2	Aligned	Parallel	false
Parallel1	Housing_right_1	Cogged_brake_1	Aligned	Parallel	false
Parallel2	Housing_left_1	Cogged_brake_2	Anti-Aligned	Parallel	false
Parallel3	Housing_right_1	M4x16_1	Aligned	Parallel	false
Parallel4	Housing_right_1	M4x16_2	Aligned	Parallel	false
Parallel5	M4x20_1	Camera_housing_assembly_1	Aligned	Parallel	false
Parallel6	Housing_left_1	M4x16_1	Aligned	Parallel	false
Parallel7	Housing_left_1	Nut_7mm_1	Aligned	Parallel	false
Parallel8	Housing_right_1	M4x54_1	Anti-Aligned	Parallel	false
Parallel9	Nut_7mm_4	Camera_housing_assembly_1	Aligned	Parallel	false

A2-4.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 61: Localización interna de las conexiones del *Camera Housing Assembly*.

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_26	(91,7019590868492, -60, -8,33681078798466)	(91,7019590868492, -60, -13,7)
Coincident_27	(0, -5,94762334620621E-15, -20)	(91,7019590868492, -60, -20)
Coincident_29	(0, -1,98254111540207E-15, 20)	(91,7019590868492, -60,0000000000001, 20)
Coincident_31	(91,7019590868492, -60,0000000000001, -7,75039573612048)	(91,7019590868492, -60,0000000000001, 13,7)
Coincident_36	(98,0469035484279, 4,81481078550641E-31, -1,51667018309149E-30)	(0, 0, 0)
Coincident_37	(98,0469035484279, 4,81481078550641E-31, -1,51667018309149E-30)	(0, 0, 0)

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_45	(58,5, 16,2442642727335, -4,99999999999999)	(24, 23,7442642727335, -4,99999999999999)
Coincident_46	(55, 16,2442642727335, 0)	(55, 16,2442642727335, 7,00000000000001)
Coincident_52	(18,5, 16,2442642727335, -4,99999999999999)	(24, 23,7442642727335, -4,99999999999999)
Coincident_53	(15, 16,2442642727335, 0)	(15, 16,2442642727335, 7,00000000000001)
Coincident_55	(4,75514007807152, -15,8973854272188, -6)	(1,25514007807154, -15,8973854272188, -6)
Coincident_57	(1,25514007807154, -15,8973854272188, -6)	(4,75514007807152, -15,8973854272188, -6)
Coincident_58	(1,25514007807154, -15,8973854272188, -51,2161214621469)	(1,25514007807152, -15,8973854272188, 10)
Coincident_60	(52,9792740578363, 12,7442642727335, 3,99999999999999)	(50,9585481156726, 16,2442642727335, 3,99999999999999)
Coincident_61	(55, 16,2442642727335, 7,00000000000001)	(55, 16,2442642727335, 6,99999999999999)
Coincident_62	(10,9585481156726, 16,2442642727335, 3,99999999999999)	(12,9792740578363, 12,7442642727335, 3,99999999999999)
Coincident_64	(15, 16,2442642727335, 7,00000000000001)	(15, 16,2442642727335, 6,99999999999999)
Coincident_68	(91,7019590868492, -60, -20)	(91,7019590868492, -60, -27)
Coincident_69	(77,7019590868504, -45,9999999999994, -30)	(84,196405587384, -60, -30)
Coincident_70	(3,27586602023529, -19,3973854272188, 6)	(-2,78631180625584, -15,8973854272187, 6)
Coincident_71	(1,25514007807152, -15,8973854272188, 10)	(1,25514007807154, -15,8973854272187, 9)
Coincident_72	(1,25514007807152, -15,8973854272188, 10)	(1,25514007807154, -15,8973854272187, 7,5)
Coincident_73	(47,3141963576479, -29, 24)	(47,3141963576479, -29, -49,8966886979028)
Coincident_79	(43,2727444733205, -29, 15)	(49,3349222998116, -32,5, 15)
Coincident_80	(47,3141963576479, -29, 24)	(47,3141963576479, -29, 18)
Coincident_83	(87,3141963576479, -29, 21)	(87,3141963576479, -29, -56,6027854363956)
Coincident_85	(47,3141963576479, -29, -26)	(47,3141963576479, -32,5, -26)
Coincident_86	(90,8141963576479, -29, -27)	(87,3141963576479, -29, -27)
Coincident_87	(87,3141963576479, -29, 23)	(87,3141963576479, -29, -56,6027854363956)
Coincident_88	(91,7019590868492, -60, -8,33681078798466)	(91,7019590868491, -60, 35)
Coincident_90	(91,7019590868491, -60, 35)	(91,7019590868492, -60, -30)

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_91	(85,2934704154842, -32,5, 18)	(83,2727444733205, -29, 18)
Distance1	(-21,9990518029889, 19,6844859104872, 0)	(-31,9990518029889, 3,33066907387547E-15, 12)
Distance2	(91,7019590868491, -60, 35)	(0, 0, 0)
Parallel10	(0, 0, 0)	(87,3141963576479, -29, 19,5)
Parallel11	(0, 0, 0)	(87,3141963576479, -29, 23)
Parallel1	(0, 0, 0)	(91,7019590868492, -60, -20)
Parallel2	(0, 0, 0)	(91,7019590868492, -60,0000000000001, 20)
Parallel3	(24, 23,7442642727335, 4,99999999999998)	(55, 16,2442642727335, 7,00000000000001)
Parallel4	(24, 23,7442642727335, 4,99999999999998)	(15, 16,2442642727335, 7,00000000000001)
Parallel5	(1,25514007807152, -15,8973854272188, 10)	(0, 0, 0)
Parallel6	(24, 23,7442642727335, 4,99999999999998)	(55, 16,2442642727335, 7,00000000000001)
Parallel7	(24, 23,7442642727335, 4,99999999999998)	(15, 16,2442642727335, 5,49999999999999)
Parallel8	(0, 0, 0)	(47,3141963576479, -29, 24)
Parallel9	(47,3141963576479, -29, 16,5)	(0, 0, 0)

Tabla 62: Localización externa de las conexiones del *Camera Housing Assembly*.

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_25	(0, 0, 1)	(0, 0, 1)
Coincident_26	(0, 0, -1)	(0, 0, -1)
Coincident_27	(0, 0, 1)	(0, 0, -1)
Coincident_29	(0, 9,91270557701033E-17, -1)	(-1,44257685818014E-17, 2,24850120005045E-17, 1)
Coincident_31	(0, -9,91270557701033E-17, 1)	(-1,44257685818014E-17, 2,24850120005045E-17, 1)
Coincident_34	(1, 0, 0)	(1, 0, 0)
Coincident_35	(0, 1, 0)	(0, 1, 0)
Coincident_36	(4,93038065763133E-32, 1, 2,77555756156289E-16)	(0, 1, 0)
Coincident_37	(1,23259516440783E-32, 2,77555756156289E-16, 1)	(0, 0, 1)
Coincident_3	(1, 0, 0)	(1, 0, 0)

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_41	(0, 0, 1)	(0, 0, 1)
Coincident_42	(0, 1, 0)	(0, 1, 0)
Coincident_43	(1, 0, 0)	(1, 0, 0)
Coincident_45	(0, -2,08166817117217E-16, 1)	(0, 2,08166817117217E-16, -1)
Coincident_46	(0, 2,08166817117217E-16, -1)	(0, 2,08166817117217E-16, -1)
Coincident_4	(0, 0, 1)	(0, 0, 1)
Coincident_52	(0, -2,08166817117217E-16, 1)	(0, 2,08166817117217E-16, -1)
Coincident_53	(0, 2,08166817117217E-16, -1)	(0, 2,08166817117217E-16, -1)
Coincident_55	(0, 0, 1)	(0, 0, -1)
Coincident_57	(0, 0, -1)	(0, 0, 1)
Coincident_58	(0, 0, 1)	(0, 0, -1)
Coincident_5	(0, 1, 0)	(0, 1, 0)
Coincident_60	(0, 0, 1)	(0, 0, -1)
Coincident_61	(0, 2,08166817117217E-16, -1)	(0, 0, -1)
Coincident_62	(0, 0, -1)	(0, 0, 1)
Coincident_64	(0, 2,08166817117217E-16, -1)	(0, 0, -1)
Coincident_68	(0, 1, 0)	(0, -1, 0)
Coincident_69	(0, 0, -1)	(0, 0, -1)
Coincident_70	(0, 0, 1)	(0, 0, -1)
Coincident_71	(0, 0, -1)	(0, 0, -1)
Coincident_72	(0, 1, 0)	(0, 1, 0)
Coincident_73	(0, 0, -1)	(0, 0, 1)
Coincident_79	(0, 0, -1)	(0, 0, 1)
Coincident_80	(0, 0, -1)	(0, 0, -1)
Coincident_83	(0, 0, -1)	(0, 0, 1)
Coincident_85	(0, 0, -1)	(0, 0, 1)
Coincident_86	(0, 0, 1)	(0, 0, -1)
Coincident_87	(0, 0, -1)	(0, 0, 1)
Coincident_88	(0, 0, -1)	(-1,38777878078145E-17, 6,93889390390726E-

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
		18, -1)
Coincident_90	(-1,38777878078145E-17, 6,93889390390726E-18, -1)	(0, 0, 1)
Coincident_91	(0, 0, 1)	(0, 0, -1)
Distance1	(-1, -8,71859612431411E-17, 0)	(-1, -1,5313029740141E-32, 1,22464679914735E-16)
Distance2	(1,38777878078145E-17, -6,93889390390726E-18, 1)	(0, 0, 1)
Lock1	(1, 0, 0)	(1, 0, 0)
Parallel10	(0, 1, 0)	(0, 1, 0)
Parallel11	(0, 1, 0)	(0, 1, 0)
Parallel1	(0, 1, 0)	(0, 1, 0)
Parallel2	(0, 1, 0)	(0, -1, 2,24850120005045E-17)
Parallel3	(0, 1, 0)	(0, 1, 2,08166817117217E-16)
Parallel4	(0, 1, 0)	(2,73691106313441E-48, 1, 2,08166817117217E-16)
Parallel5	(0, 1, 0)	(0, 1, 0)
Parallel6	(0, 1, 0)	(0, 1, 2,08166817117217E-16)
Parallel7	(0, 1, 0)	(0, 1, 0)
Parallel8	(0, 1, 0)	(-9,71445146547012E-17, -1, 0)
Parallel9	(2,22044604925031E-16, 1, 0)	(0, 1, 0)

A2-5. CAMERA TUBE ASSEMBLY

A2-5.1. TIPO DE CONEXIÓN

Tabla 63: Tipos de conexiones del *Camera Tube Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_10	Rubber_grommet_3	Camera_tube_assembly_1	false	Aligned	Coincident
Coincident_11	Aluminium_tube_21x15_B_1	Rubber_grommet_3	false	Anti-Aligned	Coincident
Coincident_1	Aluminium_tube_21x15_B_1	Camera_tube_assembly_1	false	Aligned	Coincident
Coincident_2	Aluminium_tube_21x15_B_1	Camera_tube_assembly_1	false	Aligned	Coincident
Coincident_3	Aluminium_tube_21x15_B_1	Camera_tube_assembly_1	false	Aligned	Coincident
Coincident_7	Rubber_grommet_2	Aluminium_tube_21x15_B_1	false	Anti-Aligned	Coincident
Coincident_9	Rubber_grommet_2	Camera_tube_assembly_1	false	Aligned	Coincident
Distance_1	Aluminium_tube_21x15_B_1	Rubber_grommet_2	false	Aligned	Distance
Distance_2	Rubber_grommet_3	Aluminium_tube_21x15_B_1	false	Aligned	Distance

A2-5.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 64: Localización externa de las conexiones del *Camera Tube Assembly*.

LOCALIZACIÓN DE LA CONEXIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_10	(0,0,1)	(0,0,1)
Coincident_11	(0, 1, 6,12323399573677E-17)	(0, -1, 0)
Coincident_1	(1, 0, 0)	(1, 0, 0)
Coincident_2	(0, 1, 0)	(0, 1, 0)
Coincident_3	(0, 0, 1)	(0, 0, 1)
Coincident_7	(0, -1, 0)	(0, 1, 6,12323399573677E-17)
Coincident_9	(0, 0, 1)	(0, 0, 1)
Distance_1	(0, 1, 0)	(0, 1, 0)
Distance_2	(0, 1, 0)	(0, 1, 0)

Tabla 65: Localización interna de las conexiones del Camera Assembly

LOCALIZACIÓN DE LA CONEXIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_10	(670, 5, -1,18784747643693E-15)	(0, 0, 0)
Coincident_11	(670, 8,59599015223808, -1,16586978280334E-16)	(670, 12,5, -1,18784747643693E-15)
Coincident_7	(-670, 12,5, -6,42939569552361E-16)	(-670, 8,59599015223808, -1,16586978280334E-16)
Coincident_9	(-670, 5, -6,42939569552361E-16)	(0, 0, 0)
Distance_1	(0, 0, 0)	(-670, 5, -6,42939569552361E-16)
Distance_2	(670, 5, -1,18784747643693E-15)	(0, 0, 0)

A2-6. EXTENDABLE TUBE ASSEMBLY

A2-6.1. TIPO DE CONEXIÓN

Tabla 66: Tipos de conexiones del *Extendable Tube Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_5	Aluminium_tube_21x15_A_2	Extendable_tube_assembly_1	false	Closest	Coincident
Coincident_6	Aluminium_tube_21x15_A_2	Extendable_tube_assembly_1	false	Aligned	Coincident
Coincident_7	Aluminium_tube_21x15_A_2	Extendable_tube_assembly_1	false	Aligned	Coincident
Coincident_8	Aluminium_tube_21x15_A_2	Rustproof_regulator_1	false	Anti-Aligned	Coincident
Coincident_9	Rustproof_regulator_1	Extendable_tube_assembly_1	false	Aligned	Coincident
Concentric_1	Aluminium_tube_21x15_A_2	Rustproof_regulator_1	false	Anti-Aligned	Concentric
Tangent_1	Rustproof_regulator_1	Aluminium_tube_21x15_A_2	false	Aligned	Gear

A2-6.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 67: Localización externa de las conexiones del *Extendable Tube Assembly*.

