
Tesis doctoral

Architecture of Computational Ecosystems

Angad Warang



Aquesta tesi doctoral està subjecta a la licència [Reconeixement-NoComercial-SenseObraDerivada 4.0 Internacional \(CC BY-NC-ND 4.0\)](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Esta tesis doctoral está sujeta a la licencia [Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional \(CC BY-NC-ND 4.0\)](https://creativecommons.org/licenses/by-nc-nd/4.0/)

This doctoral thesis is licensed under the [Attribution-NonCommercial-NoDerivatives 4.0 International \(CC BY-NC-ND 4.0\)](https://creativecommons.org/licenses/by-nc-nd/4.0/)

Architecture of Computational Ecosystems

authored by Angad Warang

supervised by
Ddr. Alberto T. Estévez
and Dr. Pablo Baquero

UIC
barcelona

Thesis title: **Architecture of Computational Ecosystems**

Author's name: **Angad Warang**

DOCTORAL THESIS

Universitat Internacional de Catalunya, 2021

Supervisors:

Ddr. Alberto T. Estévez

Dr. Pablo Baquero

Doctoral programme in Architecture



*In the memory of John Horton Conway (1937 - 2020),
who died from COVID-19 complications.*

*May his 'Game of Life' sustain an infinite runtime
and continue to inspire us...*

The real problem of humanity is the following:

We have

Paleolithic emotions,

Medieval institutions,

and God-like technology

-Edward O. Wilson (1929)

This page intentionally left blank

Abstract

Industry 4.0 has rapidly and significantly transformed the Architecture, Engineering, and Construction (**AEC**) industry globally since the early 2000s. The allographic architectural profession until the late 1990s has seen a paradigm shift towards a more autographic practice after the inclusion of the **algorithm** in the otherwise traditional construction of the **built form** through **design**. The rise of the **algorithm** in the AEC industry comes with robust infrastructural support from the unprecedented emergence of **computational design** and **digital fabrication**. Judging by the rate of technological progress, it is highly likely that computational design would become more autonomous and digital fabrication would become more data-driven. Due to the increased probability in the ubiquity of the **algorithm, design** (as a tool for rationalizing form, space and enclosure and creating construction documents) stands on the threshold of becoming redundant. This research and thesis are based on the assumption that **design** would be replaced by a communicational logic between the **built form** and the **algorithm**.

The research proposes to **architect** this relationship between the **built form** and the **algorithm** by theorizing, generating, taxonomizing, and prototyping a dynamic, reciprocal, symbiotic feedback loop in the form of **Computational Ecosystems (CE)**. Thus, firstly the research aims to establish strong theory for the **CE** by installing a semantic syntax. Secondly, the research aims to generate autonomous, autopoietic, context-aware feedback loops as **CE**, based on the computational framework of Cellular Automata (**CA**). Eventually, the research intends to demonstrate how digital fabrication constraints could also be used as fitness conditions to generate, taxonomize and prototype digitally created built forms in the physical world by conducting empirical evaluations with the use of additive manufacturing. The research also postulates that just like its organic counterpart, a **CE** would be inhabited by biotic and abiotic agents or **elements** (denoted as ϵ in this thesis) which are governed by organizing principles or **economies** (denoted as Ψ in this thesis).

In order to achieve the objectives, the research has implemented a methodology that applies the semantic syntax into performing operational objectives for four distinct types of **CE** based on the cardinalities of their constituent participants (i.e. ϵ and ψ parameters), forming the canonical taxonomies of **CE**. These four distinct taxa, also called as procedural sequences would be **SESE** (with singular ϵ and ψ parameters), **MESE** (with multiple ϵ parameters but a singular ψ parameter), **SEME** (with singular ϵ parameter but multiple ψ parameters), and **MEME** (with multiple ϵ and ψ parameters). To accomplish these operational objectives the research methodology involves implementing three Primary Objectives – **Case Studies**, **Simulations**, and **Prototyping** to all the four procedural sequences (i.e., **SESE**, **MESE**, **SEME**, and **MEME**).

The results generated from the four procedural sequences have then been analyzed independently and comparatively, to identify patterns and anomalies. These analyses have then been further generalized into determining canonical *modus operandi* that can be used in generating, taxonomizing, and prototyping a wide range of **CE** in the future. Furthermore, the conclusions of the research provide a roadmap for possible future research trajectories for the development and implementation of the **Architecture of Computational Ecosystems**.

Abstract (in Spanish)

Industry 4.0 ha transformado rápida y significativamente la industria de la Arquitectura, la ingeniería, y la construcción (*AEC siglas en ingles*) a nivel mundial desde principios del año 2000. El ejercicio profesional de la arquitectura alográfica hasta finales de la última década de los años 1990 ha visto un cambio de paradigma hacia una práctica más autográfica después de la inclusión del **algoritmo** en el desarrollo tradicional de la **forma construida** a través de los medios del **diseño**. Este advenimiento del **algoritmo** en la industria AEC viene con un sólido soporte de infraestructura del aumento sin precedentes del **diseño computacional** y la **fabricación digital**. Al observar la tasa de progreso tecnológico, es muy probable que el diseño computacional se vuelva más autónomo y la fabricación digital se vuelva más impulsada por los datos. Debido a esta mayor probabilidad en la ubicuidad del **algoritmo**, el **diseño** como herramienta para la racionalización de la forma, el espacio y el cerramiento y el diseño como herramienta para la creación de un documento de construcción se encuentra en el umbral de volverse redundante. La investigación y esta tesis se basan en el supuesto de que el **diseño** sería reemplazado por una lógica comunicacional entre la **forma construida** y el **algoritmo**.

La investigación propone diseñar esta nueva relación (o lógica de comunicación) entre la **forma construida** y el **algoritmo** mediante la teorización, generación, taxonomización y prototipado de un bucle de retroalimentación simbiótico, recíproco y dinámico en la forma de **Ecosistemas Computacionales** (*CE siglas en inglés*). Por lo tanto, en primer lugar, la investigación tiene como objetivo establecer una base teórica sólida para la **CE** mediante la instalación de una sintaxis semántica. En segundo lugar, la investigación tiene como objetivo generar ciclos de retroalimentación autónomos, autopoyéticos y sensibles al contexto como **CE**, basados en el marco computacional de Cellular Automata (*CA siglas en inglés*). Eventualmente, la investigación tiene la intención de demostrar cómo las restricciones de fabricación digital también podrían usarse como condiciones de

aptitud para generar, taxonomizar y prototipar formas construidas creadas digitalmente en el mundo físico mediante la realización de evaluaciones empíricas con el uso de fabricación aditiva. Al establecer la teoría, la investigación también postula que, al igual que su contraparte orgánica, una **CE** estaría habitada por agentes o **elementos** bióticos y abióticos (denotados como ϵ en esta tesis) que se rigen por principios organizativos o economías (denotados como Ψ en esta tesis).

Para lograr los objetivos, la investigación ha implementado una metodología que aplica la sintaxis semántica en la realización de objetivos operativos para cuatro tipos distintos de **CE** basados en las cardinalidades de sus participantes constituyentes (es decir, los parámetros ϵ y Ψ), formando las taxonomías canónicas de **CE**. Estos cuatro taxones distintos, también llamados como secuencias procedimentales, serían **SESE** (con parámetros ϵ y Ψ singulares), **MESE** (con parámetros ϵ múltiples pero un parámetro Ψ singular), **SEME** (con parámetro ϵ singular pero parámetros Ψ múltiples) y **MEME** (con múltiples parámetros ϵ y Ψ). Para lograr estos objetivos operativos, la metodología de investigación implica la implementación de tres objetivos principales: **estudios de casos**, **simulaciones** y **creación de prototipos** para las cuatro secuencias de procedimientos (es decir, **SESE**, **MESE**, **SEME** y **MEME**).

Los resultados generados a partir de las cuatro secuencias de procedimientos se analizaron de forma independiente y comparativa para identificar patrones y anomalías. Luego, estos análisis se han generalizado aún más para determinar el *modus operandi* canónico que se puede usar para generar, taxonomizar y crear prototipos de una amplia gama de **CE** en el futuro. Además, las conclusiones de la investigación proporcionan una hoja de ruta para posibles trayectorias futuras de investigación para el desarrollo e implementación de la **Arquitectura de Ecosistemas Computacionales**.

Acknowledgements

To constantly question one's immediate surroundings is the most infinite source of motivation one could have in their quest for information, knowledge, and (maybe even) wisdom. This internal motivation driven by an obsessive curiosity that I have experienced throughout my life would probably have remained unaccomplished if not for the inspiration and encouragement of my research supervisor and director, Ddr. Prof. Alberto T. Estévez. I thank him for providing me with an infinite amount of liberty over these past five years within research and academia. I also want to thank him for providing me with this opportunity without looking at my lack of experience in the subject, and purely based on my skills and expertise. However, with all this freedom and opportunity, and lack of experience in design research it would have been immensely overwhelming if not for the constant source of information and course correction (wherever necessary) from my friend, research supervisor, and co-director Dr. Prof. Pablo Baquero. I thank him for his unfaltering knowledge (in any discipline whatsoever) and persistent guidance through the course of this research.

I sincerely thank Dr. Prof. Dennis Dollens and Prof. Karl Chu for their inspiration in the inception of this research. I also thank all my colleagues of iBAG, especially Prof. Dragos Brescan, Prof. Mohammad Maksoud, Prof. Ariel Valenzuela, and Dr. Prof. Effimia Giannopoulou for the necessary ingredient of creative levity that can only come from collaboration in the otherwise solemn environment of academia.

Throughout this research, I have collaborated across academia and industry. I express my appreciation to all the faculty and staff of academic institutions - *RIT, Kottayam, India; IES College of Architecture, Mumbai, India; CEPT, Ahmedabad, India* for their courage to include experimental research in their curricula. I also thank the team of Studio RAP, especially Ar. Lucas Ter Hall, Ar. Wessel van Beerendonk, Ar. Olav van der Doorn, and Laura van Dixhoorn-Beijens for their sincere collaboration despite the COVID-19 restrictions.

A special thanks to my friends Ar. Sreekanth Sasidharan, Ar. Shailendra Patil, Dastagir Shaikh, Ar. Amit Tarte, Ar. Apoorva Kulkarni, Ar. Nivedita Mehrotra, Ar. Tejashree Lakras, and Ar. Abhishek Sorampuri for their unwavering support while organizing and conducting academic workshops that could offer a platform for the evaluation of this research. I also thank the student participants at all the academic institutions - RIT, Kottayam, India; IES College of Architecture, Mumbai, India; CEPT, Ahmedabad, and the constant support from the students of the Biodigital Architecture Masters from every academic year.

Granted the intellectual support from all the aforementioned people and agencies, the research had a strong infrastructural foundation. However, the research was not funded by any organization or institution. Thus, I suffered repetitive dependency on personal financial commitments while trying to manage a research that demanded unperturbed time and energy. Although it was difficult, my extended family and friends in India have supported me financially throughout the course of this research. I cannot be more thankful for their kindness, and this research and its impact are forever in their debt.

During these past four years, I got married, immigrated to a foreign country and switched careers (from practicing architect to design academician). Throughout this time, I have truly realized the importance of *family* and *home*. I thus express sincerest, heartfelt gratitude towards my wife Ar. Vaishali Rajurkar, my mother (*aai*) Mrs. Niyanta Warang, and my father (*babuji*) Mr. Ashok Warang for bearing with all my shenanigans and wordlessly expressing their unshakable trust in me.