LOCALIZACIÓN	*Radio Externo 1 (mm.)*	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_5	-	(1, 0, 0)	(1, 0, 0)
Coincident_6	-	(1, 0, 0)	(1, 0, 0)
Coincident_7	-	(0, 1, 0)	(0, 1, 0)
Coincident_8	-	(0, 1, 6,12323399573677E-17)	(-2,6983081567733E-17, -1, -6,12323399573677E-17)
Coincident_9	-	(-8,62691680194517E-18, -6,12323399573677E-17, 1)	(0, 0, 1)
Concentric_1	3,50000000000006	(0, 1, 6,12323399573677E-17)	(-5,39797156626221E-17, -1, -6,12323399573677E-17)
Tangent_1	-	(2,6983081567733E-17, 1, 6,12323399573677E-17)	(-1, 0, 0)

Tabla 68: Localización interna de las conexiones del *Extendable Tube Assembly*

LOCALIZACIÓN	*Radio Interno 1 (mm.)*	*Coordenada Interna 1*	*Radio Interno 2 (mm.)*	*Coordenada Interna 2*
Coincident_8	-	(-724, 8,59599015223808, -1,16586978280334E-16)	-	(-724, 16,3619714992187, 3,58942231660551E-16)
Coincident_9	-	(-707,94773997607, -0,248552488532394, -5,19677508317803E-16)		(0, 0, 0)

LOCALIZACIÓN	*Radio Interno 1 (mm.)*	*Coordenada Interna 1*	*Radio Interno 2 (mm.)*	*Coordenada Interna 2*
Concentric_1	3,50000000000006	(-724, 10,5, 0)	3	(-724, 15,0000000000001, 2,75545529808152E-16)
Tangent_1	-	(-690,460851868876, 9,50000000000009, 3)	9,5	(750, 0, 0)

A2-7. HANDLE ASSEMBLY (1&2)

A2-7.1. TIPO DE CONEXIÓN

Tabla 69: Tipos de conexiones del *Handle Assembly (1&2)*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
1Coincident_1	Handle_cover_1	Handle_assembly_1	false	Aligned	Coincident
1Coincident_2	Handle_1	Handle_cover_1	false	Aligned	Coincident
1Coincident_4	Handle_1	Handle_cover_1	false	Aligned	Coincident
1Coincident_6	Handle_1	Handle_assembly_1	false	Aligned	Coincident
1Coincident_7	Handle_1	Handle_assembly_1	false	Aligned	Coincident
1Coincident_8	Handle_1	Handle_assembly_1	false	Aligned	Coincident
2Coincident_1	Handle_cover_1	Handle_assembly_2	false	Aligned	Coincident
2Coincident_2	Handle_1	Handle_cover_1	false	Aligned	Coincident
2Coincident_4	Handle_1	Handle_cover_1	false	Aligned	Coincident
2Coincident_6	Handle_1	Handle_assembly_2	false	Aligned	Coincident
2Coincident_7	Handle_1	Handle_assembly_2	false	Aligned	Coincident
2Coincident_8	Handle_1	Handle_assembly_2	false	Aligned	Coincident

A2-7.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 70: Localización externa de las conexiones del *Handle Assembly (1&2).*

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
1Coincident_1	(0,0,1)	(0,0,1)
1Coincident_2	(1, 0, 0)	(1, 0, 0)
1Coincident_4	(0, 1, 0)	(0, 1, 0)
1Coincident_6	(0, 1, 0)	(0, 1, 0)
1Coincident_7	(0, 0, 1)	(0, 0, 1)
1Coincident_8	(1, 0, 0)	(1, 0, 0)
2Coincident_1	(0,0,1)	(0,0,1)
2Coincident_2	(1, 0, 0)	(1, 0, 0)
2Coincident_4	(0, 1, 0)	(0, 1, 0)
2Coincident_6	(0, 1, 0)	(0, 1, 0)
2Coincident_7	(0, 0, 1)	(0, 0, 1)
2Coincident_8	(1, 0, 0)	(1, 0, 0)

Tabla 71: Localización interna de las conexiones del *Handle Assembly (1&2)*

LOCALIZACIÓN DE LA CONEXIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
1Coincident_2	(127,5, 0, 0)	(127,5, 0, 0)
2Coincident_2	(127,5, 0, 0)	(127,5, 0, 0)

A2-8. HANDLE TUBE ASSEMBLY

A2-8.1. TIPO DE CONEXIÓN

Tabla 72: Tipos de conexiones del *Handle Tube Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_10	Aluminium_tube_25x22_1	Handle_assembly_1	false	Anti-Aligned	Coincident
Coincident_12	Handle_assembly_1	Handle_tube_assembly_1	false	Aligned	Coincident
Coincident_13	Aluminium_tube_25x22_1	Handle_assembly_2	false	Anti-Aligned	Coincident
Coincident_14	Handle_assembly_2	Handle_tube_assembly_1	false	Aligned	Coincident
Coincident_16	Aluminium_tube_25x22_1	Tube_joining_clip_1	false	Anti-Aligned	Coincident
Coincident_17	Tube_joining_clip_1	Handle_tube_assembly_1	false	Aligned	Coincident
Coincident_18	Handle_1	M5x16_1	false	Aligned	Coincident
Coincident_1	Aluminium_tube_25x22_1	Handle_tube_assembly_1	false	Aligned	Coincident
Coincident_20	M5x16_2	Handle_1	false	Anti-Aligned	Coincident
Coincident_21	Handle_1	M5x16_2	false	Aligned	Coincident
Coincident_23	M5x16_3	Tube_joining_clip_1	false	Aligned	Coincident
Coincident_24	Tube_joining_clip_1	M5x16_3	false	Aligned	Coincident
Coincident_26	Handle_1	M5x16_1	false	Anti-Aligned	Coincident
Coincident_27	M5x16_1	Nut_8mm_1	false	Anti-Aligned	Coincident
Coincident_2	Aluminium_tube_25x22_1	Handle_tube_assembly_1	false	Aligned	Coincident
Coincident_31	Handle_1	Nut_8mm_1	false	Aligned	Coincident
Coincident_35	M5x16_3	Nut_8mm_2	false	Anti-Aligned	Coincident
Coincident_36	Tube_joining_clip_1	Nut_8mm_2	false	Aligned	Coincident
Coincident_38	M5x16_2	Nut_8mm_3	false	Anti-Aligned	Coincident
Coincident_3	Aluminium_tube_25x22_1	Handle_tube_assembly_1	false	Aligned	Coincident
Coincident_40	Handle_1	Nut_8mm_3	false	Anti-Aligned	Coincident
Coincident_7	Plug_D25mm_2	Aluminium_tube_25x22_1	false	Anti-Aligned	Coincident
Coincident_8	Aluminium_tube_25x22_1	Plug_D25mm_2	false	Aligned	Coincident
Coincident_9	Plug_D25mm_2	Handle_tube_assembly_1	false	Aligned	Coincident
Distance_10	M5x16_3	Nut_8mm_2	true	Aligned	Distance

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Distance_11	M5x16_2	Nut_8mm_3	true	Aligned	Distance
Distance_2	Handle_1	Plug_D25mm_2	true	Aligned	Distance
Distance_3	Handle_1	Handle_1	true	Anti-Aligned	Distance
Distance_4	Handle_1	Tube_joining_clip_1	true	Anti-Aligned	Distance
Distance_5	M5x16_1	Handle_assembly_2	true	Aligned	Distance
Distance_6	M5x16_2	Handle_assembly_1	true	Aligned	Distance
Distance_7	M5x16_3	Tube_joining_clip_1	true	Aligned	Distance

A2-8.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 73: Localización externa de las conexiones del *Handle Tube Assembly*.

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_10	(1, 0, 0)	(-1, 0, 0)
Coincident_12	(0, 0, 1)	(0, 0, 1)
Coincident_13	(1, 0, 0)	(-1, 0, 0)
Coincident_14	(0, 0, 1)	(0, 0, 1)
Coincident_16	(1, 0, 0)	(-1, 0, 0)
Coincident_17	(0, 0, 1)	(0, 0, 1)
Coincident_18	(0,0,1)	(0,0,1)
Coincident_1	(0, 0, 1)	(0, 0, 1)
Coincident_20	(1, -3,33066907387547E-16, 0)	(-1, 0, 0)
Coincident_21	(0, 0, 1)	(0, 0, 1)
Coincident_23	(0, 0, 1)	(0, 0, 1)
Coincident_24	(1, 0, 0)	(1, 0, 0)
Coincident_26	(-1, 0, 0)	(1, 0, 0)
Coincident_27	(0, 0, 1)	(0, 0, -1)
Coincident_2	(1, 0, 0)	(1, 0, 0)
Coincident_31	(1, 0, 0)	(1, 0, 0)
Coincident_35	(0, 0, 1)	(-2,55407518051738E-18, -2,37339834966123E-20, -1)
Coincident_36	(1, 0, 0)	(1, 0, -2,55407518051738E-18)

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_38	(0, 0, 1)	(0, 0, -1)
Coincident_3	(0, 1, 0)	(0, 1, 0)
Coincident_40	(-1, 0, 0)	(1, 0, 0)
Coincident_7	(-1, 4,76191101376413E-17, 1,66533453693773E-16)	(1, 0, 0)
Coincident_8	(1, 0, 0)	(1, -4,76191101376413E-17, -1,66533453693773E-16)
Coincident_9	(4,76191101376413E-17, 1, 0)	(0, 1, 0)
Distance_10	(0, 0, -1)	(-2,55407518051738E-18, -2,37339834966123E-20, -1)
Distance_11	(0, 0, -1)	(0, 0, -1)
Distance_2	(1, 0, 0)	(1, -4,76191101376413E-17, -1,66533453693773E-16)
Distance_3	(1, 0, 0)	(-1, 0, 0)
Distance_4	(-1, 0, 0)	(1, 0, 0)
Distance_5	(0, 0, 1)	(0, 0, 1)
Distance_6	(0, 0, 1)	(0, 0, 1)
Distance_7	(0, 0, 1)	(0, 0, 1)

Tabla 74: Localización interna de las conexiones del *Handle Tube Assembly*.

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_10	(-825, 0, 0)	(748, 0, 0)
Coincident_12	(718, 0, 0)	(0, 0, 0)
Coincident_13	(-825, 0, 0)	(258, 0, 0)
Coincident_14	(228, 0, 0)	(0, 0, 0)
Coincident_16	(-825, 0, 0)	(-2,50000000000002, 0, 0)
Coincident_17	(-19,5, 0, 0)	(0, 0, 0)
Coincident_18	(228, -27,5, 1)	(228, -27,5, -7,5)
Coincident_20	(718, -27,4999999999998, -7,50000000000001)	(718, 0, 0)
Coincident_21	(718, -27,5, 1)	(718, -27,4999999999998, -7,50000000000001)
Coincident_23	(-19,5, 23,8484800354236, -8)	(-19,5, 23,8484800354236, 1)
Coincident_24	(-19,5, 0, 0)	(-19,5, 23,8484800354236, -8)
Coincident_26	(228, 0, 0)	(228, -27,5, -7,5)

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_27	(228, -27,5, -7,5)	(228, -27,5, -4,50000000000001)
Coincident_31	(225,690598923241, -23,5, -8,50000000000001)	(225,690598923241, -23,5, -8,50000000000001)
Coincident_35	(-19,5, 23,8484800354236, -8)	(-19,5, 23,8484800354236, -3,5)
Coincident_36	(-19,5, 0, 0)	(-19,5, 23,8484800354236, -5,5)
Coincident_38	(718, -27,4999999999998, -7,50000000000001)	(718, -27,4999999999998, -4,50000000000001)
Coincident_40	(718, 0, 0)	(718, -27,4999999999998, -6,50000000000001)
Coincident_7	(750, 2,88889491658085E-31, -12,5)	(750, 0, 0)
Coincident_8	(-825, 0, 0)	(740, 4,76191101376414E-16, 1,66533453693774E-15)
Coincident_9	(740, 4,76191101376414E-16, 1,66533453693774E-15)	(0, 0, 0)
Distance_10	(-17, 23,8484800354236, -8)	(-24,118802153517, 23,8484800354237, -7,5)
Distance_11	(720,5, -27,4999999999998, -7,50000000000001)	(713,381197846483, -27,4999999999998, -8,50000000000001)
Distance_2	(748, 0, 0)	(752, -9,52382202752824E-17, -9,5)
Distance_3	(258, 0, 0)	(688, 0, 0)
Distance_4	(198, 0, 0)	(-2,00000000000002, 0, 0)
Distance_5	(228, -27,5, -7,5)	(228, 0, 0)
Distance_6	(718, -27,4999999999998, -7,50000000000001)	(718, 0, 0)
Distance_7	(-19,5, 23,8484800354236, -8)	(-19,5, 0, 0)

A2-9. HINGE ASSEMBLY

A2-9.1. TIPO DE CONEXIÓN

Tabla 75: Tipos de conexiones del *Hingle Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_1	Housing_right_1	Hinge_assembly_1	false	Aligned	Coincident
Coincident_2	Main_housing_assembly_1	Hinge_assembly_1	false	Aligned	Coincident
Coincident_3	Main_housing_assembly_1	Hinge_assembly_1	false	Aligned	Coincident
Coincident_5	Main_housing_assembly_1	Static_cogged_brake_3 Static_cogged_brake_4	false	Aligned	Coincident
Coincident_6	Tightening_screw___post_1	Spring_5mm_2 Spring_5mm_3	false	Anti-Aligned	Coincident
Parallel_1	Main_housing_assembly_1	Static_cogged_brake_3 Static_cogged_brake_4	false	Aligned	Parallel

A2-9.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 76: Localización externa de las conexiones del *Hingle Assembly*.

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_1	(0,0,1)	(0,0,1)
Coincident_2	(0, 1, 0)	(0, 1, 0)
Coincident_3	(1, 0, 0)	(1, 0, 0)
Coincident_5	(0, 0, 1)	(1,13031089997104E-17, -2,22396217324619E-17, 1)
Coincident_6	(2,08166817117217E-17, -2,7755575615629E-17, -1)	(1,13031089997104E-17, -2,22396217324619E-17, 1)
Parallel_1	(1, 0, 0)	(1, -1,11022302462516E-16, -1,13031089997104E-17)

Tabla 77: Localización interna de las conexiones del *Hingle Assembly*.