Contents

ABSTRACT	V
ABSTRACT (IN SPANISH).....	VII
ACKNOWLEDGEMENTS	IX
INTRODUCTION AND THE STATE OF THE ART.....	1
1 ON THE RELEVANCE OF COMPUTATIONAL ECOSYSTEMS	2
1.1 <i>The nature of the problem</i>	2
1.1.1 The built form	3
1.1.2 The built form and the design	4
1.1.3 The built form, the design, and the algorithm	8
1.1.4 The built form and the algorithm	12
1.2 <i>The Research – What are computational ecosystems</i>	15
1.2.1 Objectives for Computational Ecosystems	16
1.2.2 Methodological framework for Computational Ecosystems.....	19
1.2.3 Expected projections for Computational Ecosystems	25
1.2.4 The relevance of Computational Ecosystems.....	29
1.3 <i>Structure</i>	30
1.3.1 Structure of the Research.....	31
1.3.2 Structure of the Thesis.....	33
THEORY	35
2 ON THEORETICAL ASSUMPTIONS FOR COMPUTATIONAL ECOSYSTEMS	36
2.1 <i>Establishing the semantics</i>	36
2.1.1 About Ecosystems	39
2.1.2 About Computation	42
2.1.3 About Computational Ecosystems.....	46
2.1.4 About the Implementation of Computational Ecosystems	59
2.2 <i>Applicability of the semantics</i>	65
2.2.1 Applicability in biology, epidemiology and behavioral sciences	66
2.2.2 Applicability in visual arts and design	69
2.3 <i>Cellular Automata as Computational Ecosystem</i>	72
2.3.1 Cellular Automata – John Von Neumann model	74
2.3.2 Cellular Automata – John Conway model.....	82
2.3.3 Cellular Automata – Stephen Wolfram model	88
2.3.4 Cellular Automata – applications in the AEC Industry.....	93
2.4 <i>Theoretical Assumptions for Computational Ecosystems</i>	98

OBJECTIVES.....	103
3 ON THE SEMANTIC SYNTAX FOR THE COMPUTATIONAL ECOSYSTEMS.	104
3.1 <i>Lexical Semantics from theoretical assumptions</i>	104
3.1.1 Element	106
3.1.2 Economy.....	109
3.1.3 Ecosystem.....	112
3.2 <i>Establishing Logical Semantics for operational objectives</i>	115
3.2.1 General Assumptions for all Procedural Sequences (CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME})	118
3.2.2 Single Element Single Economy Ecosystem (CE^{SESE}).....	121
3.2.3 Multi Element Single Economy Ecosystem (CE^{MESE}).....	123
3.2.4 Single Element Multi Economy Ecosystem (CE^{SEME}).....	125
3.2.5 Multi Element Multi Economy Ecosystem (CE^{MEME}).....	127
3.3 <i>Semantic Syntax for the Procedural Sequences</i>	129
METHODOLOGY	131
4 ON THE PROCEDURAL SEQUENCES FOR THE COMPUTATIONAL ECOSYSTEMS	132
4.1 <i>Primary objectives through Procedural sequences</i>	132
4.2 <i>Single Element Single Economy Ecosystem (CE^{SESE})</i>	135
4.2.1 Case Studies	136
4.2.2 Simulations	142
4.2.3 Prototyping.....	150
4.2.4 $CE_{cube-tower}$	158
4.3 <i>Multiple Elements Single Economy Ecosystem (CE^{MESE})</i>	160
4.3.1 Case Studies	161
4.3.2 Simulations.....	168
4.3.3 Prototyping.....	175
4.3.4 CE^{MESE}	184
4.4 <i>Single Element Multiple Economies Ecosystem (CE^{SEME})</i>	186
4.4.1 Case Studies	187
4.4.2 Simulations.....	195
4.4.3 Prototyping.....	204
4.4.4 CE^{SEME}	210
4.5 <i>Multi Elements Multi Economies Ecosystem (CE^{MEME})</i>	211
4.5.1 Case Studies	212
4.5.2 Simulations.....	219
4.5.3 Prototyping.....	228
4.5.4 CE^{MEME}	235
4.6 <i>Procedural sequences for Computational Ecosystems</i>	236
RESULTS.....	238

5 ON THE CONSEQUENCES OF THE COMPUTATIONAL ECOSYSTEMS.....	239
5.1 Results of CE^{SESE}	240
5.2 Results of CE^{MESE}	243
5.3 Results of CE^{SEME}	246
5.4 Results of CE^{MEME}	249
5.5 Conclusive thoughts on the results of all the taxa.....	252
OBSERVATIONS.....	254
6 ON THE INVESTIGATIVE ANALYSIS OF THE COMPUTATIONAL ECOSYSTEMS	255
6.1 Effects of distinct ϵ , Ψ , and N	256
6.2 Examples of possible distinct ϵ , Ψ , and N for CE	258
6.3 Effects of distinct Initial States	260
6.4 Examples of possible distinct Initial States for CE	262
6.5 The inferences of the analysis.....	264
CONCLUSIONS.....	266
7 ON THE PROSPECTIVE PROJECTIONS FOR THE COMPUTATIONAL ECOSYSTEMS	267
7.1 Probable research trajectories in Algorithm-aided-design.....	268
7.2 Probable research trajectories in Additive Manufacturing	272
7.3 Probable research trajectories in Pedagogy.....	278
7.4 Probable research trajectories in Software Development.....	284
7.5 Concluding Statements.....	287
7.5 Concluding Statements (in Spanish)	289
8 BIBLIOGRAPHY.....	291
8.1 References.....	291
8.2 Software and Hardware references.....	302
9 APPENDIX.....	304
9.1 Grasshopper definitions – CE^{SESE}	304
9.2 Grasshopper definitions – CE^{MESE}	306
9.3 Grasshopper definitions – CE^{SEME}	308
9.4 Grasshopper definitions – CE^{MEME}	310

*Introduction
and
The State of the Art*

1 | On the relevance of computational ecosystems

1.1 The nature of the problem

In a nutshell, the doctoral research seeks to theorize, prototype and taxonomize a novel design automation technique termed as **Computational Ecosystems**.

The research derives its semantics and functionalities of Computational Ecosystems from the computational logic of cellular automata. The research also relies on behavioral simulations generated by employing agent-based systems while operating within the design and programming environments of Rhinoceros, Grasshopper 3D, and several of its plugins (both inbuilt and third-party).

Although, the concept of cellular automata forms the theoretical backbone of this research, there are a very many areas of pure and applied sciences that the theoretical foundation borrows from. It is therefore critical to understand the structural framework of the research, pertaining to the hypothesis, objectives, methodology, and expected outcome before diving into the specifics of the literature. It is also necessary to understand the relevance of the research in the age of the 4th Industrial revolution while considering technological advancements in relevant domains and industries. And finally, it is important to establish the projections of the research, so as to understand an overall research trajectory that has been undertaken in the span of the past four years (i.e., the research duration).

Apart from elaborating on the abstract and laying the foundation for the theoretical background, the concluding part of this chapter also elaborates the structure of this Research and the structure of the subsequent Thesis.

1.1.1 The built form

The earliest traces of built forms, which could now be termed as architecture, belong to the Neolithic period. Although all the dwellings of this era are ruins and remains that have been through thorough archaeological scrutiny, much of the knowledge that they provide of the era is of a speculative nature. However, one can deduce basic information of the purpose, materiality and organization of these built forms. The structures, which were built mostly for a residential purpose, built with materials from the surroundings (*roughly within walking distance*) and with the knowledge and technology that was suitable to the era and early human development, have strong traces of a civilization that had quite recently adopted a sedentary lifestyle.

The built forms of this era possess subtle evidences of a civilization on the cusp of a phase transition. Their previous nomadic lifestyle from the Paleolithic and Mesolithic ages had resulted in sufficient wisdom on pattern recognition, letting the Neolithic humans gain rudimentary control over survival. The traditional hunter-gatherer settlements, which were now challenged with growing abundance of food, had an intuitive urge to build shelters. Shelters not just for themselves, but for their surplus food, their young offspring, and their newly acquired belongings such as their tools and containers. Although the essential purpose of the built form was to house the Neolithic human, the primary reason of building was protection. Protection from the weather, from beasts, from pests and from other humans.

A recent study in political economy suggests that the Neolithic economic revolution of agriculture could have been a consequence of private land ownership and not the other way around as was initially assumed and recorded (Bowles and Choi, 2019)¹. This means, that even in the Neolithic age, built forms were not mere shelters serving an objective purpose, but were a symbol of power and prosperity.

¹ Bowles, S. & Choi, J-K. (2019). The Neolithic Agricultural Revolution and the Origins of Private Property. *Journal of Political Economy*, 127(5), pp. 2186-2228.

1.1.2 The built form and the design

Cave paintings stand a strong evidence to the fact that language and documentation predates architecture and construction. However undeveloped the cognitive skills of the early human brain were, the perceptive skills were surely quite highly developed for the time. For example, the paleolithic cave paintings of Lascaux, France (dating back to 15,000 - 17,000 BCE) contain nearly 6,000 figures of animals, humans, and abstract signs. One of the figures, famously known as 'The crossed Bison', as shown in figure 1.1, has a peculiar form of perspective illusion which has been considered quite an advanced technique for the time.



Figure 1.1 – The cross Bison of Lascaux – part of the paleolithic cave paintings (15,000 – 17,000 BCE) (provenance – Marcel Ravidat, at Lascaux, France in 1940). Photo by Paula Kuitenbrouwer (2019).

Source: <https://mindfuldrawing.com/2019/08/07/crossed-bison-of-lascaux-art-study-through-drawing/>

Quite evidently, the Neanderthals were equipped with superior drawing techniques by the time they started building protective forms. However, the earliest traces of documented architectural drawings occur in excavated statues made during 2,200 BCE from the regions of ancient Egypt and Mesopotamia, roughly 12 millennia after the Lascaux cave paintings. As shown in figure 1.2, the statue of emperor Gudea, ruler of Lagash holds an architectural plan on his lap. The plan depicts construction data for the temple of Ningîrsu (ancient Mesopotamian god of hunting, law, scribes, and war), which was commissioned and built under the rule of emperor Gudea.



(a)

(b)

Figure 1.2 – (a) Diorite statue of Gudea (2120 BCE) (provenance – Ernest de Sarzec, at Tell Telloh, Iraq in 1924) – part of the statues of Gudea collection at Musée du Louvre. Photo by Daryl Mitchell (2014).

Source:[https://en.wikipedia.org/wiki/Statues_of_Gudea#/media/File:Gudea,_statue_d%C3%A9di%C3%A9_au_dieu_Ningirsu_\(Louvre,_AO_2\).jpg](https://en.wikipedia.org/wiki/Statues_of_Gudea#/media/File:Gudea,_statue_d%C3%A9di%C3%A9_au_dieu_Ningirsu_(Louvre,_AO_2).jpg) (b) image of the architectural plan of the temple of Ningîrsu on the lap of Gudea - Diorite statue of Gudea (2120 BCE) (provenance – Ernest de Sarzec, at Tell Telloh, Iraq in 1924) – part of the statues of Gudea collection at Musée du Louvre. Photo by Daryl Mitchell (2014). Source: https://www.reddit.com/r/ArtefactPorn/comments/crhobh/statue_of_gudea_of_lagash_with_architectural_plan/

As the field of archeology would evolve in the future and emerging technological upgradation and equipment might shed more clarity in this field, it later might be discovered that the act of drawing architecture indeed precedes 2,200 BCE by centuries. However, it is rather curious to note that while the ancient humans were proficient in the technology of building and adept at drawing, it took them several centuries to develop a graphical documentation for their built forms. Needless to say, it finally did happen and the built form and the design documentation formed an amalgamated process of architectural design and gradually metamorphosed into the entire architecture, engineering and construction (AEC) industry.

When Marcus Vitruvius Pollio wrote '*de architectura*' for his patron, the emperor Caesar Augustus between 30 and 15 BCE, it was a manuscript written as a guide for building projects. Being the oldest surviving treatise on the subject of the theory of architecture, '*de architectura*' has become the first book on architectural theory that has been apotheosized by architects from the medieval era to the modern digital age.

Leon Battista Alberti, later developed Vitruvius's theory into his own ideas on architectural theory written in the treatise '*de re aedificatoria*' between 1443 and 1452 CE. According to Mario Carpo (2011)², "*Alberti's invention of architectural design was crucial in shaping architectural modernity.*" Alberti was also the first in defining perspective images as vestiges of light rays on a surface thereby laying the ground work for architectural documentation being a key aspect in constructing the built form. Fueled by the humanist movement at the peak of the renaissance era, Alberti claimed that architects should not be makers but designers. As a consequence, this helped architecture to transition from its "*autographic status as a craft (conceived and made by artisan builders) into its modern allographic definition as an art (designed by one to be constructed by others).*"³

² Carpo M (2011). *The Alphabet and the Algorithm*. The MIT Press, pp. 10-19.

³ Ibidem

Initiated by Alberti's vision, the rudimentary architectural documentation of the time soon translated into notational systems of scaled architectural drawings and orthographic projections, and the **built form** and the **design** were soon fused into a unified process with two equally important reciprocals.

At the turn of the 19th century, and with the arrival of **the first industrial revolution**, the built form and the design had equally benefitted. The built form saw a revolutionary transformation in the construction practices and technology brought in by the new industrially produced material. The iron industry, glass manufacturing, cement and most notably the groundbreaking innovations in the concrete industry, pushed the built form to become taller and bigger. The design on the other hand, benefitted largely from the industrial innovations in the printing industry. While, the architectural drawings were previously being made manually, the continuous printing press made sure that drawings could now be printed.

With its technological contributions to several industries, the first industrial revolution also triggered the practices of mechanization and mass production of identical objects. The built form and the design both were immensely impacted by this new phenomenon. With cheaply produced identical construction elements such as doors, windows, and staircases, the built form became more monotonous and functional. With the rising issues of housing the ever-increasing workforce, "*functionality took over as the leading standard and the principle ontology*" (Jencks, 1984)⁴. As a corollary, the printing press assisted the design by becoming an enabler in copies of drawings. It became more convenient for the architect to delegate all the different stages and aspects of a conventional construction project to several consultants who would then provide expertise within their scope of work.

Thus, the **built form** and the **design** were faster and more efficient in executing a construction project to house the modern industrious human and the industry.

⁴ Jencks C (1984). *The language of post-modern architecture*. Rizzoli International Publications.

1.1.3 The built form, the design, and the algorithm

The **second industrial revolution** was besmirched by the two world wars, nonetheless the built form and the design thrived and flourished despite the chaos and destruction. Increase in productivity, improved living standards, rapid urbanization within the industrialized countries and improved public health and sanitization triggered large scale construction projects. With such a high demand, the role of the architect as a designer became more prominent and mainstream as the technology enabled the built form and the design to be more independent of its location.

Building materials were no longer location specific, as international logistics were more efficient and sped up. “*The transportation revolution that occurred between 1820 and 1850*” (O’Rourke et al, 2002)⁵ triggered globalization and the global uniformity and availability of construction materials and techniques. Although supported by imperialism and colonization, the built form was transcendental. The design however, had not achieved this feat until the arrival of computers and the digital revolution or the **third industrial revolution**. According to Carpo (2011)⁶, “*architectural design is a purely informational operation, its processes are defined by a specific range of cultural and media technologies.*” The invention of the Internet and the rise of home computers in 1970s and 80s that triggered the conversion of the analog to the digital was the cultural and media revolution that architectural design needed. Design could now be done remotely while the construction was executed and monitored on-site. Although the architect was physically more detached from the site, the design could still be as involved as before. At the turn of the millennium with the internet and the home computer becoming mainstream throughout the world, an architect could now be globally present for their designs. The **built form** and the **design** were equally transcendental and global.

⁵ O’ Rourke K.H. and Williamson, J. (2002). When did globalization begin? *European Review of Economic History*, 6(1), pp. 23-50.

⁶ Carpo M (2011). *The Alphabet and the Algorithm*. The MIT Press, pp. 10-19.

The AEC industry has since caught up with the global socio-economic impact of the digital revolution in the form of out-sourcing. While the built form is constructed sustainably by reducing the embodied energy, the design is generated by collaborations of several consultants around the globe at every stage before, during and after construction. With the insurgence of building information modelling (BIM), a construction project becomes more reliant on the project management consultant (PMC), than the architect. Furthermore, the AEC industry has seen the rise of paperless construction projects, as shown in figure 1.3, where a contractor implements state-of-the-art gadgets and software applications (apps) to execute a project without printing the drawings.



Figure 1.3 – iOS application Autodesk PlanGrid in use. Source: <https://www.plangrid.com/nl/>

A contractor would have the design documentation directly delivered on an iPad application as the stage-wise delivery of drawings in whichever form the construction project dictates. The PMC can then have a much bigger control on the execution of the project and update the architect about all the stages of different parts of the projects simultaneously and dynamically. This new dynamic relationship between the built form and the design has been absolutely unprecedented throughout the history of construction. But the relationship is possible and strongly reliant on the existence of a third element – ***the algorithm***.

Triggered by the ‘*entscheidungsproblem*’ (Hilbert and Ackermann, 1928)⁷ the mid-20th century saw a paradigm shift in mathematics and computation contributed by Alan Turing and John Von Neumann. The idea of modern computation and the thinking machine, which would later develop into artificial intelligence also has its conceptual roots in the works published during this era. Aided by significant technological revolutions of the time by major tech-organizations, the computer had become more personal, mobile, robust and affordable in the late 90s. At the turn of the millennium, the computer was not just mainstream, but was either directly or indirectly responsible for all major social, economic and political reform. However, it did not remain as a personal appliance or gadget, that would have a preset, predetermined mode of operation and use. The computer provided users with the freedom of creating their own computation, that would consequently create new tools and develop new advanced technologies. Moreover, the computer gave humanity a new procedural way of solving problems by means of algorithms.

As history dictates, the AEC industry took some time to incorporate the algorithm in its workflow. But, unlike their predecessors who took millennia or centuries to upgrade and update, the modern digital architect took a mere couple of decades.

⁷ Hilbert, D. & Ackermann, W. (1928). *Principles of Mathematical Logic*. Providence, Rhode Island, USA: AMS Chelsea Publishing, pp. 113-134.

When David Rutten created the program 'Explicit History' in 2007, it was used as a digital interface to create generative algorithms within the environment of CAD 3D modelling software Rhinoceros 3D. Since its arrival and subsequent evolution into 'Grasshopper' the program has become endemic to young architects and designers with limited or no prior knowledge or experience of programming, but willing to design with the aid of algorithms. Grasshopper, Autodesk Dynamo, GenerativeComponents and Archimatix serve as visual programming languages that help architects design their own bespoke, discreet design programs.

Although, the algorithm was in its infancy and couldn't make a substantial contribution to the AEC industry in the early 2000s, the 2010s have seen a considerable rise in architecture realized by the algorithm alone. This sudden surge in the use of algorithms in the last two decades has also been sufficiently supported by the ***fourth industrial revolution*** or industry 4.0

Industry 4.0 is essentially epitomized by non-standardization and decentralized production of objects. The idea of identical repetition of products that marked the last century, has transformed into the idea of mass customization. The production industry has seen a return of bespoke products in the market. However, these products are now produced digitally and not manually. Aided by novel digital fabrication techniques from additive manufacturing to robotic building, the architect has been quite successful in hacking into industrial equipment by means of the algorithm. Design algorithms are already being employed by research groups, universities and architectural practices to analyze, model and build.

It would now be prudent to summarize the journey of the architect - If the advent of the built form equipped the Neanderthals with protection and power, and the addition of the design made the architect truly transcendental, the rise of the algorithm has certainly transformed architecture back from an allographic profession into an autographic craft. Rather than the chief designer, an architect is more of a digital craftsman now.

1.1.4 The built form and the algorithm

After industry 4.0, design research as a self-sustaining industry has evolved considerably by creating strong collaborative workflows between education, academic research and practice-based research. A very many design researchers have been exploring and experimenting the dynamic and yet discrete link that the **algorithm** forms between the **built form** and the **design**. But, because the existing relationship between the built form and design had emerged as a corollary to the contemporary art movements which were prevalent during the third industrial revolution, a lot of above-mentioned design research had a somewhat restricted, rational, functionalistic approach to them.

While criticizing on architectural design approach at the beginning of the industry 4.0, Neri Oxman quoted in her seminal doctoral research thesis that *“the institutionalized separation between form, structure, and material, is deeply embedded in the modernist design theory. Paralleled by a methodological partitioning between modelling, analysis and fabrication, it has resulted in geometric-driven form generation in the early 21st century architecture and such prioritization of form over material has been carried into the development and design logic of CAD”* (Oxman, 2010).⁸ This partitioning is quite evident in BIM. *“BIM has led organizations towards a more integrated process of design, procurement, construction and facilities management within a single contract delivery document and information hub, thereby revolutionizing the construction industry”* (Allen and Shakantu, 2016)⁹. But in the process, the architect has also been categorized into **design architect & delivery architect**, where the latter has gained higher prominence and economic relevance.

⁸ Oxman, N. (2010). *Material-based Design Computation*. PhD Thesis. Massachusetts institute of Technology.

⁹ Allen, C. and Shakantu, W. (2016). The BIM revolution: a literature review on rethinking the business of construction. In: *11th International Conference on Urban Regeneration and Sustainability*, Bilbao: WITconferences pp. 919-930 Available at: <https://www.witpress.com/elibrary/wit-transactions-on-ecology-and-the-environment/204/35716> [Accessed 17 May 2019].

However, with the aid of computational design and digital fabrication, to a certain extent, designers have been successful in creating **algorithms** which can **design built forms** by simultaneously performing modelling, analysis and fabrication. Following her afore mentioned critique on traditional architectural design, with extensive research in the field of Material-based Design Computation, Oxman has paved the way for computationally-enabled form-finding procedures. Many research projects that emphasize on the synthesis of such algorithms, have gained prominence not just in the field of education and academic research, but also in practice-based research.

Figure 1.4 illustrates the computational workflow adopted during the modelling, analysis and fabrication of a research building - Urbach Tower designed and built by the Institute for Computational Design and Construction (ICD) at the Universitat Stuttgart, Germany. The figure shows how an algorithm could generate a streamlined workflow that embeds structural analysis (deformation), Material modelling (CLT utilization) and Fabrication logic (connection angles).