LOCALIZACIÓN DE LA CONEXIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_5	(0, 0, 0)	(91,7019590868491, -60, 0)
Coincident_6	(91,7019590868489, -59,9999999999997, 34,4532277122771)	(91,7019590868491, -60, -30,1377551354924)
Parallel_1	(0, 0, 0)	(0,701959086849127, -60, 1,02858291897365E-15)

A2-10.MAIN HOUSING ASSEMBLY

A2-10.1. TIPO DE CONEXIÓN

Tabla 78: Tipos de conexiones del *Main Housing Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_105	Cogged_brake_2	Tightening_screw___cap_1	false	Aligned	Coincident
Coincident_106	Cogged_brake_1	Nut_13mm_1	false	Aligned	Coincident
Coincident_107	Tightening_screw___post_1	Nut_13mm_1	false	Anti-Aligned	Coincident
Coincident_25	Housing_left_1	Nut_13mm_1	false	Aligned	Coincident
Coincident_26	Housing_right_1	Cogged_brake_1	false	Aligned	Coincident
Coincident_27	Housing_right_1	Cogged_brake_1	false	Anti-Aligned	Coincident
Coincident_29	Housing_left_1	Cogged_brake_2	false	Anti-Aligned	Coincident
Coincident_31	Housing_left_1	Cogged_brake_2	false	Aligned	Coincident
Coincident_34	Housing_left_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_35	Housing_left_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_3	Housing_right_1	Main_housing_assembly_1	false	Closest	Coincident
Coincident_45	M4x16_1	Housing_right_1	false	Anti-Aligned	Coincident
Coincident_46	Housing_right_1	M4x16_1	false	Aligned	Coincident
Coincident_4	Housing_right_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_52	M4x16_2	Housing_right_1	false	Anti-Aligned	Coincident
Coincident_53	Housing_right_1	M4x16_2	false	Aligned	Coincident
Coincident_55	M4x20_1	Housing_right_1	false	Anti-Aligned	Coincident
Coincident_57	Housing_right_1	M4x20_1	false	Anti-Aligned	Coincident
Coincident_58	Housing_right_1	M4x20_1	false	Anti-Aligned	Coincident
Coincident_5	Housing_right_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_60	Housing_left_1	Nut_7mm_7	false	Anti-Aligned	Coincident
Coincident_61	M4x16_1	Nut_7mm_7	false	Aligned	Coincident
Coincident_62	Nut_7mm_6	Housing_left_1	false	Anti-Aligned	Coincident
Coincident_64	M4x16_2	Nut_7mm_6	false	Aligned	Coincident
Coincident_68	Cogged_brake_1	Nut_13mm_1	false	Anti-Aligned	Coincident

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_69	Cogged_brake_1	Nut_13mm_1	false	Aligned	Coincident
Coincident_70	Housing_left_1	Nut_7mm_8	false	Anti-Aligned	Coincident
Coincident_71	M4x20_1	Nut_7mm_8	false	Aligned	Coincident
Coincident_72	M4x20_1	Nut_7mm_8	false	Aligned	Coincident
Coincident_73	M4x54_1	Main_housing_assembly_1	false	Anti-Aligned	Coincident
Coincident_79	Housing_left_1	Nut_7mm_9	false	Anti-Aligned	Coincident
Coincident_80	M4x54_1	Nut_7mm_9	false	Aligned	Coincident
Coincident_82	Housing_left_1	Nut_7mm_5	false	Anti-Aligned	Coincident
Coincident_83	Nut_7mm_5	Main_housing_assembly_1	false	Anti-Aligned	Coincident
Coincident_85	Housing_right_1	M4x54_1	false	Anti-Aligned	Coincident
Coincident_86	Housing_right_1	M4x54_2	false	Anti-Aligned	Coincident
Coincident_87	M4x54_2	Main_housing_assembly_1	false	Anti-Aligned	Coincident
Coincident_88	Housing_cap_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_89	Housing_cap_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_90	Housing_cap_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_91	Housing_cover_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_92	Housing_cover_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_93	Housing_cover_1	Main_housing_assembly_1	false	Aligned	Coincident
Coincident_94	Housing_cap_1	M5x30_flat_head_1	false	Anti-Aligned	Coincident
Concentric_1	M5x30_flat_head_1	Housing_cap_1	false	Aligned	Concentric
Lock_1	Nut_7mm_7	Nut_7mm_6	false	Closest	Gear
Parallel_10	Housing_left_1	Nut_7mm_5	false	Aligned	Parallel
Parallel_11	Housing_right_1	M4x54_2	false	Aligned	Parallel
Parallel_1	Housing_right_1	Cogged_brake_1	false	Aligned	Parallel
Parallel_2	Housing_left_1	Cogged_brake_2	false	Anti-Aligned	Parallel
Parallel_3	Housing_right_1	M4x16_1	false	Aligned	Parallel
Parallel_4	Housing_right_1	M4x16_2	false	Aligned	Parallel
Parallel_5	M4x20_1	Main_housing_assembly_1	false	Aligned	Parallel

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Parallel_6	Housing_left_1	M4x16_1	false	Aligned	Parallel
Parallel_7	Housing_left_1	Nut_7mm_6	false	Aligned	Parallel
Parallel_8	Housing_right_1	M4x54_1	false	Anti-Aligned	Parallel
Parallel_9	Nut_7mm_9	Main_housing_assembly_1	false	Aligned	Parallel
Perpendicular1	M5x30_flat_head_1	Main_housing_assembly_1	false	Closest	Perpendicular
Tangent_1	Housing_cap_1	M5x30_flat_head_1	false	Aligned	Gear

A2-10.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 79: Localización externa de las conexiones del *Main Housing Assembly.*

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_105	(-1,44257685818014E-17, 2,24850120005045E-17, 1)	(-2,08166817117217E-17, 2,7755575615629E-17, 1)
Coincident_106	(0, 0, -1)	(0, 0, -1)
Coincident_107	(2,08166817117217E-17, -2,7755575615629E-17, -1)	(0, 0, 1)
Coincident_25	(0, 0, 1)	(0, 0, 1)
Coincident_26	(0, 0, -1)	(0, 0, -1)
Coincident_27	(0, 0, 1)	(0, 0, -1)
Coincident_29	(0, 9,91270557701033E-17, -1)	(-1,44257685818014E-17, 2,24850120005045E-17, 1)
Coincident_31	(0, -9,91270557701033E-17, 1)	(-1,44257685818014E-17, 2,24850120005045E-17, 1)
Coincident_34	(1, 0, 0)	(1, 0, 0)
Coincident_35	(0, 1, 0)	(0, 1, 0)
Coincident_3	(1, 0, 0)	(1, 0, 0)
Coincident_45	(0, -2,08166817117217E-16, 1)	(0, 2,08166817117217E-16, -1)
Coincident_46	(0, 2,08166817117217E-16, -1)	(0, 2,08166817117217E-16, -1)
Coincident_4	(0, 0, 1)	(0, 0, 1)
Coincident_52	(0, -2,08166817117217E-16, 1)	(0, 2,08166817117217E-16, -1)
Coincident_53	(0, 2,08166817117217E-16, -1)	(0, 2,08166817117217E-16, -1)
Coincident_55	(0, 0, 1)	(0, 0, -1)
Coincident_57	(0, 0, -1)	(0, 0, 1)

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_58	(0, 0, 1)	(0, 0, -1)
Coincident_5	(0, 1, 0)	(0, 1, 0)
Coincident_60	(0, 0, 1)	(0, 0, -1)
Coincident_61	(0, 2,08166817117217E-16, -1)	(0, 0, -1)
Coincident_62	(0, 0, -1)	(0, 0, 1)
Coincident_64	(0, 2,08166817117217E-16, -1)	(0, 0, -1)
Coincident_68	(0, 1, 0)	(0, -1, 0)
Coincident_69	(0, 0, -1)	(0, 0, -1)
Coincident_70	(0, 0, 1)	(0, 0, -1)
Coincident_71	(0, 0, -1)	(0, 0, -1)
Coincident_72	(0, 1, 0)	(0, 1, 0)
Coincident_73	(0, 0, -1)	(0, 0, 1)
Coincident_79	(0, 0, 1)	(0, 0, -1)
Coincident_80	(0, 0, -1)	(0, 0, -1)
Coincident_82	(0, 0, 1)	(0, 0, -1)
Coincident_83	(0, 0, -1)	(0, 0, 1)
Coincident_85	(0, 0, -1)	(0, 0, 1)
Coincident_86	(0, 0, -1)	(0, 0, 1)
Coincident_87	(0, 0, -1)	(0, 0, 1)
Coincident_88	(0, 0, 1)	(0, 0, 1)
Coincident_89	(0, 0, 1)	(0, 0, 1)
Coincident_90	(1, 0, 0)	(1, 0, 0)
Coincident_91	(1, 0, 0)	(1, 0, 0)
Coincident_92	(0, 0, 1)	(0, 0, 1)
Coincident_93	(0, 1, 0)	(0, 1, 0)
Coincident_94	(7,01633115693011E-02, 0,997535518019098, 5,76379831362939E-09)	(-7,01633115693011E-02, -0,997535518019097, -5,76379832750717E-09)
Lock_1	(1, 0, 0)	(1, 0, 0)
Parallel_10	(0, 1, 0)	(0, 1, 0)

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Parallel_11	(0, 1, 0)	(0, 1, 0)
Parallel_1	(0, 1, 0)	(0, 1, 0)
Parallel_2	(0, 1, 0)	(0, -1, 2,24850120005045E-17)
Parallel_3	(0, 1, 0)	(0, 1, 2,08166817117217E-16)
Parallel_4	(0, 1, 0)	(2,73691106313441E-48, 1, 2,08166817117217E-16)
Parallel_5	(0, 1, 0)	(2,73691106313441E-48, 1, 2,08166817117217E-16)
Parallel_6	(0, 1, 0)	(0, 1, 2,08166817117217E-16)
Parallel_7	(0, 1, 0)	(0, 1, 0)
Parallel_8	(0, 1, 0)	(-9,71445146547012E-17, -1, 0)
Parallel_9	(2,22044604925031E-16, 1, 0)	(0, 1, 0)
Perpendicular1	(1,66533453693773E-16, -5,77803822322809E-09, 1)	(1, 0, 0)
Tangent_1	(1, 0, 0)	(-7,01633115693011E-02, -0,997535518019097, -5,76379832750717E-09)

Tabla 80: Localización interna de las conexiones del *Main Housing Assembly*.

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_105	(91,7019590868492, -60,0000000000001, 13,7)	(91,7019590868489, -59,9999999999997, 33,1156991315141)
Coincident_106	(77,7019590868504, -45,9999999999994, -30)	(84,196405587384, -60, -30)
Coincident_107	(91,7019590868489, -59,9999999999997, 34,4532277122771)	(91,7019590868492, -60, -30)
Coincident_26	(91,7019590868492, -60, -8,33681078798466)	(91,7019590868492, -60, -13,7)
Coincident_27	(0, -5,94762334620621E-15, -20)	(91,7019590868492, -60, -20)
Coincident_29	(0, -1,98254111540207E-15, 20)	(91,7019590868492, -60,0000000000001, 20)
Coincident_31	(91,7019590868492, -60,0000000000001, -7,75039573612048)	(91,7019590868492, -60,0000000000001, 13,7)
Coincident_45	(58,5, 16,2442642727335, -4,99999999999999)	(24, 23,7442642727335, -4,99999999999999)
Coincident_46	(55, 16,2442642727335, 0)	(55, 16,2442642727335, 7,00000000000001)
Coincident_52	(18,5, 16,2442642727335, -4,99999999999999)	(24, 23,7442642727335, -4,99999999999999)
Coincident_53	(15, 16,2442642727335, 0)	(15, 16,2442642727335, 7,00000000000001)

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_55	(4,75514007807152, -15,8973854272188, -6)	(1,25514007807154, -15,8973854272188, -6)
Coincident_57	(1,25514007807154, -15,8973854272188, -6)	(4,75514007807152, -15,8973854272188, -6)
Coincident_58	(1,25514007807154, -15,8973854272188, -51,2161214621469)	(1,25514007807152, -15,8973854272188, 10)
Coincident_60	(52,9792740578363, 12,7442642727335, 3,99999999999999)	(50,9585481156726, 16,2442642727335, 3,99999999999999)
Coincident_61	(55, 16,2442642727335, 7,00000000000001)	(55, 16,2442642727335, 6,99999999999999)
Coincident_62	(10,9585481156726, 16,2442642727335, 3,99999999999999)	(12,9792740578363, 12,7442642727335, 3,99999999999999)
Coincident_64	(15, 16,2442642727335, 7,00000000000001)	(15, 16,2442642727335, 6,99999999999999)
Coincident_68	(91,7019590868492, -60, -20)	(91,7019590868492, -60, -27)
Coincident_69	(77,7019590868504, -45,9999999999994, -30)	(84,196405587384, -60, -30)
Coincident_70	(3,27586602023529, -19,3973854272188, 6)	(-2,78631180625584, -15,8973854272187, 6)
Coincident_71	(1,25514007807152, -15,8973854272188, 10)	(1,25514007807154, -15,8973854272187, 9)
Coincident_72	(1,25514007807152, -15,8973854272188, 10)	(1,25514007807154, -15,8973854272187, 7,5)
Coincident_73	(47,3141963576479, -29, 24)	(47,3141963576479, -29, -49,8966886979028)
Coincident_79	(49,3349222998116, -32,5, 15)	(43,2727444733205, -28,9999999999999, 15)
Coincident_80	(47,3141963576479, -29, 24)	(47,3141963576479, -28,9999999999999, 18)
Coincident_82	(85,2934704154842, -32,5, 18)	(83,2727444733205, -29, 18)
Coincident_83	(87,3141963576479, -29, 21)	(87,3141963576479, -29, -56,6027854363956)
Coincident_85	(47,3141963576479, -29, -26)	(47,3141963576479, -32,5, -26)
Coincident_86	(87,3141963576479, -29, -27)	(90,8141963576479, -29, -27)
Coincident_87	(87,3141963576479, -29, 23)	(87,3141963576479, -29, -56,6027854363956)
Coincident_94	(15,4657087018032, -35,2538541868868, -4,38911336231851E-08)	(18,104899347079, 2,26841134765843, 1,72913948784381E-07)
Parallel_10	(0, 0, 0)	(87,3141963576479, -29, 19,5)
Parallel_11	(0, 0, 0)	(87,3141963576479, -29, 23)
Parallel_1	(0, 0, 0)	(91,7019590868492, -60, -20)
Parallel_2	(0, 0, 0)	(91,7019590868492, -60,0000000000001, 20)
Parallel_3	(24, 23,7442642727335,	(55, 16,2442642727335, 7,00000000000001)