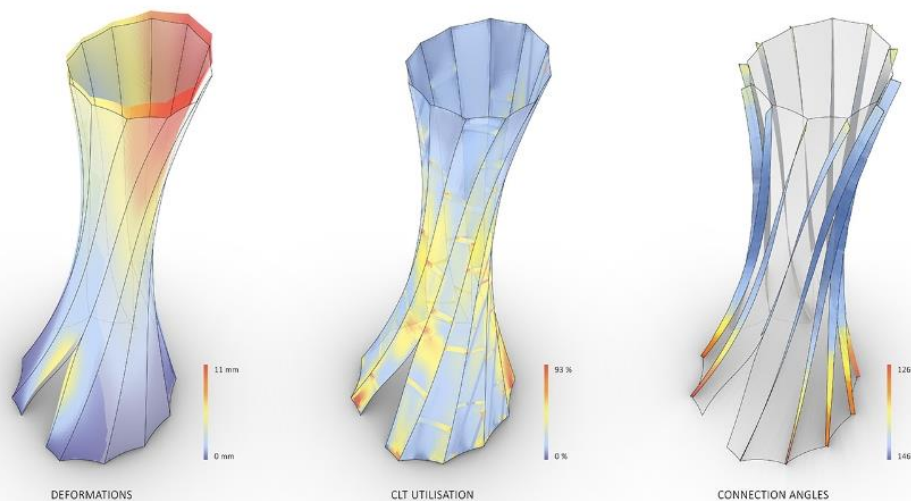


Figure 1.4 – Optimization workflow – Urbach tower – ICD/ITKE, Stuttgart, Germany. Source: https://www.icd.uni-stuttgart.de/img/wp-content/gallery/urbach_process_icd-itke/URBACH-TURM_-process_-03.jpg?__scale=w:1000,h:1000,q:100,t:3

The Urbach Tower (Wood et al, 2020)¹⁰ and many such research projects are testimony to the fact that an algorithm could be used to generate, maintain, optimize, and execute a design process while transcending the methodical separation in the workflow. It is also quite clear from the aforementioned examples, that algorithm has a unique position in the existing ecosystem of the built form and design. Thus, it could be astute to make the following assumptions:

- As digital fabrication becomes more data driven, the role of design in the AEC industry as a mode of generating the construction document becomes more redundant. For example, a robot doesn't require the plan, section and elevation to construct a building, it merely needs the tool-path generated over the model of the building to conduct the fabrication.
- On the other hand, as computational design becomes more autonomous, the role of design as the mediator of a construction project becomes more redundant. For example, the design data for the structure, material, services and equipment will neither be methodically separated into modelling, analysis and fabrication nor will be collected from different specific consultants, but dynamically updated and optimized through cloud data.

As the **algorithm** finds a more intimate dialogue with the **built form**, we are now on an evolutionary threshold of establishing a new symbiotic relationship between the two. Some might argue that with the role of **design** becoming redundant, the role of the designer suffers a similar fate. The argument, however is not rational. With the algorithm replacing the design, the designer would have a more critical, intellectual and an overall autographic position in the AEC industry.

¹⁰ Wood, D., Grönquist, P., Bechert, S., Aldinger, L., Riggerbach, D., Lehmann, K., Rüggeberg, M., Burgert, I., Knippers, J., and Menges, A. (2020). From Machine Control to Material Programming Self-Shaping Wood Manufacturing of a High Performance Curved CLT Structure – Urbach Tower. In: *Fabricate 2020 Making Resilient Architecture*, London: UCL press pp. 50-57 Available at: <https://www.uclpress.co.uk/products/154646> [Accessed 15 Jun. 2020].

1.2 The Research – What are computational ecosystems

What if the **built form** was constructed, monitored and governed by an autonomous, unbiased **algorithm**?

What if this **algorithm** was dynamically constructed, monitored and governed by the **built form**?

The research attempts to establish a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm by making computational design more autonomous and digital fabrication more data driven. The research also attempts to empirically demonstrate a fluid design workflow that performs modelling, analysis and fabrication to generate form, structure and enclosure for an architectural intent.

To implement autonomy in computational design, the research pursues the empirical hypothesis: **Cellular Automata can be employed as a computational framework to generate, taxonomize and prototype design automation algorithms**. These design automation algorithms will be created by using predetermined data meant for an architectural design such as the structure, functional arrangement, topographical data, climatic data, services and equipment.

To prototype the design automation algorithms using digital fabrication the research pursues the empirical hypothesis: **Fabrication data in the form of G-code can be used as a fitness condition to generate, taxonomize and prototype digitally generated built forms**. These built forms will be prototyped through additive manufacturing.

The process of fabricating the **built forms** that could be generated in this way will dictate the writing and re-writing of the **algorithm** that will eventually generate the built form, creating a dynamic, reciprocal, symbiotic loop. The creation, maintenance and evaluation of this feedback loop is the **architecture of computational ecosystems**.

1.2.1 Objectives for Computational Ecosystems

In the context of the research hypotheses established both for the built form and the algorithm, the study pursues the architecture of computational ecosystems by allocating methodical and empirical objectives. To find, theorize, establish, taxonomize, and prototype a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm, the research follows three primary objectives.

The **primary objectives** are as followed:

- Primary objective I (**case studies**) – To understand and evaluate energy flows and nutrient cycles within existing ecosystems. These could also be existing or already established computational ecosystems such as bio-based optimization algorithms. Here, cases of these existing ecosystems will be studied for the interoperability and symbiosis between their constituent agents to establish a theoretical framework for the research.
- Primary objective II (**simulations**)– To simulate these dynamic ecosystems for predetermined biotic and abiotic elements in computational environments to form protective, habitational spaces. The simulations will be performed by implementing the Cellular Automata and Agent Based Modelling. Here, the simulated ecosystems will be taxonomized to understand and evaluate the effect of the biotic and abiotic elements on the algorithm and vice versa.
- Primary objective III (**prototyping**) – To prototype the fabrication strategy for the optimized ecosystems considering a predetermined additive manufacturing technique. The prototyping will be carried out by implementing the fused deposition modelling (FDM) technique. Here, the generated G-code will be provided as a feedback into the simulation of the ecosystem.

All the primary objectives will be considered as research template that drives all the experimentations and findings. However, the research relies strongly on establishing theoretical background to attain the said empirical objectives. Therefore, the research needs to set up **literary objectives** as part of **secondary objectives**.

The **literary objectives** are as followed:

- Literary objective I (**Lexical semantics**) – To apprise terminologies and diction from the fields of biology, ecology, computation, applied mathematics, applied mechanics, fabrication, manufacturing and economics relevant to the research as a literary aid to establish the structure of thought. Here, the lexical semantics of the research will be formed. These terminologies will be further implemented in the research.
- Literary objective II (**Logical semantics**) – To repurpose existing terminologies and diction in the fields of biology, ecology, computation, applied mathematics, applied mechanics, fabrication, manufacturing and economics relevant to the research as a literary aid to establish the structure of process. Here, the logical semantics of the research will be formed. These semantics will be used in the research to transcribe concepts into algorithms.

A key motive behind establishing literary objectives is to familiarise and conceptualize the research in the context of contemporary researches across domains, disciplines and industries. It is also necessary that the terminologies have been widely accepted and have been evaluated in their respective domains. Moreover, their contradictions are also considered before acknowledging them as the core theoretical structure of this research. Also, because the research is based on an ambitious concept that is both futuristic and hypothetical – (**addressing the increasing redundancy of design within the built form and the algorithm with respect to Industry 4.0**) the literary objectives help in grounding the research in concepts and semantics that are well known, proven, and tested.

After setting up the literary objectives and establishing an inventory for the semantics, the research will implement all the primary objectives, and put the hypothesis and the resultant theory to test. This will be accomplished by pursuing three **operational objectives** as part of **the secondary objectives** of the research.

The **operational objectives** are as followed:

- Operational objective I (**Taxonomies**) – To categorically and sequentially assess and catalogue the generated computational ecosystems into taxonomies of possible functioning ecosystems. Here the primary objectives of the **case studies** and **simulations** will be fulfilled to establish functioning computational ecosystems by establishing different types of biotic and abiotic elements in the computational environment.
- Operational objective II (**Evaluation**) – To conduct user tests for the evaluation, trouble-shooting and versioning of taxonomized computational ecosystems. Here the primary objectives of **simulations** will be fulfilled to test functioning computational ecosystems for bugs and errors by conducting workshops with students and professionals of the AEC industry.
- Operational objective III (**Fabrication**) – To prototype the process of fabricating a built form by using the computational ecosystem as an autonomous algorithm. Here the primary objectives of **simulations** and **prototyping** will be fulfilled to fabricate the functioning, taxonomized and bug-tested computational ecosystems into a built form by partnering with the AEC industry serving a functional purpose for computational ecosystem.

The results generated and demonstrated in this way would be successful in establishing a more dynamic, reciprocal, symbiotic relationship between the built form and the algorithm while making computational design more autonomous and digital fabrication more data driven.

1.2.2 Methodological framework for Computational Ecosystems

The research methodology is strongly dictated by the primary, literary and operational objectives. Although planned meticulously it has room to accommodate deviations and anomalies in the experimentations and findings. The methodological framework as illustrated in this subchapter also serves as a roadmap for a design workflow that performs modelling, analysis and fabrication to generate form, structure and enclosure for an architectural intent.

The understanding, segregation and execution of the methodology however relies on establishing key lexical and logical semantics. Considering the fact that these concepts will be further elucidated in the forthcoming chapters (namely, 2| On the theoretical assumptions for the Computational Ecosystems ; 3| On the semantic syntax for the Computational Ecosystems), it becomes necessary to illustrate them concisely. The methodological framework builds up on the conceptual constructs of the trifacta – ***Elements, Economies, and Ecosystems***.

- ***Element*** – although the meaning of element is more open ended, in chemistry an element could be “*defined as one of a class of substances that cannot be separated into simpler substances by chemical means*” (Boyle, 1661).¹¹ In data science, the term data element is an atomic unit of data that has precise meaning or precise semantics. Thus, in the context of this research, elements signify forms, agents and systems that come together to simulate biotic and abiotic behaviors and semantics. These elements could be considered as agents which can’t be further divided into other agents. Examples of an element could be Platonic solids, Archimedean solids, point clouds, passive agents, active agents, cognitive agents, service equipment, structural members or fabrication material.

¹¹ Boyle, R. (1661). *The Sceptical Chymist*. London: J. Cadwell

- **Economy** – in the field of economics, “*economy is defined as a social domain that emphasizes the practices, discourses, and material expressions associated with production, use and management of resources by different agents*” (Paul, 2014).¹² Although, etymologically economy transcends to household management, the contemporary concept of economy as a set of transactions of goods and services expressed in a certain currency is a recent connotation as early as the 20th century. In the context of this research, Economy implies to the fundamental objective of the symbiotic elements as a means of survival, growth and decay. It is thus essential to say that stability in an economy equates to survival, excess leads to growth and insufficiency leads to decay. Examples of an economy could be static structural stability, kinetic structural stability, functional adequacy, functional compatibility, contextual compatibility, climatic optimization, and fabrication constraints. The ideas pertaining to the possibilities of economies as a fitness condition will be further explicated in the forthcoming chapters (namely, 2| and 3|).
- **Ecosystem** - in ecological sciences, ecosystems are “*dynamically interacting systems of organisms, the communities they make up, and the non-living components of their environment. They can be defined as a community made up of living organisms, the biotic elements and non-living components, the abiotic elements. These biotic and abiotic elements interact through nutrient cycles and energy flows*” (Odum, 1971).¹³ The study of Ecosystem in ecology thus involves ‘the flow of energy and materials through organisms and physical environments.’ In the context of this research, an Ecosystem implies to a design automation algorithm that will produce a built environment but also relies upon the morphological, structural and fabrication constraints of the built form.

¹² James, P. (2014). *Urban Sustainability in Theory and practice: Circles of Sustainability*. London: Routledge. Pg 53

¹³ Odum, E. P. (1971). *Fundamentals of Ecology*. Philadelphia: Saunders.

Fundamentally, the elements would serve as the building blocks of the computational ecosystem that represent the biotic or abiotic, while fulfilling survival conditions laid down by the economy inherent to the ecosystem they belong to and coherent with the ecosystems they inhabit. The intricate relationships between the element, economy and the ecosystem serve as a functional template for the methodological framework of the research. These relationships also serve as a theoretical foundation for the empirical results achieved by incorporating cellular automata in generating, taxonomizing and prototyping design automation algorithms.

This relationship and the specifics of its operations will be further entangled in the forthcoming chapters. Additionally, the concluding part of this chapter (1.3 Structure) further elaborates the structure of this Research and the structure of this Thesis, so as to provide a definitive roadmap in terms of the methodological framework.

Figure 1.5 below, illustrates the conceptual relationship of the element, economy and ecosystem in a graphical style. As shown in the figure, in the relevance of the research, the chess pieces serve as an example of elements, chess moves and rules serve as an example of the economy while chess tactics, strategies and gambits serve as an example of the ecosystem.

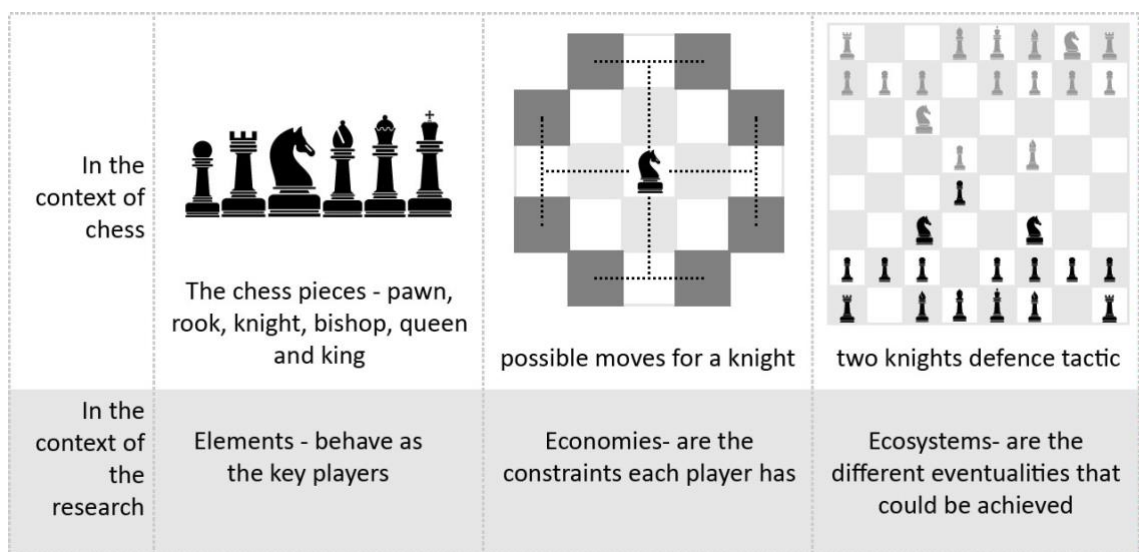


Figure 1.5 – Infographic explaining the concept of element, economy and ecosystem with the example of Chess (the board game). Illustration and graphics by Author (June 2019).

With the key concepts of the element, economy and ecosystem established succinctly, the methodological framework can be clearly rationalized. Moreover, the relationship between the built form and the algorithm, which was introduced and resolved in the hypotheses can be distinctly projected. To achieve the primary and the operational objectives of the research, the methodology follows four procedural sequences. These procedural sequences are devised to extrapolate different possible arrangements within the context of computational ecosystems explained thus far. They rely on the following conceptual constructs:

- An ecosystem can only exist if the element and the economy are established.
- An ecosystem can exist with a single element and a single economy.
- An ecosystem can exist with multiple elements and a single economy.
- An ecosystem can exist with a single element and multiple economies.
- An ecosystem can exist with multiple elements and multiple economies.

This clear distinction of the possible permutable ecosystems is evidently based on the classification systems laid out in Flynn's taxonomy (Flynn, 1972)¹⁴. This distinction helps in sequencing the methodological framework in the form of the procedural sequences (which will be explained in detail on the next page). Moreover, it sets up an operational template that will be helpful in installing a taxonomy for the research. As this distinction between the ecosystems is based on the arrangements of the cardinality of elements and economies, there are four and only four possible types of ecosystems that can be generated employing this methodology. The distinction also disallows any prospects for deviations that could lead to anomalies across taxa.

¹⁴ Flynn, M-J. (1972). Some Computer Organizations and their Effectiveness. *IEEE Transactions on Computers*, C-21(9), pp. 948-960.

The *procedural sequences* are as followed:

- **Single Element Single Economy Ecosystem (SESE)** – An ecosystem that consists of elements of a singular, independent typology who follow an economy that represents a rule set with a singular independent goal. As shown in the figure 1.6 below, the ecosystem could have elements that refer to the morphological norms of a hexahedron, while following an economy that pertains to structural stability. In other words, here an SESE would be represented by cubes, that could form the tallest towers without falling apart.



Figure 1.6 – Infographic illustrating an example of a Single Element Single Economy Ecosystem (SESE). Illustration and graphics by Author (June 2019).

- **Multi Element Single Economy Ecosystem (MESE)** – An ecosystem that consists of elements of multiple, independent typologies who follow an economy that represents a rule set with a singular independent goal. As shown in the figure 1.7 below, the ecosystem could have elements that refer to the morphological norms of a hexahedron and an icosahedron, while following an economy that pertains to structural stability. In other words, here an MESE would be represented by cubes and cuboctahedra, that could form the tallest towers without falling apart.



Figure 1.7 – Infographic illustrating an example of a Multi Element Single Economy Ecosystem (SESE). Illustration and graphics by Author (June 2019).

- **Single Element Multi Economy Ecosystem (SEME)** – An ecosystem that consists of elements of a singular, independent typology who follow economies that represent rule sets with multiple interdependent goals. As shown in the figure 1.8 below, the ecosystem could have elements that refer to the morphological norms of a hexahedron, while following economies that pertain to structural stability and buoyancy. In other words, here an SEME would be represented by cubes, that could form the tallest towers on a liquid without falling apart and without sinking.

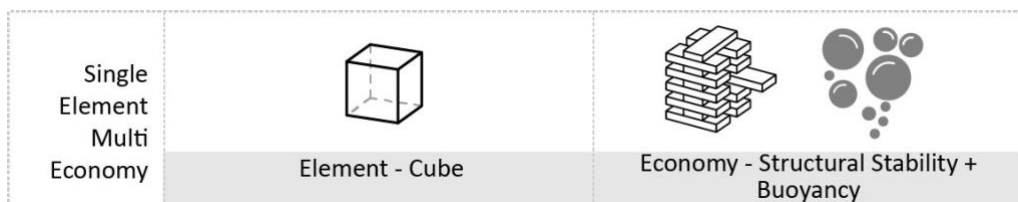


Figure 1.8– Infographic illustrating an example of a Single Element Multi Economy Ecosystem (SEME). Illustration and graphics by Author (June 2019).

- **Multi Element Multi Economy Ecosystem (MEME)** – An ecosystem that consists of elements of multiple, independent typologies who follow economies that represent rule sets with multiple interdependent goals. As shown in the figure 1.9 below, the ecosystem could have elements that refer to the morphological norms of a hexahedron and an icosahedron, while following economies that pertain to structural stability and buoyancy. In other words, here an MEME would be represented by cubes and cuboctahedra, that could form the tallest towers on a medium without falling apart and sinking.

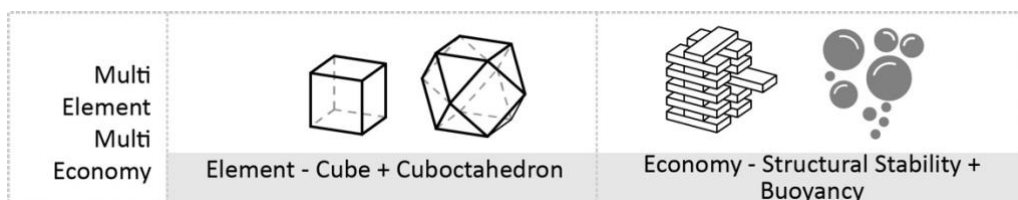


Figure 1.9– Infographic illustrating an example of a Single Element Multi Economy Ecosystem (SEME). Illustration and graphics by Author (June 2019).

1.2.3 Expected projections for Computational Ecosystems

In this way, each taxon represents a unique yet related combination of the cardinality of typologies for the elements and economies. In other words, SESE, MESE, SEME and MEME will be treated as different methodologies to create and to taxonomize the ecosystems.

As outlined in the methodological framework, the four taxa will be subjected to the primary objectives discretely. The objectives will be drawn out considering their relevance for each taxon. For example, an SESE that represents a single element and a single economy will undergo the following primary objectives:

- **Case studies** – Understanding and evaluating energy flows and nutrient cycles within existing ecosystems (both organic and digital) with a single element and a single economy.
- **Simulations** – Simulating dynamic ecosystems for predetermined biotic or abiotic element that follows a singular economy. The simulations will be performed in computational environments to form protective, habitational spaces.
- **Prototyping** – Prototyping a fabrication strategy for the optimized ecosystems considering a predetermined additive manufacturing technique.

Note that the above examples of operational trajectories for the formation of an ecosystem are only specific to an SESE. The other taxa would have slightly variant primary objectives that would be specific to their use cases. An elaboration on the topics of elements, economies and computational ecosystems will be done in further chapters (namely, 2| On the theoretical assumptions for the Computational Ecosystems ; 3| On the semantic syntax for the Computational Ecosystems).

As the basis of the research and this thesis is on the concept of the built form and the algorithm entering a recursive feedback loop, the outcomes are projected to be iterative but relevant. For this reason, the operational objectives viz., Taxonomies and Evaluation become very crucial. Moreover, the empirical nature of the hypothesis further dictates the operational objectives viz., Evaluation and Fabrication. It is important to understand this aspect of the research, because the process of taxonomizing, will generate a plenty of visual information, which will be the iterative variants specific to each taxon. In other words, each iterative variant in each taxon for the specific procedural sequence will be represented by means of images and with information about its attributes such as elements and economies.

As the theoretical foundations of the research are based on the Cellular Automata, the generation of all the iterations in the respective taxa (SESE, SEME, MESE and MEME) would be dependent on “*recursive algorithms*” (Soare, 1996).¹⁵ The iterative variants for each taxon however, would be dependent on the starting condition for each iteration – the Initial State. An elaboration on the Initial States and their relation with the elements, economies and computational ecosystems will be done in further chapters (namely, 2| On the theoretical assumptions for the Computational Ecosystems ; 3| On the semantic syntax for the Computational Ecosystems).

Considering the amount of variations generated by the subtle changes in both the initial states, elements and economies, the projections for the computational ecosystems require a much larger system of classification than the one stated as part of the procedural sequences (SESE, MESE, SEME and MEME). Thus, apart from the implementation of recursive algorithms to undertake autonomy in computational design the research is compelled to implement genetic algorithms to organize the generated and taxonomized ecosystems in the order of their fitness.

¹⁵ Soare R. I. (1996). Computability and Recursion. *The Bulletin of Symbolic Logic*, [online] Volume 2(3), pp. 284-321. Available at: <https://www.jstor.org/stable/420992> [Accessed 8 Apr. 2020].