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
	4,99999999999998)	
Parallel_4	(24, 23,7442642727335, 4,99999999999998)	(15, 16,2442642727335, 7,00000000000001)
Parallel_5	(1,25514007807152, -15,8973854272188, 10)	(0, 0, 0)
Parallel_6	(24, 23,7442642727335, 4,99999999999998)	(55, 16,2442642727335, 7,00000000000001)
Parallel_7	(24, 23,7442642727335, 4,99999999999998)	(15, 16,2442642727335, 5,49999999999999)
Parallel_8	(0, 0, 0)	(47,3141963576479, -29, 24)
Parallel_9	(47,3141963576479, -28,9999999999999, 16,5)	(0, 0, 0)
Perpendicular1	(18,104899347079, 2,26841134765843, 1,72913948784381E-07)	(0, 0, 0)
Tangent_1	(-35,669803821529, 21,5668377081877, -1,02694154418544E-10)	(16, -27,6576541929145, -1,04083407524633E-15)

Tabla 81: Localización externa de la conexión concéntrica del *Main Housing Assembly.*

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Concentric_1	(7,01633115693011E-02, 0,997535518019097, 5,76379832750717E-09)	(7,01633115693011E-02, 0,997535518019097, 5,76379831362939E-09)

Tabla 82: Localización interna de la conexión concéntrica del *Main Housing Assembly.*

LOCALIZACIÓN	*Radio Interno 1 (mm.)*	*Coordenada Interna 1*	*Radio Interno 2 (mm.)*	*Coordenada Interna 2*
Concentric_1	694,73827619670	(16,2104899347079, -24,6650476388572, 1,72913939416875E-08)	785,39816339744	(16, -27,6576541929145, -5,62421038806474E-24)

A2-11.STATIC BRAKE ASSEMBLY (1&2)

A2-11.1. TIPO DE CONEXIÓN

Tabla 83: Tipos de conexiones del *Static Brake Assembly (1&2).*

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
1Coincident_10	Spring_5mm_2	Static_cogged_brake_3	false	Anti-Aligned	Coincident
1Coincident_11	M3x20_flat_head_1	Static_brake_assembly_2	false	Anti-Aligned	Coincident
1Coincident_2	Static_cogged_brake_3	Static_brake_assembly_2	false	Aligned	Coincident
1Coincident_3	Static_brake_assembly_2	Static_cogged_brake_3	false	Aligned	Coincident
1Coincident_4	Static_cogged_brake_3	Static_brake_assembly_2	false	Aligned	Coincident
1Coincident_9	Static_cogged_brake_3	Spring_5mm_2	false	Anti-Aligned	Coincident
1Concentric_1	Static_cogged_brake_3	M3x20_flat_head_1	false	Aligned	Concentric
1Distance_1	Spring_5mm_2	Static_cogged_brake_3	false	Anti-Aligned	Distance
1Distance_2	M3x20_flat_head_1	Static_cogged_brake_3	false	Aligned	Distance
1Parallel_1	M3x20_flat_head_1	Static_brake_assembly_2	false	Closest	Parallel
2Coincident_10	Spring_5mm_3	Static_brake_assembly_2	false	Anti-Aligned	Coincident
2Coincident_11	M3x20_flat_head_2	Static_brake_assembly_2	false	Anti-Aligned	Coincident
2Coincident_2	Static_cogged_brake_4	Static_brake_assembly_2	false	Aligned	Coincident
2Coincident_3	Static_brake_assembly_2	Static_cogged_brake_4	false	Aligned	Coincident
2Coincident_4	Static_cogged_brake_4	Static_brake_assembly_2	false	Aligned	Coincident
2Coincident_9	Static_cogged_brake_4	Spring_5mm_3	false	Anti-Aligned	Coincident
2Concentric_1	Static_cogged_brake_4	M3x20_flat_head_2	false	Aligned	Concentric
2Distance_1	Spring_5mm_3	Static_cogged_brake_4	false	Anti-Aligned	Distance
2Parallel_1	M3x20_flat_head_2	Static_brake_assembly_2	false	Aligned	Parallel

A2-11.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 84: Localización externa de las conexiones del *Static Brake Assembly (1&2).*

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_10	(-1, 0, 2,77555756156289E-17)	(1, 0, 0)
Coincident_11	(-2,77555756156289E-17, 0, -1)	(0, 0, 1)
Coincident_2	(0, 0, 1)	(0, 0, 1)
Coincident_3	(1, 0, 0)	(1, 0, 0)
Coincident_4	(0, 1, 0)	(0, 1, 0)
Coincident_9	(0, -1, 0)	(0, 1, 0)
Distance_1	(0, -1, 0)	(0, 1, 0)
Distance_2	(0, 1, 0)	(0, 1, 0)
Parallel_1	(0, 1, 0)	(0, 1, 0)

Tabla 85: Localización interna de las conexiones del *Static Brake Assembly (1&2).*

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_10	(0, -22,62, 0)	(0, 0, 0)
Coincident_11	(-48, -6, 0)	(0, 0, 0)
Coincident_9	(0, 18,7, 0)	(0, -30,1377551354924, 0)
Distance_1	(0, 4, 0)	(0, 0, 0)
Distance_2	(-48, -6, 0)	(0, 0, 0)
Parallel_1	(-48, -6, 0)	(0, 0, 0)

Tabla 86: Localización externa de la conexión concéntrica del *Static Brake Assembly (1&2).*

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Concentric_1	(0, -1, -6,12323399573677E-17)	(0, -1, 0)

Tabla 87: Localización interna de la conexión concéntrica del *Static Brake Assembly (1&2).*

LOCALIZACIÓN	*Radio Interno 1 (mm.)*	*Coordenada Interna 1*	*Radio Interno 2 (mm.)*	*Coordenada Interna 2*
Concentric_1	785,39816339745	(-48, 14, 1,06221177613038E-31)	643,501108793285	(-48, 12, 0)

A2-12.TELESCOPIC TUBES ASSEMBLY

A2-12.1. TIPO DE CONEXIÓN

Tabla 88: Tipos de conexiones del *Telescopic Tubes Assembly.*

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_1	Handle_tube_assembly_1	Telescopic_tubes_assembly_2	false	Aligned	Coincident
Coincident_2	Handle_tube_assembly_1	Telescopic_tubes_assembly_2	false	Aligned	Coincident
Coincident_3	Handle_tube_assembly_1	Telescopic_tubes_assembly_2	false	Aligned	Coincident
Coincident_4	Aluminium_tube_25x22_1	Rustproof_regulator_1	false	Anti-Aligned	Coincident
Coincident_7	Aluminium_tube_25x22_1	Aluminium_tube_21x15_A_2	false	Aligned	Coincident

A2-12.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 89: Localización externa de las conexiones del *Telescopic Tubes Assembly.*

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_1	(0,0,1)	(0,0,1)
Coincident_2	(0, 1, 0)	(0, 1, 0)
Coincident_3	(1, 0, 0)	(1, 0, 0)
Coincident_4	(0, 1, 6,12323399573677E-17)	(5,39797156626221E-17, -1, -8,91219437293006E-17)
Coincident_7	(1, 0, 0)	(1, -1,73419523766487E-17, -1,23259516440783E-32)

Tabla 90: Localización interna de las conexiones del Telescopic Tubes Assembly.

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_4	(162, 10,2158215272058, -1,39865592769434E-16)	(162, 16,3619714992187, -8,25249251020524E-15)
Coincident_7	(-825, 0, 0)	(-1312, 2,57593919233319E-14, -1,03536427830114E-14)

A2-13.TIGHTENING SCREW ASSEMBLY

A2-13.1. TIPO DE CONEXIÓN

Tabla 91: Tipos de conexiones del *Tightening Screw Assembly*.

TIPO DE CONEXIÓN	*Tipo de estructura conectada -1*	*Tipo de estructura conectada -2*	*Reverso*	*Tipo de alineamiento*	*Subclase de conexión*
Coincident_1	Tightening_screw___cap_1	Tightening_screw_assembly_2	false	Aligned	Coincident
Coincident_2				Aligned	
Coincident_3				Aligned	
Coincident_4		Tightening_screw___post_1		Anti-Aligned	
Coincident_5				Anti-Aligned	
Coincident_6	Tightening_screw___post_1	Tightening_screw_assembly_2		Aligned	

A2-13.2. INFORMACIÓN SOBRE LA CONEXIÓN

Tabla 92: Localización externa de las conexiones del *Tightening Screw Assembly*.

LOCALIZACIÓN	*Coordenada Externa 1*	*Coordenada Externa 2*
Coincident_1	(0,0,1)	(0,0,1)
Coincident_3	(0, 1, 0)	(0, 1, 0)
Coincident_2	(1, 0, 0)	(1, 0, 0)
Coincident_4	(1, 0, 0)	(-1, 0, 0)
Coincident_5	(-1, 0, 0)	(1, 0, 0)
Coincident_6	(0, 0, 1)	(0, 0, 1)

Tabla 93: Localización interna de las conexiones del *Tightening Screw Assembly*.

LOCALIZACIÓN	*Coordenada Interna 1*	*Coordenada Interna 2*
Coincident_4	(-1,33752858076303, 0, 0)	(0, 0, 0)
Coincident_5	(0, 12, 0)	(0, 0, 0)
Coincident_6	(-64, 0, 0)	(0, 0, 0)

A2-14.TULUM

A2-14.1. TIPO DE CONEXIÓN

Tabla 94: Tipos de conexiones existentes en el *TULUM®*.

TIPO DE CONEXIÓN	*Is Connected Type Structure 1*	*Is Connected Type Structure 2*	*Can Be flipped*	*Has Alignment Type*	*Has Type*
Coincident_10	Telescopic_tubes_assembly_2	Tulum	false	Aligned	Coincident
Coincident_12	Hinge_assembly_1	Camera_tube_assembly_1	false	Anti-Aligned	Coincident
Coincident_13	Hinge_assembly_1	Camera_tube_assembly_1	false	Anti-Aligned	Coincident
Coincident_14	Telescopic_tubes_assembly_2	Camera_assembly_1	false	Aligned	Coincident
Coincident_1	Telescopic_tubes_assembly_2	Tulum	false	Aligned	Coincident
Coincident_2	Telescopic_tubes_assembly_2	Tulum	false	Aligned	Coincident
Coincident_3	Telescopic_tubes_assembly_2	Tulum	false	Aligned	Coincident
Coincident_5	Hinge_assembly_1	Telescopic_tubes_assembly_2	false	Aligned	Coincident
Coincident_6	Telescopic_tubes_assembly_2	Hinge_assembly_1	false	Aligned	Coincident
Coincident_7	Hinge_assembly_1	Camera_tube_assembly_1	false	Aligned	Coincident
Tangent_1	Telescopic_tubes_assembly_2	Hinge_assembly_1	false	Anti-Aligned	Tangent
Tangent_2	Camera_tube_assembly_1	Camera_assembly_1	false	Anti-Aligned	Tangent

A2-14.2. INFORMACIÓN SOBRE LA CONEXIÓN

El sistema no ha recogido la información sobre la localización de las conexiones del TULUM®.

Anexo 3: Tabla de propiedades de la clases *Ensamblaje* y *Parte*.

Partiendo de la lista de las propiedades de dichas clases (Tabla 16) en la Tabla 95 se incluye a continuación las propiedades de Objeto y de tipo de datos de la clase Non_Agent_Structure y si son declaradas o inferidas.

Tabla 95: Propiedades de Objeto y de tipo de datos de la clase Non_Agent_Structure

Propiedad de Objeto[xx]	Tipo de propiedad
hasAction	INFERIDA
hasBehavior	DECLARADA
hasFunction	INFERIDA
has_Environment	INFERIDA
hasMaterial	DECLARADA
isAssembledBy	DECLARADA
isPartOf	DECLARADA
hasConnectionType	DECLARADA [ii]
_hasConnectionType_Structure1	DECLARADA [ii]
_hasConnectionType_Structure2	DECLARADA [yy]
isConnectedTo	INFERIDA
_hasAngleWith	INFERIDA
_hasDistanceTo	INFERIDA
_isCamFollowerTo	INFERIDA
_isCoincidentTo	INFERIDA
_isConcentricTo	INFERIDA
_isGearTo	INFERIDA
_isParallelTo	INFERIDA
_isPerpendicularTo	INFERIDA
_isSymmetricTo	INFERIDA
_isTangentTo	INFERIDA
_isUnknownTo	INFERIDA

Propiedad de tipo de datos	Tipo de propiedad
hasMass	DECLARADA

[xx] Ya incluida en el apartado 5.3.5 las propiedades de objeto hasBehavior, isAssembledBy, isPartOf, hasMass, hasMaterial)

[yy] Estas propiedades están declaradas en el apartado 2 al ser información relativa al tipo de conexión.

A3-1. PROPIEDADES DECLARADAS

Tabla 96: Instancias de la clase *Assembly* del TULUM®

Nombre de la instancia	Propiedades			
	hasBehavior	**isAssembledBy**	**isPartOf**	**hasMass**
Camera_assembly_1	Conducir, Grabar, Reproducir, Sujetar	*Camera_housing_assembly_1, Static_brake_assembly_1*	Tulum___english	605,6 gr.
Camera_housing_assembly_1	Conducir, Sujetar	*Accessory_holder_1, Cogged_brake_1, Cogged_brake_2, Housing_left_1, Housing_right_1, M4x16_1, M4x16_2, M4x20_1, M4x54_1, M4x54_2, Nut_13mm_1, Nut_7mm_1, Nut_7mm_2, Nut_7mm_3, Nut_7mm_4, Nut_7mm_5, TV_camera_housing_1, Tightening_screw_assembly_2*	*Camera_assembly_1*	372,8 gr.
Camera_tube_assembly_1	Extender	Aluminium_tube_21x15_B_1, Rubber_grommet_2, Rubber_grommet_3	Tulum___english	261,8 gr.
Extendable_tube_assembly_1	Extender	Aluminium_tube_21x15_A_2, Rustproof_regulator_1	*Telescopic_tubes_assembly_2*	249,2 gr.
Handle_assembly_1 & 2)	Agarrar	Handle_1, Handle_cover_1	*Handle_tube_assembly_1*	83,7 gr.
Handle_tube_assembly_1	Agarrar, Extender	Aluminium_tube_25x22_1, Handle_assembly_1, Handle_assembly_2, M5x16_1, M5x16_2, M5x16_3, Nut_8mm_1, Nut_8mm_2, Nut_8mm_3, Plug_D25mm_2, Tube_joining_clip_1	*Telescopic_tubes_assembly_2*	676,8 gr.
Hinge_assembly_1	Conducir, Dar_vueltas, Frenar, Mover, Sujetar	Main_housing_assembly_1, Static_brake_assembly_1, Static_brake_assembly_2	Tulum___english	482,2 gr.
Main_housing_assembly_1	Conducir	Cogged_brake_1, Cogged_brake_2, Housing_cap_1, Housing_cover_1, Housing_left_1, Housing_right_1, M4x16_1, M4x16_2, M4x20_1, M4x54_1, M4x54_2, M5x30_flat_head_1, Nut_13mm_1, Nut_7mm_5, Nut_7mm_6, Nut_7mm_7, Nut_7mm_8, Nut_7mm_9, Tightening_screw_assembly_2	*Hinge_assembly_1*	249,5 gr.
Static_brake	Amortiguar,	M3x20_flat_head_1, Spring_5mm_2,	*Camera_assembl*	232,7 gr.

Nombre de la instancia	Propiedades			
	hasBehavior	**isAssembledBy**	**isPartOf**	**hasMass**
_assembly_1	Dar_vueltas, Frenar	Static_cogged_brake_3	*y_1, Hinge_assembly_1*	
Static_brake_assembly_2	Amortiguar, Dar_vueltas, Frenar	M3x20_flat_head_2, Spring_5mm_3 Static_cogged_brake_4	*Hinge_assembly_1*	232,7 gr.
Telescopic_tubes_assembly_2	Agarrar, Extender	*Extendable_tube_assembly_1, Handle_tube_assembly_1*	Tulum___english	926,0 gr.
Tightening_screw_assembly_2	Agarrar, Dar_vueltas, Fijar, Mover	*Tightening_screw___cap_1, Tightening_screw___post_1*	*Camera_housing_assembly_1, Main_housing_assembly_1*	38,3 gr.
Tulum___english	Agarrar, Conducir, Grabar, Reproducir, Sujetar	Camera_assembly_1, Camera_tube_assembly_1, Hinge_assembly_1, Telescopic_tubes_assembly_2	-	2275,6 gr.

Tabla 97: Instancias de la clase Part del TULUM®

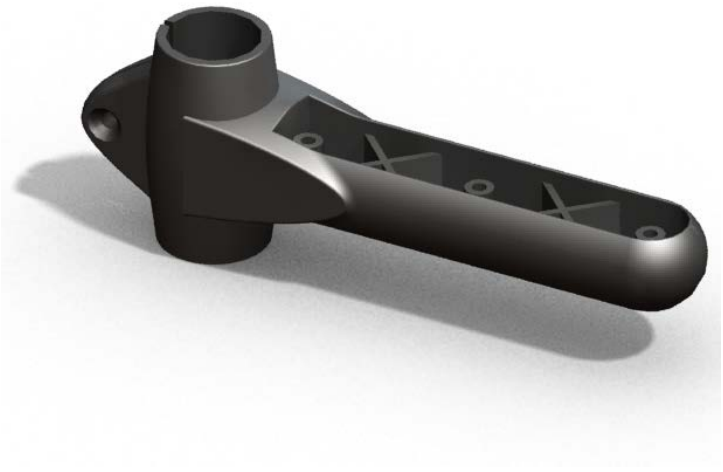
Nombre de la instancia	Propiedades			
	hasBehavior	**isPartOf**	**hasMaterial**	**hasMass**
Accessory_holder_1	Conducir, Sujetar	*Camera_housing_assembly_1,*	ABS_I	39,34 gr.
Aluminum Tube (21x15 A2, B1 & 25x22)	Canalizar, Extender	*Camera_tube_assembly_1, Extendable_tube_assembly_1, Handle_tube_assembly_1,*	_6060_Alloy	247,5 gr.
Cogged Brake (1&2)	Frenar	*Camera_housing_assembly_1, Main_housing_assembly_1*	ABS_I	10,095 gr.
Handle_1	Agarrar	*Handle_ assembly_1, Handle_ assembly_2*	ABS_I	75,12 gr.
Handle_cover_1	Cubrir	*Handle_ assembly_1, Handle_ assembly_2*	ABS_I	83,7 gr.
Housing cap	Cubrir	*Main_housing_assembly_1*	ABS_I	18,137 gr.
Housing cover	Cubrir	*Main_housing_assembly_1*	ABS_I	16,048 gr.
Housing (left & right)	Cubrir, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	ABS_I	63,345 gr. 62,634 gr.
M3x20 flat head (1 & 2),	Mover, Sujetar	*Static_brake_assembly_1, Static_brake_assembly_2*	X10Cr13__mart_410_	1,207 gr.

Nombre de la instancia	Propiedades			
	hasBehavior	**isPartOf**	**hasMaterial**	**hasMass**
M4x54 (1 & 2),	Mover, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	5,843 gr.
M5x16 (1,2,3),	Mover, Sujetar	*Handle_tube_assembly_1*	X10Cr13__mart_410_	2,934 gr.
M4x16 (1 & 2),	Mover, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	2,166 gr.
M4x20,	Mover, Sujetar	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	2,553 gr.
M5x30 flat head	Mover, Sujetar	*Main_housing_assembly_1*	X10Cr13__mart_410_	5,02 gr.
Nut 7 mm	Fijar, Mover	*Camera_housing_assembly_1,*	X10Cr13__mart_410_	0,817 gr.
Nut 13 mm	Fijar, Mover	*Camera_housing_assembly_1, Main_housing_assembly_1*	X10Cr13__mart_410_	4,984 gr.
Nut 8 mm	Fijar, Mover	*Camera_housing_assembly_1,*	X10Cr13__mart_410_	1,32 gr.
Plug D25mm	Canalizar	*Handle_tube_assembly_1*	SBR	12,888 gr.
Rubber_grommet(2 & 3)	Canalizar	*Camera_tube_assembly_1*	SBR	7,118 gr.
Rustproof regulator	Sujetar	*Extendable_tube_assembly_1*	_6060_Alloy	1,845 gr.
Spring 5 mm (2 & 3)	Amortiguar	*Static_brake_assembly_1, Static_brake_assembly_2*	_50CrV4	167,44 gr.
Static Cogged Brake	Frenar, Sujetar	*Static_brake_assembly_1, Static_brake_assembly_2*	ABS_I	64,054 gr.
TV Camera Housing	Grabar, Reproducir	*Camera_housing_assembly_1*	_6060_Alloy	123,165 gr.
Tightening screw cap	Agarrar, Mover	*Tightening_screw_assembly_2*	ABS_I	17,634 gr.
Tightening screw post	Fijar	*Tightening_screw_assembly_2*	X10Cr13__mart_410	20,671 gr.
Tube_joining_clip_1	Soltar, Sujetar	*Handle_tube_assembly_1*	ABS_I	47,567 gr.

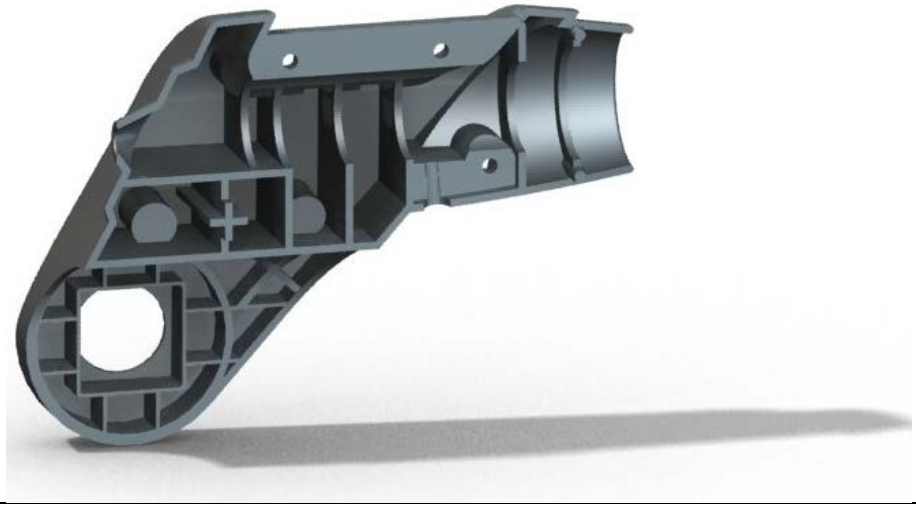
A3-2. PROPIEDADES INFERIDAS

En relación a las propiedades sobre las conexiones que se establecen entre las piezas y los ensamblajes, sólo algunas se identifican con determinadas instancias. La información de las propiedades *hasAction*, *hasFunction* y *has_Environment* y aquellas referidas al tipo de conexión se pueden encontrar en el Aptdo. 5.3.9 y siguientes.

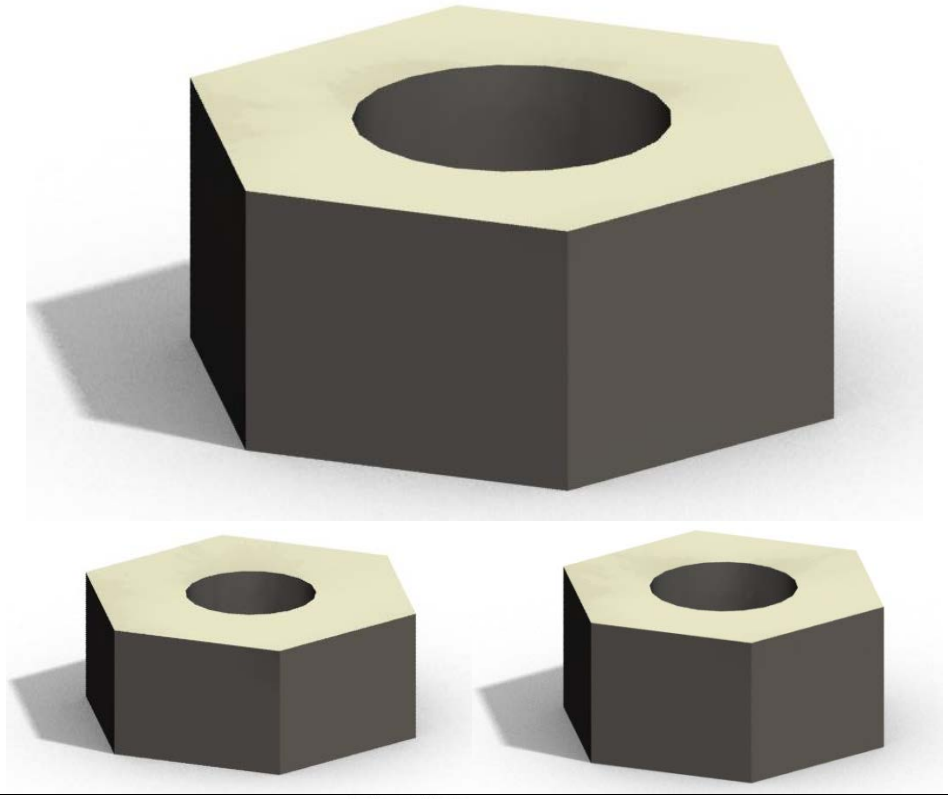
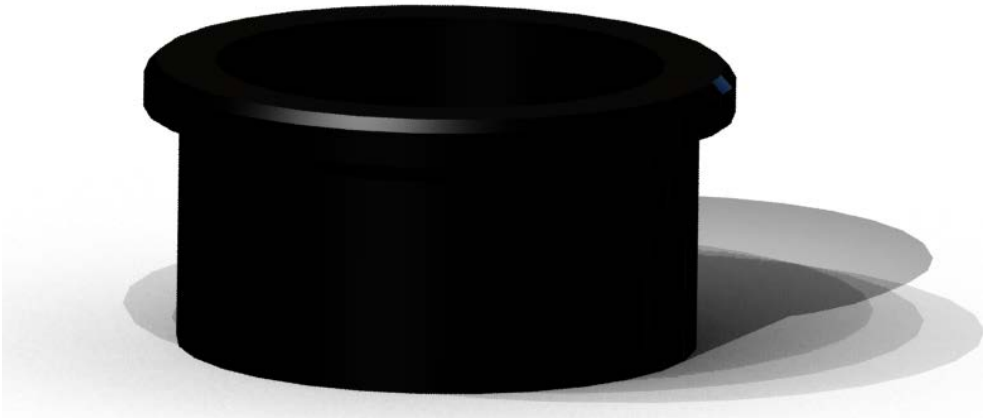
Anexo 4: Imágenes de la estructura del TULUM®[zz].

Estructura	Imagen
Accessory holder	
Aluminum Tube (21x15 A2, B1 & 25x22)	
Cogged Brake (1&2)	
Handle	

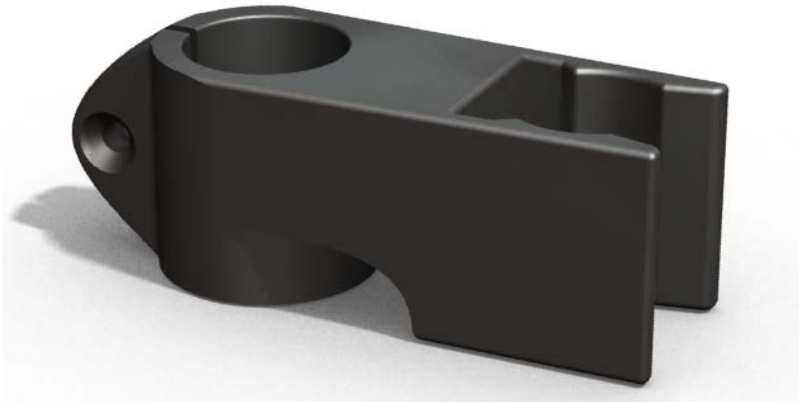
[zz] Las imágenes de este anexo son parte de un trabajo realizado en el Grupo de Ingeniería de Diseño (GID) por Jessica Abad-Kelly.