A clear application of genetic algorithms in the context of this research has been thoroughly expand upon in the further chapters (namely On the consequences in the Architecture of Computational Ecosystems and 6| On the investigative analysis of the Computational Ecosystems). But it becomes important to understand the scope and impact of implementing genetic algorithms before summarizing the relevance of this research. In the context of this research, genetic algorithms have been implemented purely as evolutionary algorithms within the computational environment of Rhino 7 and Grasshopper. Galapagos, which is a default tool-set in Grasshopper has been used to generate results using its evolutionary solver.

As explained in the further chapters (namely 6| On the investigative analysis of the Computational Ecosystems 7| On the prospective projections for the Computational Ecosystems) attempts have been made in setting up a Fabrication workflow that could be adopted to train and prototype Computational Ecosystems for certain procedural sequences (especially MEME) that could conform to multiple economies while fabricating the results of the said Computational Ecosystem. However, these applications demand state of the art digital fabrication tools and assembly, which are not readily available in all academic institutes. Thus, the research has sought collaborations within the AEC industry (as thoroughly elaborated in 7|). Also, it has been sufficiently established in the further chapters how this trajectory could be further pursued to develop Computational Ecosystem as a discipline that expands the fields of computational design and digital fabrication while performing dynamic, reciprocal, symbiotic feedback loops between the ***built form*** and the ***algorithm***.

The research aspires to compute and not computerise existing ecosystems while upholding the design process. It appreciates the very simple yet strong idea that ***Form follows Force*** and ***Force follows Form***. “The research attempts to find ways of ingesting the Architecture of Computation in the Computation of Architecture.”¹⁶

¹⁶ Warang, A. (2017). *Towards the Architecture of Computation*. MS Architecture Thesis. Universitat Internacional de Catalunya (UIC) Barcelona.

Although the afore mentioned explanations, definitions and examples pertaining to the outline of the research objectives and methodology sufficiently elaborate on the approach taken by this research, a graphical illustration of the research objectives, their role in the fulfilment of the hypothesis and their position in the methodology becomes highly essential. In a way, this graphical illustration also underlines the nature of this research – to introduce and elaborate novel concepts with theoretical background (in terms of citing peer-reviewed research), empirical proof (in terms of providing practical examples), and visual representation (in terms of illustrating with relevant diagrams and images). Fig. 1.10 thus illustrates the above idea.

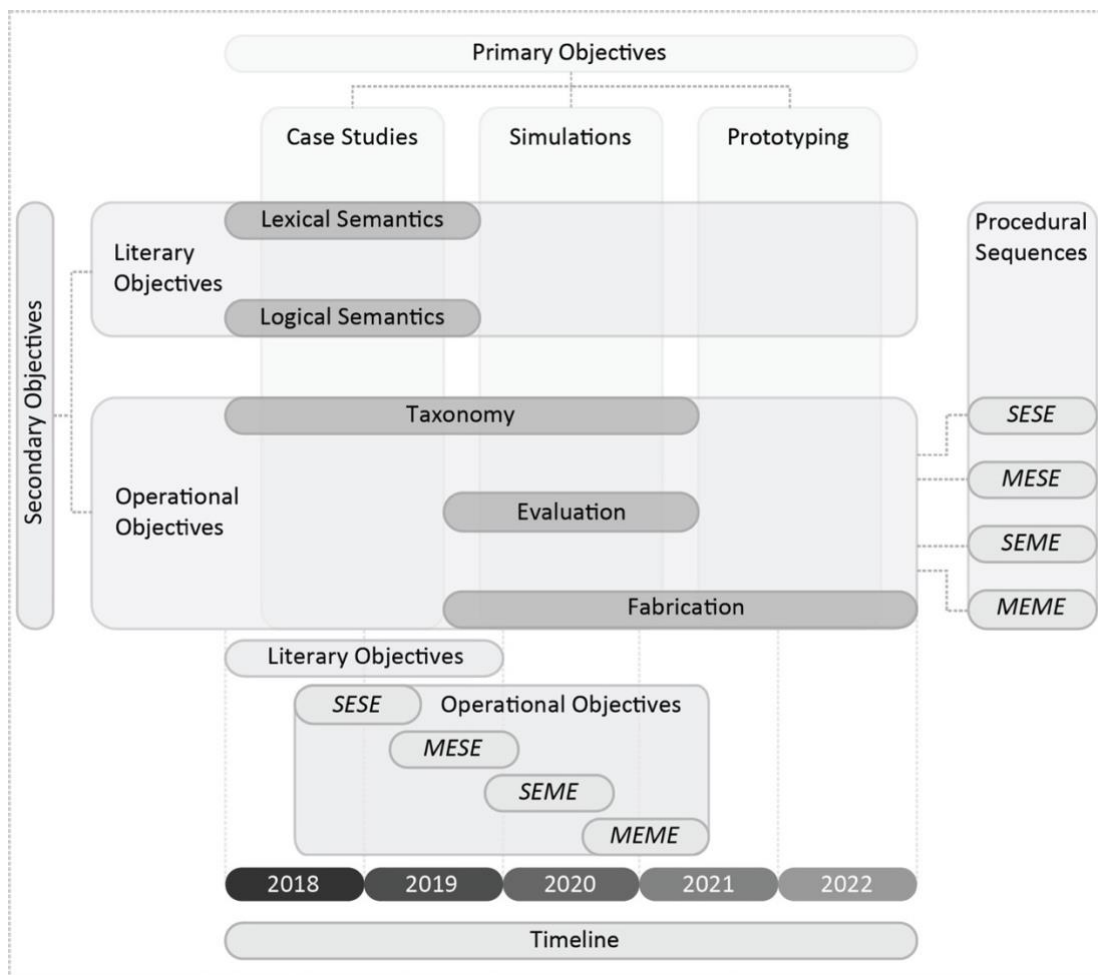


Figure 1.10– Infographic illustrating the inter-relations between the research hypothesis, research objectives, and research methodology with respect to the research timeline. Illustration and graphics by Author (January 2018).

1.2.4 The relevance of Computational Ecosystems

Considering that this thesis documents an attempt to establish a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm, the Architecture of Computational Ecosystems could be considered as a research that furthers into the fields of Computational Design and Digital Fabrication. However, the domain of this research is based in several research fields and industries. For example, the research borrows most of its semantics, and the ideological and theoretical frameworks from several fields such as computational sciences, evolutionary sciences, behavioral sciences and ecology. So, in a way, it also contributes back to these fields by demonstrating their empirical applications for research in the AEC industry.

As one of its hypotheses focuses on practical applications in the field of digital fabrication, the research has significant impact in the domain of additive manufacturing industry. This impact is clearly demonstrated in the upcoming chapters (namely 4| On the procedural sequences for the Architecture of Computational Ecosystems 5| On the consequences in the Architecture of Computational Ecosystems). As discussed later, the research has potential to develop design and fabrication algorithms in the additive manufacturing industry depending on the specificity of the fabrication equipment.

Moreover, the research serves as a significant reference point in the cultural zeitgeist of the Industry 4.0. It provides an operational handbook to demonstrate how design research can restore the autographic status of Architecture with the help of the current Industrial revolution and many more to come. Owing to the nature of its dependencies and impact in several fields the research is relevant to contemporary ideologies of breaking the institutionalised separations of disciplines and curriculum to employ a dialogue between hitherto unheard compatibilities. The research attempts to establish a biodigital dialogue in the field of architecture, while providing a blueprint on how more such dialogues could be established in the future.

1.3 Structure

Now that the research objectives, research methodology and the overarching relevance of the research in the technological zeitgeist is clearly summarized, it becomes more imperative to explain how the entire research and this subsequent thesis is structured. Also, elaborating on the abstract, which helps in laying the theoretical foundation for the research would mean that this would be the perfect space to write about the structure of the thesis before beginning the theoretical intricacies in detail.

The explanation of the structure would thus be bifurcated into the structure of the Research and the structure of the Thesis. While the former elaborates on the means of fulfilling the **operational objectives** (as elaborated in 1.2.1 Objectives for Computational Ecosystems in this Chapter) with the **Procedural Sequences** in the form of the SESE, MESE, SEME and MEME (as elaborated and illustrated in 1.2.2 Methodological framework for Computational Ecosystems in this Chapter); the latter explains how this doctoral thesis helps in conveying the entire process of finding, theorizing, establishing, taxonomizing, and prototyping dynamic, reciprocal, symbiotic relationships between the built form and the algorithm.

In other words, the structure of the research explains how the research objectives were executed across the span of four years, while the structure of the thesis explains how this execution of the research objectives has been elucidated in this doctoral thesis. Moreover, the structure of the research highlights and accentuates upon the various forms of industry experts, collaborators, and research facilities involved in this research. Whereas, the structure of the thesis focuses on specifying the sequence of interactions and involvements with these industry experts, collaborators, and research facilities. The initial timeline established before commencing the research will also be briefly illustrated here; and with it, the undeniably inevitable modifications and the reasons behind them will also be acknowledged.

1.3.1 Structure of the Research

As the research was focused on finding, theorizing, establishing, taxonomizing, and prototyping a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm, the research had to adopt a multi-disciplinary approach. The two disciplines or industries it interacted with were computational sciences and the AEC industry. In the context of the **primary objectives** (as elaborated in 1.2.1 Objectives for Computational Ecosystems in this Chapter), consultations and collaborations with the discipline of computational sciences were responsible for the **Case Studies** and **Simulations**. Whereas, interactions and collaborations within the AEC industry were responsible for the **Case Studies** and **Prototyping**.

Chronologically, the research could be categorized in the following three phases. Note that these phases echo the **primary objectives**:

- **Case studies** – In this phase, both the **literary objectives** (as elaborated in 1.2.1) **Lexical Semantics** and **Logical Semantics** were accomplished. As establishing the semantics formed the theoretical foundation for the research, most of the study was undertaken at the Doctoral School in Architecture of the Universitat Internacional de Catalunya (UIC, Barcelona). However, some semantics needed corrections and revisions, which were incorporated after consulting experts from the discipline of computational sciences for the **Simulations** and experts from the AEC industry for the **Prototyping**. These modifications have been updated in this thesis and will be explained in further chapters (namely, 2| On the theoretical assumptions for the Computational Ecosystems ; 3| On the semantic syntax for the Computational Ecosystems). As some case studies, especially the ones that required clear understanding of evolutionary sciences and ecology could not be done solely in UIC, advice and guidance of experts from relevant fields and disciplines was sought to make the research exhaustive and profound.

- **Simulations** – In this phase, two of the three **operational objectives** namely **Taxonomies** and **Evaluation** were executed for all the **Procedural Sequences** which are, SESE, MESE, SEME and MEME. Although the meticulous taxonomical classification gives a methodological advantage to study all possibilities of Computational Ecosystems, the process followed for the simulation of each taxon remains the same. As these simulations were done in different phases, this part of the research is the most distributed in terms of time and space. For example, majority of the computational logic was set up at UIC. However, the simulations were validated and verified for bugs by conducting workshops for students of Architecture and Design by partnering with various universities in India. These universities were as follows : Department of Architecture, Rajiv Gandhi Institute of Technology (RIT, Kottayam) ; Indian Education Society's College of Architecture, Mumbai (IES COA, Mumbai). The specific results collected during the workshops are documented thoroughly in the upcoming chapters (namely 4| On the procedural sequences for the Architecture of Computational Ecosystems 5| On the consequences in the Architecture of Computational Ecosystems).
- **Prototyping** – In this phase, two of the three **operational objectives** namely **Evaluation** and **Fabrication** were executed for all the **Procedural Sequences** which are, SESE, MESE, SEME and MEME. In this phase, as the algorithm was developed, evaluated and tested for bugs, its application in generating a built form was focused. To prototype the Computational Ecosystems, a research stay was undertaken at Studio RAP, Rotterdam (<https://studiorap.nl/#/>) which is an advanced Architectural practice that specializes in Robotic construction. Here, the research was prototyped by using a 6-axis robotic arm – **KUKA P6** while performing additive manufacturing with a modified pneumatic extrusion of pottery clay. Various types of clay with different composition of grog and different drying processes were also experimented for optimization. The specific results collected during the research stay are documented thoroughly in the upcoming chapters.