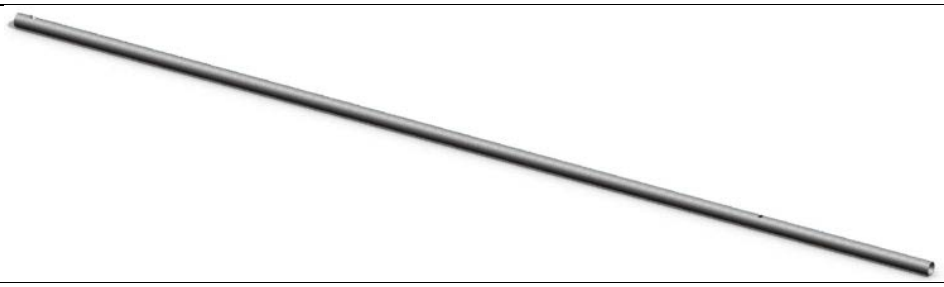
Estructura	Imagen
Handle cover	
Housing cap	
Housing cover	
Housing (left & right)	

Estructura	Imagen
	
M3x20 flat head (1 & 2), M4x16 (1 & 2), M4x20, M4x54 (1 & 2), M5x16 (1, 2, 3), M5x30 flat head	

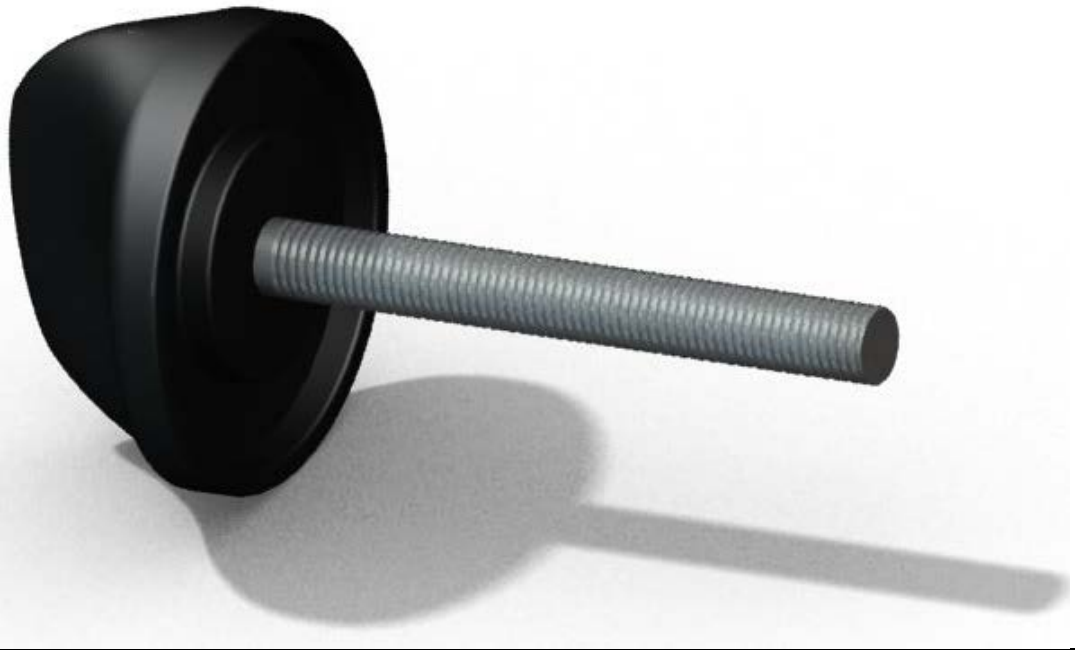
Estructura	Imagen
	
Nut 13 mm, 7 mm (1 - 9), 8 mm (1- 3)	
Plug D 25 mm	

Estructura	Imagen
Rubber grommet (2 & 3)	
Rustproof regulator	
Spring 5 mm (2 & 3)	
Static Cogged Brake	
TV Camera Housing	

Estructura	Imagen
Tightening screw cap	
Tightening screw post	
Tube joining clip	
Camera Assembly	

Estructura	Imagen
Camera Housing Assembly	
Camera Tube Assembly	
Extendable Tube Assembly	
Handle Assembly (1 & 2)	

Estructura	Imagen
Handle Tube Assembly	
Hinge Assembly	
Main Housing Assembly	

Estructura	Imagen
Static Brake Assembly (1 & 2)	
Telescopic tubes Assembly	
Tightening Screw Assembly ®	

Estructura	Imagen
TULUM	

Anexo 5: Resultados de los casos de inferencia de conocimiento del TULUM®.

A5-1.1. Caso 1: Inferencia de estructuras a partir de funciones del TULUM®

Tabla 98: Inferencia de una serie de estructuras a partir de las funciones del TULUM®

Función	Estructura
Alargar	Extendable_tube_assembly_1
	Static_cogged_brake_4
	M4x16_1
	M3x20_flat_head_1
	M4x54_2
	M4x54_1
	Aluminium_tube_25x22_1
	Housing_left_1
	Tube_joining_clip_1
	Aluminium_tube_21x15_B_1
	Accessory_holder_1
	M5x16_3
	Rustproof_regulator_1
	Static_cogged_brake_3
	Camera_tube_assembly_1
	Housing_right_1
	M3x20_flat_head_2
	M4x16_2
	M4x20_1
	M5x16_1
	Telescopic_tubes_assembly
	Camera_assembly_1
	Hinge_assembly_1
	M5x16_2
	M5x30_flat_head_1
	Extendable_tube_assembly_1
	Camera_housing_assembly_1
	Handle_tube_assembly_1
	Aluminium_tube_21x15_A_2
	TULUM

Función	Estructura
Atornillar	M4x20_1
	Nut_7mm_9
	Nut_8mm_2
	Nut_7mm_6
	M4x54_2
	M5x16_1
	M5x30_flat_head_1
	Hinge_assembly_1
	Nut_7mm_2
	Nut_7mm_4
	M5x16_3
	Nut_13mm_1
	Nut_7mm_7
	Nut_7mm_1
	M4x16_2
	Tightening_screw___post_1
	M4x54_1
	Nut_7mm_5
	Tightening_screw_assembly_2
	M3x20_flat_head_1
	Nut_7mm_3
	Nut_7mm_8
	Nut_8mm_3
	M5x16_2
	M4x16_1
	M3x20_flat_head_2
	Tightening_screw___cap_1
	Nut_8mm_1
Desplegar	Main_housing_assembly_1
	Tube_joining_clip_1
	Accessory_holder_1
	Camera_assembly_1
	Tulum___english
	Hinge_assembly_1
	Camera_housing_assembly_1

Función	Estructura
Enganchar	Static_cogged_brake_4
	Accessory_holder_1
	Main_housing_assembly_1
	M5x16_1
	Camera_assembly_1
	M4x54_1
	Camera_housing_assembly_1
	M4x54_2
	Hinge_assembly_1
	M3x20_flat_head_1
	M5x16_2
	M4x16_2
	M5x30_flat_head_1
	M4x16_1
	Tulum___english
	Static_cogged_brake_3
	M4x20_1
	Rustproof_regulator_1
	M5x16_3
	Housing_right_1
	Housing_left_1
	Tube_joining_clip_1
	M3x20_flat_head_2
Girar	M3x20_flat_head_2
	Nut_7mm_1
	Spring_5mm_2
	Nut_13mm_1
	M4x54_2
	Nut_7mm_4
	Nut_7mm_5
	Spring_5mm_3
	M3x20_flat_head_1
	M4x16_1
	Nut_7mm_2
	Hinge_assembly_1

Función	Estructura
	M5x16_2
	Nut_8mm_3
	Nut_8mm_1
	Nut_7mm_8
	Static_brake_assembly_2
	Nut_7mm_9
	M4x54_1
	Tightening_screw___cap_1
	M5x16_3
	M5x16_1
	Tightening_screw_assembly_2
	Nut_7mm_3
	Static_brake_assembly_1
	M4x16_2
	M4x20_1
	Nut_7mm_6
	Nut_7mm_7
	Nut_8mm_2
	M5x30_flat_head_1
Inspeccionar_casco	M5x30_flat_head_1
	M3x20_flat_head_1
	Camera_housing_assembly_1
	M4x54_2
	M4x16_2
	M5x16_2
	Housing_right_1
	M4x16_1
	Main_housing_assembly_1
	M4x54_1
	Tube_joining_clip_1
	Static_cogged_brake_3
	Accessory_holder_1
	Hinge_assembly_1
	Rustproof_regulator_1
	Housing_left_1

Función	Estructura
	Camera_assembly_1
	M3x20_flat_head_2
	TV_camera_housing_1
	M5x16_3
	M5x16_1
	Static_cogged_brake_4
	Tulum___english
	M4x20_1
Manipular	Handle_tube_assembly_1
	Handle_assembly_2
	Tulum___english
	Tightening_screw___cap_1
	Telescopic_tubes_assembly_2
	Handle_1
	Handle_assembly_1
	Tightening_screw_assembly_2
Plegar	M4x54_2
	Static_cogged_brake_3
	Tulum___english
	Housing_left_1
	M5x16_2
	Main_housing_assembly_1
	M5x16_3
	Housing_right_1
	M5x16_1
	M4x16_2
	Rustproof_regulator_1
	M4x16_1
	Camera_assembly_1
	M5x30_flat_head_1
	M4x20_1
	M3x20_flat_head_2
	M3x20_flat_head_1
	Accessory_holder_1
	M4x54_1

Función	Estructura
	Hinge_assembly_1
	Camera_housing_assembly_1
	Tube_joining_clip_1
	Static_cogged_brake_4
Proteger	M3x20_flat_head_2
	M5x30_flat_head_1
	M4x54_1
	Hinge_assembly_1
	M5x16_2
	Nut_8mm_3
	Nut_7mm_9
	Nut_7mm_7
	Nut_7mm_1
	Tightening_screw_assembly_2
	M4x20_1
	Tightening_screw___post_1
	Nut_7mm_8
	Nut_7mm_4
	M4x16_1
	M4x54_2
	Tightening_screw___cap_1
	Nut_13mm_1
	Nut_8mm_1
	Nut_7mm_6
	Nut_7mm_5
	M3x20_flat_head_1
	M5x16_1
	M4x16_2
	M5x16_3
	Nut_8mm_2
	Nut_7mm_2
	Nut_7mm_3
Rotar	Spring_5mm_2
	Static_brake_assembly_1
	Hinge_assembly_1

Función	Estructura
	Static_cogged_brake_4
	Cogged_brake_2
	Static_cogged_brake_3
	Static_brake_assembly_2
	Spring_5mm_3
	Tightening_screw_assembly_2
	Cogged_brake_1
Utilizar	Accessory_holder_1
	Aluminium_tube_25x22_1
	Nut_8mm_1
	Main_housing_assembly_1
	Nut_7mm_7
	Rustproof_regulator_1
	Static_cogged_brake_4
	M3x20_flat_head_2
	Aluminium_tube_21x15_A_2
	Housing_left_1
	Aluminium_tube_21x15_B_1
	Static_cogged_brake_3
	Static_brake_assembly_2
	Handle_assembly_1
	Nut_8mm_3
	Handle_tube_assembly_1
	Nut_13mm_1
	Nut_7mm_2
	Static_brake_assembly_1
	M5x16_1
	Housing_right_1
	Tightening_screw_assembly_2
	Tube_joining_clip_1
	Cogged_brake_2
	Tulum___english
	Nut_7mm_3
	Nut_7mm_9
	Camera_tube_assembly_1

Función	Estructura
	Extendable_tube_assembly_1
	Spring_5mm_2
	Spring_5mm_3
	Nut_8mm_2
	M4x16_2
	Nut_7mm_8
	M4x54_2
	Nut_7mm_1
	Handle_1
	M4x16_1
	Telescopic_tubes_assembly_2
	Tightening_screw___cap_1
	Hinge_assembly_1
	M4x20_1
	M5x30_flat_head_1
	M5x16_2
	Camera_housing_assembly_1
	Camera_assembly_1
	M5x16_3
	Handle_assembly_2
	Nut_7mm_6
	M4x54_1
	Nut_7mm_4
	Nut_7mm_5
	Cogged_brake_1
	M3x20_flat_head_1
Ver	Camera_assembly_1
	TV_camera_housing_1
	Tulum___english

A5-1.2. Caso 2: Inferencia de funciones a partir de estructuras del TULUM®

Tabla 99: Inferencia de una serie de funciones a partir de las estructuras del TULUM®

Función	Estructura
Accessory_holder_1	Alargar
	Desplegar
	Enganchar
	Plegar
Aluminium_tube_21x15_A_2	Alargar
Aluminium_tube_21x15_B_1	Alargar
Aluminium_tube_25x22_1	Alargar
Camera_assembly_1	Alargar
	Desplegar
	Enganchar
	Plegar
	Ver
Camera_housing_assembly_1	Alargar
	Desplegar
	Enganchar
	Plegar
Camera_tube_assembly_1	Alargar
Cogged_brake_1	Rotar
Cogged_brake_2	Rotar
Extendable_tube_assembly_1	Alargar
Handle_1	Manipular
Handle_assembly_1	Manipular
Handle_assembly_2	Manipular
Handle_tube_assembly_1	Alargar
	Manipular
Hinge_assembly_1	Alargar
	Atornillar
	Desplegar
	Enganchar
	Girar
	Plegar
	Rotar

Función	Estructura
Housing_left_1	Alargar
	Enganchar
	Plegar
Housing_right_1	Alargar
	Enganchar
	Plegar
M3x20_flat_head_1	Alargar
	Atornillar
	Enganchar
	Girar
M3x20_flat_head_2	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M4x16_1	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M4x16_2	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M4x20_1	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M4x54_1	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar

Función	Estructura
M4x54_2	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M5x16_1	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M5x16_2	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M5x16_3	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
M5x30_flat_head_1	Alargar
	Atornillar
	Enganchar
	Girar
	Plegar
Main_housing_assembly_1	Desplegar
	Enganchar
	Plegar
Nut_13mm_1	Girar
Nut_7mm_1	Atornillar
	Girar
Nut_7mm_2	Atornillar
	Girar
Nut_7mm_3	Atornillar
	Girar

Función	Estructura
Nut_7mm_4	Atornillar
	Girar
Nut_7mm_5	Atornillar
	Girar
Nut_7mm_6	Atornillar
	Girar
Nut_7mm_7	Atornillar
	Girar
Nut_7mm_8	Atornillar
	Girar
Nut_7mm_9	Atornillar
	Girar
Nut_8mm_1	Atornillar
	Girar
Nut_8mm_2	Atornillar
	Girar
Nut_8mm_3	Atornillar
	Girar
Rustproof_regulator_1	Alargar
	Enganchar
	Plegar
Spring_5mm_2	Girar
	Rotar
Spring_5mm_3	Girar
	Rotar
Static_brake_assembly_1	Girar
	Rotar
Static_brake_assembly_2	Girar
	Rotar
Static_cogged_brake_3	Alargar
	Enganchar
	Plegar
	Rotar
Static_cogged_brake_4	Alargar
	Enganchar

Función	Estructura
	Plegar
	Rotar
Telescopic_tubes_assembly_2	Alargar
	Manipular
Tightening_screw___cap_1	Atornillar
	Girar
	Manipular
Tightening_screw___post_1	Atornillar
Tightening_screw_assembly_2	Atornillar
	Girar
	Manipular
Tube_joining_clip_1	Alargar
	Enganchar
	Plegar
TULUM	Alargar
	Desplegar
	Enganchar
	Manipular
	Plegar
	Ver
TV_camera_housing_1	Ver

A5-1.3. Caso 4: Inferencia de los tipos de conexiones del TULUM®

A5-1.3.1. TIPO DE RELACIÓN COINCIDENTE

Tabla 100: Inferencia de datos de la Regla 4 para el caso de la relación coincidente con la propiedad de relación *isCoincidentTo*

Estructura conectada1	Estructura conectada2
Handle_1	Handle_assembly_2
Aluminium_tube_21x15_A_2	Rustproof_regulator_1
M4x16_1	Housing_right_1
Housing_right_1	M4x16_1
Aluminium_tube_25x22_1	Rustproof_regulator_1
Handle_1	M5x16_1
Housing_right_1	Tightening_screw___post_1
Main_housing_assembly_1	Static_cogged_brake_4
Tube_joining_clip_1	M5x16_3

Estructura conectada1	Estructura conectada2
M5x16_3	Tube_joining_clip_1
Housing_right_1	Hinge_assembly_1
Handle_1	Handle_assembly_1
Handle_cover_1	Handle_assembly_1
Hinge_assembly_1	Telescopic_tubes_assembly_2
Telescopic_tubes_assembly_2	Hinge_assembly_1
Aluminium_tube_21x15_B_1	Rubber_grommet_3
Tightening_screw___cap_1	Tightening_screw___post_1
Housing_right_1	M4x54_2
M4x54_2	Housing_right_1
Housing_right_1	Main_housing_assembly_1
M4x54_1	Nut_7mm_4
Housing_left_1	Nut_7mm_8
Tightening_screw___post_1	Spring_5mm_3
Static_cogged_brake_4	Housing_right_1
Rustproof_regulator_1	Extendable_tube_assembly_1
Housing_right_1	Camera_housing_assembly_1
M4x54_1	Main_housing_assembly_1
M5x16_2	Nut_8mm_3
M5x16_3	Nut_8mm_2
Nut_7mm_5	Camera_housing_assembly_1
Telescopic_tubes_assembly_2	Camera_assembly_1
Nut_7mm_5	Main_housing_assembly_1
Housing_cap_1	Main_housing_assembly_1
Plug_D25mm_2	Handle_tube_assembly_1
Handle_1	Nut_8mm_3
Static_cogged_brake_4	Static_brake_assembly_2
Static_brake_assembly_2	Static_cogged_brake_4
Static_brake_assembly_2	Static_cogged_brake_3
Static_cogged_brake_3	Static_brake_assembly_2
Housing_right_1	M4x54_1
Aluminium_tube_25x22_1	Handle_tube_assembly_1
Aluminium_tube_25x22_1	Aluminium_tube_21x15_A_2
Main_housing_assembly_1	Hinge_assembly_1
Housing_right_1	Cogged_brake_1

Estructura conectada1	Estructura conectada2
Aluminium_tube_25x22_1	Handle_assembly_2
Handle_1	Nut_8mm_1
Aluminium_tube_21x15_B_1	Camera_tube_assembly_1
M5x16_1	Nut_8mm_1
M4x54_1	Nut_7mm_9
Plug_D25mm_2	Aluminium_tube_25x22_1
Aluminium_tube_25x22_1	Plug_D25mm_2
Nut_7mm_6	Housing_left_1
Housing_left_1	Cogged_brake_2
Housing_left_1	Nut_7mm_9
Tightening_screw___post_1	Tightening_screw_assembly_2
Static_cogged_brake_4	Spring_5mm_3
Nut_7mm_1	Housing_left_1
Handle_tube_assembly_1	Telescopic_tubes_assembly_2
Accessory_holder_1	Camera_housing_assembly_1
Tube_joining_clip_1	Nut_8mm_2
Nut_7mm_4	Housing_left_1
Housing_left_1	Nut_7mm_7
Aluminium_tube_21x15_A_2	Extendable_tube_assembly_1
Handle_assembly_1	Handle_tube_assembly_1
M4x16_1	Nut_7mm_7
M4x16_2	Nut_7mm_6
TV_camera_housing_1	Camera_housing_assembly_1
Spring_5mm_3	Static_brake_assembly_2
M4x20_1	Nut_7mm_3
Housing_left_1	Camera_housing_assembly_1
M4x54_2	Camera_housing_assembly_1
Housing_right_1	M4x20_1
M4x20_1	Housing_right_1
Rubber_grommet_3	Camera_tube_assembly_1
M4x16_1	Nut_7mm_2
M4x16_2	Nut_7mm_1
Tightening_screw___post_1	Nut_13mm_1
Cogged_brake_2	Tightening_screw___cap_1
M5x16_2	Handle_1

Estructura conectada1	Estructura conectada2
Handle_1	M5x16_2
Tube_joining_clip_1	Handle_tube_assembly_1
Housing_cap_1	M5x30_flat_head_1
Aluminium_tube_25x22_1	Tube_joining_clip_1
Housing_cover_1	Main_housing_assembly_1
Housing_left_1	Nut_7mm_3
Housing_left_1	Nut_7mm_2
Telescopic_tubes_assembly_2	Tulum___english
Housing_left_1	Nut_7mm_5
Housing_right_1	M4x16_2
M4x16_2	Housing_right_1
Tightening_screw___post_1	Spring_5mm_2
M4x54_2	Main_housing_assembly_1
Tightening_screw___cap_1	Tightening_screw_assembly_2
M3x20_flat_head_1	Static_brake_assembly_2
Static_cogged_brake_3	Housing_right_1
Cogged_brake_1	Nut_13mm_1
Static_cogged_brake_3	Spring_5mm_2
M4x20_1	Nut_7mm_8
Housing_left_1	Main_housing_assembly_1
Rubber_grommet_2	Camera_tube_assembly_1
Aluminium_tube_25x22_1	Handle_assembly_1
Handle_1	Handle_cover_1
Handle_cover_1	Handle_assembly_2
Rubber_grommet_2	Aluminium_tube_21x15_B_1
M3x20_flat_head_2	Static_brake_assembly_2
Hinge_assembly_1	Camera_tube_assembly_1
M4x54_1	Camera_housing_assembly_1
Main_housing_assembly_1	Static_cogged_brake_3
Handle_assembly_2	Handle_tube_assembly_1
Spring_5mm_2	Static_brake_assembly_2

A5-1.3.2. TIPO DE RELACIÓN DISTANTE

Tabla 101: Inferencia de datos de la Regla 4 para el caso de la relación *distante*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Distante	*hasDistanceTo*	M5x16_1	Handle_assembly_2
		M5x16_2	Handle_assembly_1
		M3x20_flat_head_1	Static_cogged_brake_3
		M5x16_3	Tube_joining_clip_1
		Aluminium_tube_21x15_B_1	Rubber_grommet_2
		Rubber_grommet_3	Aluminium_tube_21x15_B_1
		Spring_5mm_3	Static_cogged_brake_4
		Spring_5mm_2	Static_cogged_brake_3
		Housing_right_1	TV_camera_housing_1
		Tightening_screw___cap_1	Camera_housing_assembly_1
		Handle_1	Tube_joining_clip_1
		Handle_1	Plug_D25mm_2
		M5x16_2	Nut_8mm_3
		M5x16_3	Nut_8mm_2
		Handle_1	Handle_1

A5-1.3.3. TIPO DE RELACIÓN CONCÉNTRICA

Tabla 102: Inferencia de datos de la Regla 4 para el caso de la relación *concéntrica*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Concéntrica	*isConcentricTo*	Static_cogged_brake_4	M3x20_flat_head_2
		M5x30_flat_head_1	Housing_cap_1
		Static_cogged_brake_3	M3x20_flat_head_1
		Aluminium_tube_21x15_A_2	Rustproof_regulator_1

A5-1.3.4. TIPO DE RELACIÓN BLOQUEADA

Tabla 103: Inferencia de datos de la Regla 4 para el caso de la relación *bloqueada*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Bloqueada (*Lock*)	*isGearTo*	Nut_7mm_7	Nut_7mm_6
		Rustproof_regulator_1	Aluminium_tube_21x15_A_2
		Housing_cap_1	M5x30_flat_head_1
		Nut_7mm_2	Nut_7mm_1

A5-1.3.5. TIPO DE RELACIÓN PARALELA

Tabla 104: Inferencia de datos de la Regla 4 para el caso de la relación *paralela*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Paralela	*isParallelTo*	M3x20_flat_head_2	Static_brake_assembly_2
		TV_camera_housing_1	Static_brake_assembly_2
		Main_housing_assembly_1	Static_cogged_brake_3
		Main_housing_assembly_1	Static_cogged_brake_4
		Nut_7mm_4	Camera_housing_assembly_1
		Housing_left_1	Cogged_brake_2
		Nut_7mm_9	Main_housing_assembly_1
		Housing_left_1	Nut_7mm_5
		Housing_left_1	M4x16_1
		Housing_right_1	M4x54_2
		M3x20_flat_head_1	Static_brake_assembly_2
		M4x20_1	Camera_housing_assembly_1
		Housing_right_1	M4x16_1
		Housing_left_1	Nut_7mm_6
		Housing_right_1	M4x16_2
		Housing_left_1	Nut_7mm_1
		Housing_right_1	Cogged_brake_1
		Housing_right_1	M4x54_1
		M4x20_1	Main_housing_assembly_1

A5-1.3.6. TIPO DE RELACIÓN PERPENDICULAR

Tabla 105: Inferencia de datos de la Regla 4 para el caso de la relación *perpendicular*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Perpendicular	*isPerpendicularTo*	M5x30_flat_head_1	Main_housing_assembly_1

A5-1.3.7. TIPO DE RELACIÓN TANGENTE

Tabla 106: Inferencia de datos de la Regla 4 para el caso de la relación *tangente*

Tipo de relación	Propiedad de la relación	Estructura conectada1	Estructura conectada2
Tangente	*isTangentTo*	Telescopic_tubes_assembly_2	Hinge_assembly_1
		Camera_tube_assembly_1	Camera_assembly_1

A5-1.4. Caso 8: Inferencia de los porcentajes de los materiales en los ensamblajes del TULUM®

Tabla 107: Inferencia de los porcentajes de los materiales en los ensamblajes del TULUM®

Ensamblaje	Partes	Material	Porcentaje (%)
Camera_housing_assembly_1	TV_camera_housing_1	_6060_Alloy	33,03
	Housing_left_1	ABS_I	16,99
	Housing_right_1		16,80
	Accessory_holder_1		10,55
	Cogged_brake_1		2,71
	Cogged_brake_2		2,71
	M4x54_1	X10Cr13__mart_410_	1,57
	M4x54_2		1,57
	Nut_13mm_1		1,34
	M4x20_1		0,68
	M4x16_1		0,58
	M4x16_2		0,58
	Nut_7mm_1		0,22
	Nut_7mm_2		0,22
	Nut_7mm_3		0,22
	Nut_7mm_4		0,22
	Nut_7mm_5		0,22
Camera_tube_assembly_1	Aluminium_tube_21x15_B_1	_6060_Alloy	94,56
	Rubber_grommet_2	SBR	2,72
	Rubber_grommet_3		2,72
Extendable_tube_assembly_1	Aluminium_tube_21x15_A_2	_6060_Alloy	99,26
	Rustproof_regulator_1		0,74
Handle_assembly_1	Handle_1	ABS_I	89,81
	Handle_cover_1		10,19
Handle_assembly_2	Handle_1	ABS_I	89,81
	Handle_cover_1		10,19
Handle_tube_assembly_1	Aluminium_tube_25x22_1	_6060_Alloy	64,46
	Tube_joining_clip_1	ABS_I	7,03
	Plug_D25mm_2	SBR	1,90
	M5x16_1	X10Cr13__mart_410_	0,43
	M5x16_2		0,43
	M5x16_3		0,43
	Nut_8mm_1		0,20

Ensamblaje	Partes	Material	Porcentaje (%)
	Nut_8mm_2		0,20
	Nut_8mm_3		0,20
Main_housing_assembly_1	Housing_left_1	ABS_I	25,38
	Housing_right_1		25,10
	Housing_cap_1		7,27
	Housing_cover_1		6,43
	Cogged_brake_1		4,05
	Cogged_brake_2		4,05
	M4x54_1	X10Cr13__mart_410_	2,34
	M4x54_2		2,34
	M5x30_flat_head_1		2,01
	Nut_13mm_1		2,00
	M4x20_1		1,02
	M4x16_1		0,87
	M4x16_2		0,87
	Nut_7mm_5		0,33
	Nut_7mm_6		0,33
	Nut_7mm_7		0,33
	Nut_7mm_8		0,33
	Nut_7mm_9		0,33
Static_brake_assembly_1	Spring_5mm_2	_50CrV4	71,96
	Static_cogged_brake_3	ABS_I	27,53
	M3x20_flat_head_1	X10Cr13__mart_410_	0,52
Static_brake_assembly_2	Spring_5mm_3	_50CrV4	71,96
	Static_cogged_brake_4	ABS_I	27,53
	M3x20_flat_head_2	X10Cr13__mart_410_	0,52
Tightening_screw_assembly_2	Tightening_screw___post_1	X10Cr13__mart_410_	53,96
	Tightening_screw___cap_1	ABS_I	46,04

Anexo 6: Métrica de OntoFaBES aplicada al TULUM®

Tabla 108: Recuento de los axiomas de clase

Métrica	Recuento de axiomas
Subclases	28
Clases equivalentes	7
Clases disjuntas	14
GCI	0
GCI oculto	5