1.3.2 Structure of the Thesis

After the completion of the aforementioned stages of **Case Studies**, **Simulations** and **Prototyping**, the research was compiled into this thesis. The structure of this thesis is also quite chronological in the sense that most of its chapters correspond to the order and sequence in which the research was done.

As the **Procedural Sequences** in the form of SESE, MESE, SEME and MEME had chartered the entire research trajectory, there were two approaches in which this thesis could have been structured. The first approach would be where all the theories, methodologies, observations, and conclusions would be done individually for each Taxon. Whereas, the second approach would be where all the Taxa would be explained categorically for all their theories, methodologies, observations, and conclusions. As the research also focusses on creating a feedback loop between the **built form** and the **algorithm**, a hybrid approach that involved an optimized amalgamation of the above two approaches had to be adopted.

Thus, the thesis is structured in such a way that the theory is laid down for all the taxa collectively in chapter 2 | On theoretical assumptions for Computational Ecosystems and chapter 3 | On the semantic syntax for the Computational Ecosystems. Whereas chapter 4 | On the procedural sequences for the Architecture of Computational Ecosystems and chapter 5 | On the consequences in the Architecture of Computational Ecosystems which illustrate the methodology and the results for the **Procedural Sequences** are elaborated for the four Taxa individually. The observations and conclusions however, which are elucidated in chapter 6 | On the investigative analysis of the Computational Ecosystems and chapter 7 | On the prospective projections for the Computational Ecosystems are illustrated and explained for all the taxa collectively. These clear distinctions in the documentation of the key aspects of the research in this thesis provide a more meticulous and direct approach in which the research methodology can also be used and replicated for further researches.

*Theorizing, Taxonomizing and Prototyping operational Ecosystems in
Computational environments*

As chapters 4| and 5| (which illustrate the methodology and the results in the form of the *Procedural Sequences* elaborated for the four Taxa) would be demonstrating most of these taxa graphically in the form of sequential images at every periodic stage of their respective ecosystems, these chapters would be filled with a lot of graphical information without any considerable textual explanations. However, as these sequential images are a result of visual programming performed in Rhinoceros and Grasshopper3D and several of its related plugins, the visual programming scripts have been attached in additional annexures at the end of this thesis.

These annexures will also serve as an empirical database for all the procedural sequences that form a bulk of this research. As most of these scripts were developed by the author by conducting student workshops in several universities in India (as explained in 1.3.1), these scripts have also been made available online on the author's website (<https://angadwarang.wixsite.com/2021/resources>).

Theory

2 | On theoretical assumptions for Computational Ecosystems

2.1 Establishing the semantics

As a corollary to the previous chapter, which established the relevance of this research, and introduced the hypothesis, objectives, methodology and expected results, this chapter elucidates the theoretical assumptions for the research.

The **built form** and the **algorithm** have been independently theorized to a considerable degree, often aided and abetted by design (as explained in chapter 1 | On the relevance of Computational Ecosystems) by imminent theorists, historians and researchers so far. While the theory for the former has a lot of empirical presence, for example in the AEC industry, the latter has been assiduously theorized in the field of computer science, more importantly in the theory of computation. There are lots of researches conducted in the field of architecture under the umbrella of design research during the advent of the third industrial revolution that have borrowed theoretical foundations from the field of computation. A unified theory however, that could correlate the roles of the **built form** and the **algorithm** in either field is rather insufficient.

The research on the other hand, based on the significant relevance of the fourth industrial revolution, relies strongly on the informed assumption that the built form and the algorithm would be co-existent and symbiotic without the existence of design in the near future. The research also claims that this relationship would be dynamic and reciprocal, further diminishing the role of design in the AEC industry. These claims put the research in the dangerous area of sensationalism. If not elaborated upon, it further demeans all the successive **operational objectives** that are prototyped and taxonomized in the course of this research.

Thus, in order to refrain from the aforementioned sensationalism, the research is obligated to form its own theoretical foundation. These foundations would be laid by establishing the semantics related to computational ecosystems and the cellular automata that already exist in the field of computational theory.

Computational theory, on the other hand has often been establishing its theoretical foundations on many natural systems and biological processes. Owing to the fact that computation primarily revolves around problem solving, bio-inspired algorithms have been frequently employed to solve complex computational problems in a simple and intuitive manner. In fact, the early 21st century has seen a considerable rise in the research and implementation of bio-inspired algorithms. These algorithms essentially begin with the hypothesis that an analytical study of the behaviors of natural systems and biological processes of some exceptionally intelligent biological specimen can give us a thorough and accurate understanding of how to translate their behavioral nuances into algorithmic models. These algorithmic models are then tested, versioned and rectified to imitate the natural systems or biological processes in an inert, non-biological, computational environment. After successful or unsuccessful imitation of the natural systems these algorithms are then extensively theorized to be applied into the field of computation.

As the computational power keeps increasing exponentially (aided by the assumptions and conclusions of Moore's Law), the above-mentioned process has seen a phenomenal advancement in the past couple of decades. Researchers have long started analytical studies of the tropic dynamics of ecosystems and translating the energy transactions between biotic and abiotic elements into algorithmic models. The concept of computational ecosystems, thus is not completely novel and has the reputation of being a theoretically established phenomenon in the field of computation (Parpinelli and Lopes, 2014).¹⁷

¹⁷ Parpinelli R. S. and Lopes, H. S. (2014). A computational ecosystem for optimization: review and perspectives for future research. *Memetic Computing*, 7(1), pp. 29-41.

It would thus be quite judicious to lay the theoretical foundations of this research on the theoretical review done in the work of Parpinelli and Lopes. It would also be logical to base the semantics of this research on the aforementioned work, and its theoretical foundation.

However, before establishing the concepts of computational ecosystems as theorized by Parpinelli and Lopes, this thesis would lay down the semantics for the research in terms of developing a reciprocal relationship between the built form and the algorithm. In other words, it seeks to establish the theoretical basis of Ecosystems, Computation and Computational Ecosystems while asking the following questions:

- **About Ecosystems** or Which ecosystems are being referred to? – Here the thesis would be elaborating upon the specific concepts and terminologies the research borrows from the field of ecology and the overall constraints in understanding what an ecosystem is and how to limit it.
- **About Computation** or Which computational environments does the research operate in? – Here the thesis would be elucidating algorithms and computational systems that are inspired from the field of biology while determining those, that this research relies upon for its theoretical framework.
- **About Computational Ecosystems** or What does a computational ecosystem exactly mean? – Here the thesis would be elaborating the concept of computational ecosystems as theorized by Parpinelli and Lopes, in the context of establishing a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm.

The answers to the above questions would lead in establishing the theoretical foundations for the research and help in reducing the sensational claims of the research hypothesis mentioned previously.

2.1.1 About Ecosystems

After having rejected the previously established terms ‘complex organism’ and ‘biotic community’, Prof. A G Tansley settled with the concept of ‘ecosystem’ while drawing analogies with physics – “*including not only the organism complex, but also the whole complex of physical factors*” (Tansley, 1935)¹⁸. Tansley, in his field of study – ecology, goes on to say that, “*though the organisms may claim our primary interest, when we are trying to think fundamentally, we cannot separate them from their spatial environment, with which they form one physical system.*”

He further suggests that these ecosystems could be considered as basic units of nature, where the organic parts (organisms) and inorganic factors have constant interchanges of various kinds amongst them. These ecosystems, he says are of various sizes ranging from the universe down to the atom. This creates a considerable obstacle in the identification and analysis of specific ecosystems. For this problem, he suggests a mundane mental isolation of the systems. He says, “*these isolates, become the actual objects of our study, whether the isolate be a solar system, a planet, a climatic region, a plant or animal community, an individual organism, an organic molecule or an atom. Actually, the systems we isolate are not only included in larger ones, but they also overlap, interlock and interact with one another.*”

The **built form** has a similar property of being a conceivably boundless entity. It could range from a tiny piece of jewellery (in terms of jewellery design) to an entire metropolitan region (in terms of urban planning) and everything within that spectrum. Apart from the theoretical background of defining an ecosystem, Tansley’s methodological framework of isolating an ecosystem conceptually, is quite crucial for this research in stating that the idea of the built form could be similarly isolated into different built forms across scales or **design across scales**.

¹⁸ Tansley A.G. (1935). The use and abuse of vegetational concepts and terms. *Ecology*, 16(3), pp. 284-307.

The idea that “*inorganic factors such as drought, fire, salt, cold, or similar, being considered as important and distinct determinants for the growth, survival and maintenance of life*” was prevalent since as early as the late 19th century (Warming, 1895).¹⁹ The systemic differentiation and the semantic designation into biotic and abiotic elements, however, occurred much later.

The research focuses on understanding the distinct roles played by the biotic organisms and the abiotic factors in forming a symbiotic relationship that is much crucial in the existence of the ecosystem. It thus relies heavily on the understanding and explanations in the seminal book ‘The economy of Nature’ (Ricklefs, 2008).²⁰ Ricklefs points out that we often mention the living and the nonliving as opposites. But although we can easily discern, “*life does not exist in isolation from its abiotic environment. Life depends on the physical world.*” He further claims that, “*many conditions favorable for the development and maintenance of life rely on the activities of living organisms.*” In the context of this research, a similar co-existence for the **built form** and the **algorithm** needs to be formed.

The most important distinction however, that Ricklefs points out between the biotic and the abiotic is that the former have a more purposeful existence over the latter. He says that, “*their structures, physiological processes, and behaviors, shaped by evolutionary responses to natural selection, are directed toward procuring energy and resources that are ultimately used to produce offspring.*” (Ricklefs, 2008).²¹ This subtle detail clearly dictates the research to realize that both the built form and the algorithm are lifeless, purposeless abiotic concepts. The research, however, needs to embed some purpose, some function, some **life** that would serve as a driving force for the symbiotic relationship between the built form and the algorithm.

¹⁹ Warming, E. (1895). *Plantesamfund - Grundtræk af den økologiske Plantegeografi*. Copenhagen: P.G. Philipsens Forlag, 335 pp.

²⁰ Ricklefs, R. E. (2008). *The Economy of Nature*. 6th ed. New York: W. H. Freeman and Company, 620 pp.

²¹ Ibidem

While laying down the foundations of ecology by defining what an ecosystem would be, Tansley postulates that “*organization is the imminent outcome of the interactions and consequent mutual adjustment of the components of an ecosystem*” (Tansley, 1935)²². In other words, he says that the biotic and abiotic elements of an ecosystem are inadvertently striving for a certain state of dynamic equilibrium (Phillips, 1931)²³. He further theorizes that if an organization does not occur, an incipient system breaks out. There is in fact a kind of natural selection of incipient systems, and those which can attain the most stable equilibrium survive the longest.