Tabla 109: Recuento de los axiomas de propiedad de objeto

Métrica	Recuento de axiomas
Subpropiedades de objeto	19
Propiedades inversas	25
Propiedades simétricas	11
Dominio de propiedades	41
Rango de propiedades	42

Tabla 110: Recuento de los axiomas de propiedad de datos

Métrica	Recuento de axiomas
Subpropiedades de datos	40
Propiedades funcionales	37
Dominio de propiedades	12
Rango de propiedades	31

Tabla 111: Recuento de los axiomas de individuos

Métrica	Recuento
Aserción de clases	555
Aserción de propiedades de objetos	2692
Aserción de propiedades de datos	2686

Tabla 112: Recuento de los axiomas de anotaciones

Métrica	Recuento
Aserción de anotaciones	103
Dominio de propiedad de anotaciones	7

Tabla 113: Características relativas a la dimensión "Herramienta" para OntoFaBES

FACTORES	VALORES
HABILIDADES	**Alto**
Uso local	soportado
Uso en red	soportado
Uso basado en Internet	soportado
Claridad de la interfaz del usuario	muy alto
Tiempo de respuesta	alto
Fiabilidad	medio
VISUALIZACIÓN	**Muy alto**
El navegador muestra la completa información de los términos	soportado
El navegador permite la selección a nivel de detalle	soportado
El navegador muestra la taxonomía	soportado
El navegador muestra relaciones *ad hoc*	soportado
EDICIÓN	**Muy alto**
La herramienta construye en el mismo lenguaje	soportado
La herramienta permite la edición en cualquier momento	soportado
La herramienta muestra la taxonomía gráficamente	soportado
La herramienta permite la definición de nuevas relaciones	soportado
INTERACCIÓN	**Alto**
La herramienta permite un uso independiente	soportado
La herramienta suministra interfaces de acceso	soportado
Existe documentación sobre las interfaces de acceso	alto
Las interfaces de acceso son *Open Source*	soportado
Documentación sobre interfaces de programación de acceso	alto
ASPECTOS METODOLÓGICOS	**Alto**
La herramienta soporta el ciclo de vida completo	no soportado
La herramienta apoya actividades de desarrollo importantes	soportado
La herramienta suministra documentación sobre productos creados	soportado
La herramienta chequea la consistencia	soportado
ASPECTOS COOPERATIVOS	**Muy Alto**
La herramienta crea grupos de trabajo	soportado
La herramienta permite trabajo simultáneo	soportado
La herramienta mira ontologías editadas	soportado
La herramienta mira términos editados	soportado
La herramienta notifica los cambios al grupo	soportado
La herramienta identifica los cambios del usuario	soportado
TRADUCCIÓN	**Medio**
La herramienta importa datos de otras lenguas	soportado
La herramienta importa datos de otras lenguas de marcado	soportado
La herramienta exporta a otras lenguas	soportado
La herramienta exporta a otras lenguas de marcado	soportado
Las traducciones pierden un mínimo de semántica	no soportado
La traducción es supervisada	no soportado
INTEGRACIÓN	**Medio**
Facilidad de integración	alto
Dificultad de referir a nuevos términos	bajo
La herramienta permite la selección de términos para su integración	soportado
La herramienta chequea la consistencia en su integración o fusión	no soportado
Asistencia para fusión manual	soportado
Fusión semi-automática	no soportado

Tabla 114: Características relativas a la dimensión "Lenguaje" para OntoFaBES

FACTORES	VALORES
CONCEPTOS/INSTANCIAS/HECHOS/PETICIONES	**muy alto**
Permite las instancias de clases	soportado
Tiene metaclases	soportado
Puede definir clases sin metaclases	soportado
Claridad de la interfaz del usuario	soportado
Permite hechos	soportado
Permite peticiones	soportado
ATRIBUTOS	**muy alto**
Puede definir atributos de clases	soportado
Puede definir atributos de instancias	soportado
Puede definir atributos locales	soportado
Puede definir atributos globales	soportado
Puede definir atributos polimórficos	soportado
Puede definir atributos sobre excepciones	soportado
FACETAS	**muy alto**
Tiene valores de atributos por defecto	soportado
Tiene tipos de atributos	soportado
Puede definir la cardinalidad de los atributos	soportado
Permite la definición de conocimiento sobre el procedimiento	soportado
Permite nuevas facetas	soportado
RELACIONES	**muy alto**
Permite la definición de funciones	soportado
Hay relaciones arbitrarias enésimas	soportado
Se permite la definición de relaciones *ad hoc*	soportado
Se puede restringir el tipo en las relaciones	soportado
Se puede restringir el valor en las relaciones	soportado
Se definen las operaciones a realizar	soportado
Se pueden declarar las propiedades en las relaciones	soportado
TAXONOMÍAS	**muy alto**
Contienen la relación: *Subclase de*.	soportado
Contienen la relación: *No es subclase de*.	soportado
Múltiples subclases de clases	soportado
Múltiples instancias derivadas de otras instancias	soportado
AXIOMAS	**muy alto**
Permite los axiomas embebidos en términos	soportado
Permite axiomas independientes	soportado
Permite axiomas en lógica de primer orden	soportado
Permite axiomas en lógica de segundo orden	soportado
REGLAS DE PRODUCCIÓN	**muy alto**
Permite reglas disyuntivas en las reglas de producción	soportado
Permite reglas conjuntivas en las reglas de producción	soportado
Cada regla tiene definido un mecanismo de encadenamiento	soportado
Cada regla tiene definida una prioridad	soportado
Procedimientos en los resultados en la reglas de producción	soportado

Tabla 115: Características relativas a la dimensión "Contenido" para OntoFaBES

FACTORES	VALORES
CONCEPTOS	**muy alto**
Conceptos esenciales	muy alto
Conceptos esenciales en los niveles superiores	muy alto
Los conceptos se describen apropiadamente en lenguaje natural	muy alto
La especificación formal de los conceptos coincide con el lenguaje natural	muy alto
Los atributos definen los conceptos	muy alto
Número de conceptos	muy alto
RELACIONES	**alto**
Relaciones esenciales	muy alto
Las relaciones corresponden conceptos apropiados	muy alto
La especificación formal de las relaciones coincide con el lenguaje natural	muy alto
Aridad especificada	alto
Propiedades formales de las relaciones	muy alto
Número de relaciones	muy alto
TAXONOMÍA	**alto**
Perspectivas diversas	alto
Apropiadas *no-subclases-de*	muy alto
Particiones exhaustivas apropiadas	muy alto
Particiones disjuntas apropiadas	muy alto
Máxima profundidad	alto
Media de subclases	muy alto
AXIOMAS	**alto**
Los axiomas resuelven cuestiones	muy alto
Los axiomas infieren conocimiento	muy alto
Los axiomas verifican la consistencia	muy alto
Los axiomas no se enlazan a conceptos	muy alto
Número de axiomas	alto

Tabla 116: Características relativas a la dimensión "Metodología" para OntoFaBES

FACTORES	VALORES
PRECISIÓN	**alto**
Delimitación de fases	muy alto
Especificación de actividades por fases	muy alto
Especificación de personal por fases	muy alto
Especificación de técnicas por fases	alto
Especificación de productos finales por fases	Medio
USABILIDAD	**alto**
Claridad de actividades y descripción de técnicas	alto
Calidad de manuales	alto
Manuales con ejemplos completos	alto
MADUREZ	**medio**
Número de ontologías desarrolladas	bajo
Número de diferentes dominios	medio
Importancia de las ontologías desarrolladas	alto

Tabla 117: Características relativas a la dimensión "Costes" para OntoFaBES

FACTORES	VALORES
Uso de las licencias de la ontología	bajo
Costes estimados de hardware y software	bajo
Costes de las interfaces de acceso	bajo
Utiliza	muy bajo

Anexo 7: Código XML de la ontología OntoFaBES aplicada al TULUM®

Se indica a continuación una muestra parcial del código XML de la ontología OntoFaBES aplicada al caso del TULUM® referida a su parte inicial. El código completo de OntoFaBES se puede encontrar en http://www.derpilgrim.es/OntoFaBES.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
<!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
<!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
<!ENTITY bp "http://bioportal.bioontology.org#" >
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
<!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
<!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
<!ENTITY swrla "http://swrl.stanford.edu/ontologies/3.3/swrla.owl#" >
<!ENTITY sqwrl "http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#" >
]>

<rdf:RDF xmlns="http://www.FaBES.com/OntoFaBES.owl#"
        xml:base="http://www.FaBES.com/OntoFaBES.owl"
        xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl#"
        xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
        xmlns:swrl="http://www.w3.org/2003/11/swrl#"
        xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
        xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
        xmlns:bp="http://bioportal.bioontology.org#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:owl="http://www.w3.org/2002/07/owl#">
        <owl:Ontology rdf:about="">
            <owl:versionInfo rdf:datatype="&xsd;string"
        >TODO: hasAQB_Mass tiene una propiedad a a&#241;adir luego para explicar el dise&#241;o
axiom&#225;tico</owl:versionInfo>
        <rdfs:comment rdf:datatype="&xsd;string"
>Avanzamos en la ontolog&#237;a para definir las reglas en SWRL y a&#241;adir aquellas clases que sean necesarias para su
correcta definici&#243;n. Iremos haciendo las anotaciones oportunas.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string"
            >La conductividad t&#233;rmica est&#225; ligada a la conductividad el&#233;ctrica. Los metales provocan la
corrosi&#243;n por interecci&#243;n el&#233;ctrica.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string"
            >environTo y la de hasEnvironment ya no son funcionales...(Hay que modificarlo en la tesis). Por otro lado se modifican
los entornos de aplicaci&#243;n a las diferentes partes del TULUM,y luego se debe generar que la regla cuando encuentre el
ensamblaje q corresponde</rdfs:comment>
            <owl:imports rdf:resource="http://sqwrl.stanford.edu/ontologies/built-ins/3.4/sqwrl.owl"/>
            <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrla.owl"/>
</owl:Ontology>
        <owl:AllDifferent>
            <owl:distinctMembers rdf:parseType="Collection"/>
        </owl:AllDifferent>
        <swrl:Variable rdf:ID="A"/>
        <swrl:Variable rdf:ID="a"/>
        <swrl:Variable rdf:ID="A2"/>
        <swrl:Variable rdf:ID="A3"/>
        <swrl:Variable rdf:ID="A4"/>
        <swrl:Variable rdf:ID="A5"/>
        <swrl:Variable rdf:ID="Ag"/>
        <swrl:Variable rdf:ID="Am"/>
        <swrl:Variable rdf:ID="Amub"/>
        <swrl:Variable rdf:ID="Amub2"/>
        <swrl:Variable rdf:ID="AP"/>
```

```
        <swrl:Variable rdf:ID="APQ"/>
        <swrl:Variable rdf:ID="As"/>
        <swrl:Variable rdf:ID="As1"/>
        <swrl:Variable rdf:ID="As2"/>
        <swrl:Variable rdf:ID="Ass"/>
        <swrl:Variable rdf:ID="Ass2"/>
        <swrl:Variable rdf:ID="ATQ"/>
        <swrl:Variable rdf:ID="B"/>
        <swrl:Variable rdf:ID="b"/>
        <swrl:Variable rdf:ID="Ba"/>
        <swrl:Variable rdf:ID="Baf"/>
        <swrl:Variable rdf:ID="Bap"/>
        <swrl:Variable rdf:ID="Bpr"/>
        <swrl:Variable rdf:ID="CT"/>
        <swrl:Variable rdf:ID="CTS1"/>
        <swrl:Variable rdf:ID="CTS2"/>
        <swrl:Variable rdf:ID="E"/>
        <swrl:Variable rdf:ID="el"/>
        <swrl:Variable rdf:ID="F"/>
        <swrl:Variable rdf:ID="Fach"/>
        <swrl:Variable rdf:ID="Faf"/>
        <swrl:Variable rdf:ID="Fap"/>
        <swrl:Variable rdf:ID="G"/>
        <swrl:Variable rdf:ID="gb"/>
        <swrl:Variable rdf:ID="greatMaterial"/>
        <swrl:Variable rdf:ID="hH"/>
        <swrl:Variable rdf:ID="hP"/>
        <swrl:Variable rdf:ID="hT"/>
        <swrl:Variable rdf:ID="hTC"/>
        <swrl:Variable rdf:ID="M"/>
        <swrl:Variable rdf:ID="m"/>
        <swrl:Variable rdf:ID="M1"/>
        <swrl:Variable rdf:ID="M2"/>
        <swrl:Variable rdf:ID="mass"/>
        <swrl:Variable rdf:ID="mass1"/>
        <swrl:Variable rdf:ID="massy"/>
        <swrl:Variable rdf:ID="N"/>
        <swrl:Variable rdf:ID="n"/>
        <swrl:Variable rdf:ID="n2"/>
        <swrl:Variable rdf:ID="p"/>
        <swrl:Variable rdf:ID="p1"/>
        <swrl:Variable rdf:ID="p2"/>
        <swrl:Variable rdf:ID="pc"/>
        <swrl:Variable rdf:ID="PESO"/>
        <swrl:Variable rdf:ID="peso"/>
        <swrl:Variable rdf:ID="PESO_Ensamblaje"/>
        <swrl:Variable rdf:ID="porc"/>
        <swrl:Variable rdf:ID="porcentaje"/>
        <swrl:Variable rdf:ID="s"/>
        <swrl:Variable rdf:ID="s1"/>
        <swrl:Variable rdf:ID="s2"/>
        <swrl:Variable rdf:ID="s3"/>
        <swrl:Variable rdf:ID="Si"/>
        <swrl:Variable rdf:ID="St"/>
        <swrl:Variable rdf:ID="Str"/>
        <swrl:Variable rdf:ID="Str2"/>
        <swrl:Variable rdf:ID="t2"/>
        <swrl:Variable rdf:ID="Type"/>
        <swrl:Variable rdf:ID="Y1"/>
        <swrl:Variable rdf:ID="Y2"/>
        <Material rdf:ID="_50CrV4">
            <hasDensity rdf:datatype="&xsd;float">7.85</hasDensity>
            <hasElasticModulus rdf:datatype="&xsd;float">200000.0</hasElasticModulus>
            <hasPoissonRatio rdf:datatype="&xsd;float">260000.0</hasPoissonRatio>
            <hasShearModulus rdf:datatype="&xsd;float">79000.0</hasShearModulus>
[…]
```