As a corollary to Tansley, Ricklefs further theorizes that, ecological systems are governed by basic physical and biological principles – they obey the laws of physics; they exist in dynamic states ; and they evolve over time. He says, “*because life is so special, it exists in equilibrium with its environment. But life is not a perpetual motion machine. What the organism loses to its surroundings, is not returned to it for free. The organism must procure energy or materials to replace its losses. To do this, it must expend energy. It must replace the lost energy by metabolizing food or stored reserves, which it must expend energy to capture and assimilate. Thus, the price of maintaining a living system in a dynamic state is **energy**.*” (Ricklefs, 2008).²⁴ Hence, in the context of the research, the dynamic, reciprocal, symbiotic relationship between the built form and the algorithm would have to be realized by creating a system that conforms to the following assumptions:

- The system is inhabited by elements that have purposeful existence such as **life**.
- The system has a specific currency for the state of equilibrium such as **energy**.

²² Tansley A.G. (1935). The use and abuse of vegetational concepts and terms. *Ecology*, 16(3), pp. 284-307.

²³ Phillips, J. (1931). The Biotic Community. *Journal of Ecology*, [online] 19(1). p 1-24. Available at: www.jstor.org/stable/2255934 [Accessed 12 May 2021].

²⁴ Ricklefs, R. E. (2008). *The Economy of Nature*. 6th ed. New York: W. H. Freeman and Company, 620 pp.

2.1.2 About Computation

What are the fundamental capabilities and limitations of computers? – Mathematical logicians, while exploring the meaning of computation have been asking this question since the 1930s. After almost a century worth of technological advancements, now this question is being answered not just with theoretical hypothesis but also with empirical proofs. Computational theory and the field of computation that emerges out of the above question is further trifurcated into “**Complexity theory** – *classifying problems as easy and hard to solve* ; **Computability theory** – *classifying problems as solvable and not solvable* ; **Automata theory** – *defining properties and applications of mathematical models of computation* “(Sipser, 2006)²⁵. Although the trifurcation provides a detailed path into the study of computation, the overarching concept of problem solving remains integral to the application of computation and answering the above-mentioned question. Moreover, finding the best solution from all possible solutions becomes a highly pursued objective amongst all computational models. In other words, optimization is one of the most commonly sought computational pursuit.

Optimization problems can either be deterministic or stochastic in nature. “*While the deterministic problems require enormous computational efforts and tend to fail as the problem size increases, bio inspired stochastic optimization algorithms could serve as computationally efficient alternatives for optimization*” (Binitha and Sathya, 2012)²⁶. As the relationship between the built form and the algorithm operates within the AEC industry in the context of the research, it’s primary goal would be to attain optimization at various scales of design, thus making the implementation of **bio inspired stochastic optimization algorithms** highly essential.

²⁵ Sipser, M. (2006). *Introduction to the Theory of Computation*. 2nd ed. Boston: Thomson Course Technology, 431 pp.

²⁶ Binitha, S. and Sathya, S. S. (2012). A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2), pp. 137-151.

Computational problem-solving and optimization models have often been derived or inspired from natural bio-based systems. *“The abundant diversity, dynamism, robustness, and complexity in these ecosystems always finds the optimal solution to solve its problem maintaining perfect balance among its components”*. This is the thrust behind bio inspired computing. More importantly, the concept of drawing inspirations from a bio-based system underlines the fact that a purpose-based system (considering that all bio-based systems have a predominant goal of survival and reproduction) has more profound methods of optimization. *“Within all the possible relations that could be established in an ecosystem, cooperation between entities is a universal mechanism”* (Chen and Zhu, 2008).²⁷

This cooperation could be within the species - homogeneous cooperation (also called social evolution), or between species - heterogeneous cooperation (also called symbiosis). Chen and Zhu describe in their research how a particular type of Swarm Intelligence (SI) algorithm (which in itself is based on the homogenous cooperation) – Particle Swarm Optimization (PSO) has major drawbacks of converging prematurely and not exhibiting sufficient population diversity. To solve this problem, they theorize that an ecosystem-based computer simulation model should not only adopt the social evolution perspective (homogeneous cooperation) but also the symbiosis theory (heterogeneous cooperation). Thus, their proposed model PS²O (so called as it contains two hierarchies and is based on the PSO) injects fitness conditions pertaining to individuals of different species apart from the rules set up for individuals of the same species as per the canonical PSO algorithm (Chen and Zhu, 2008).²⁸

The relationship between the built form and the algorithm that the research seeks, similarly needs to be based on the combination of the **homogenous and heterogenous cooperation** established in the PS²O.

²⁷ Chen H. and Zhu Y. (2008). Optimization based on symbiotic multi-species coevolution. *Applied Mathematics and Computation*, 205(2008), pp. 47-60.

²⁸ Ibidem.

As Parpinelli and Lopes have theorized, “*computational problem-solving methodologies involve two branches: exact methods and (meta-)heuristic methods. The latter has proven to be efficient in solving hard and complex optimization problems, particularly where traditional methods fail. Bio-inspired algorithms are such meta-heuristics that imitate the intricate strategies of nature as many biological processes can be thought of as optimization processes*” (Parpinelli and Lopes, 2014).²⁹

“*Ant Colony Optimization (ACO) algorithm*” is one such example that studies and extrapolates the homogenous cooperation amongst ants as a novel approach to solve stochastic combinatorial optimization problems (combinatorial optimization problems that are based on randomness or uncertainties such as minimum spanning tree and traveling salesman problem) (Dorigo, Maniezzo, and Colorni 1996).³⁰ The algorithm exploits the phenomenon of Stigmergy (Grassé, 1959)³¹. “*Stigmergy refers to the indirect communication amongst self-organizing emergent systems via individuals modifying their local environment.*” (Binitha and Sathya, 2012)³². In the case of the ACO, stigmergy is displayed by the intelligence of the ants in depositing a pheromone trail to communicate with other ants which then make a probabilistic decision to form the shortest walk between the ant’s nest and the food source.

The relationship between the ***built form*** and the ***algorithm*** that the research seeks, would require a similar ***stigmergy*** between the participating biotic and abiotic elements. This stigmergy could either be communicated as part of a homogenous cooperation or a heterogenous cooperation or both. The information communicated by the stigmergy could vary depending on the optimization strategy adopted by the ecosystem.

²⁹ Parpinelli, R. S. and Lopes, H. S. (2014). A computational ecosystem for optimization: review and perspectives for future research. *Memetic Computing*, 7(1), pp. 29-41.

³⁰ Dorigo, M., Maniezzo, V. and Colorni A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26(1), pp. 1-13.

³¹ Grassé, P.-P. (1963). Les phénomènes sociaux chez les Animaux. *Cahiers de l’Institut de Science économique appliquée*. Suppl. 139, V, pp. 7–23.

³² Binitha, S. and Sathya, S. S. (2012). A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2), pp. 137-151.

Thus, continuing on the theoretical groundwork laid down by Sipser in the field of Computational theory (i.e. establishing the fundamental theoretical constructs of the Complexity theory, the Computability theory and the Automata theory), it becomes essential for the research to focus on understanding the existing assumptions and biases in the field of computation and establish an operational vocabulary before identifying intelligent biological organisms and ecosystems and further attempting to perform analytical study of the behaviors of natural systems and biological processes in these biological organisms and ecosystems.

In the dynamic, reciprocal, symbiotic relationship that the research seeks to develop between the built form and the algorithm, optimization would become both the principal objective and the means of achieving this objective. This dynamic, reciprocal, symbiotic relationship that has previously been theorized to be the design of a system (across scales) that conforms to specific rules (as explained in 2.1.1 About Ecosystems) would essentially have to be a **bio inspired stochastic optimization algorithm**. Although the composition and creation of this algorithm would be a translation of the behaviors of natural systems and/or biological processes within intelligent biological organisms, it could also be derived from the energy interactions amongst biological organisms. In other words, the recreation of the natural strategies that the algorithm attempts would be from both organisms and groups of organisms (i.e. homogenous and heterogenous cooperation).

Hence, in the context of the research, the dynamic, reciprocal, symbiotic relationship between the built form and the algorithm would be realized by creating **bio inspired stochastic optimization algorithms** that conform to the following criteria:

- The inhabitant/participant elements would exhibit both **homogenous and heterogenous cooperation**.
- The specific currency for the state of equilibrium would be communicated using concepts similar to **stigmergy**.

2.1.3 About Computational Ecosystems

After answering important questions regarding the scope and relevance of the vast field of ecosystems and computation in the context of this research, it would now be reasonable to expand upon the concepts of computational ecosystem by Parpinelli and Lopes. In other words, it would now be prudent to ask and answer the question,

What does a **computational ecosystem** mean?

And more importantly,

- How does a computational ecosystem help in establishing a dynamic, reciprocal, symbiotic relationship between the **built form** and the **algorithm**?
- How does its implementation result into **computational design** becoming autonomous and **digital fabrication** becoming data driven?
- How does it represent a fluid design workflow that performs **modelling**, **analysis** and **fabrication** simultaneously to generate form, structure and enclosure for architectural intent?

Answering the above questions would help the research in establishing the theoretical framework and the resultant semantics in alignment with the research hypothesis, apart from helping reduce the sensationalistic claims (as mentioned in 2.1).

Moreover, establishing the theoretical structure, describing the algorithmic logic, and providing examples of existing computational ecosystems would not only help in fulfilling the literary objectives of the research but also assist the research in forming syllogistic rationale into the operational objectives (as mentioned in 1.2.1) while providing a crucial insight in fulfilling the logical hypothesis (as explained in 1.2).

To unify the concepts of computation and ecosystem into a computational ecosystem, however, the research strongly bases its argument on the theoretical unification across the fields of mathematics, computation, ecology and biology. Although inspiration from nature has played a central role in the field of computation since the early 21st century (as mentioned in 2.1.2), **Natural Computing** which emerged in the same era can be considered as a significant benchmark.

“Natural Computing is the field of research that, based on or inspired by nature, allows the development of new computational tools which could be software, hardware or wetware, for problem solving. These computational tools could lead to the synthesis of natural patterns, behaviors and organisms, and may result in the design of novel computer systems that use natural media to compute” (de Castro, 2007)³³. He further clarifies that Natural Computing testimonies against the specialization of disciplines thus suggesting a unified theory that attempts to merge biology and computation. It would have the following areas of investigation –

- **Computing inspired by nature** – *“This area involves development of problem-solving techniques, especially computational tools in the form of algorithms by implementing inspirations from nature.”*
- **Simulation and emulation of nature by means of computing** – *“This area involves synthetic processes aimed at creating patterns, forms behaviors and organisms by mimicking natural phenomena thus increasing our understanding of nature.”*
- **Computing with natural materials** – *“This area corresponds to the use of natural materials to perform computation, that could substitute or supplement the current silicon-based computers.”* (de Castro, 2007)³⁴

³³ De Castro, L.N. (2007) *Fundamentals of Natural Computing – basic concepts, algorithms, and applications*, Boca Raton: Taylor & Francis group, 638 pp.

³⁴ Ibidem

Although bio inspired optimization algorithms are quite effective within the limited computational time and resources, they become less efficient when the problem is scaled or made more complex. This problem has been sensationally theorized and documented as the '**curse of dimensionality**' (Bellman and Dreyfus, 2003)³⁵, which means that "*the efficacy of bio inspired optimization algorithms recedes exponentially as the dimensionality of the algorithm increases.*" At these higher dimensionalities (often described with the theoretical explanation of hyperspace), firstly, the solution space of a problem increases exponentially i.e., the optimization returns a lot of variance in the solution space and secondly, the characteristics of the problem change drastically i.e., the starting variables may start becoming incompatible or might need additional properties to still be considered in the optimization. In the context of this research, it is beyond the bounds of computational possibility that the optimization would demand an exponential increase in the dimensionality, however, nonetheless a more efficient and robust optimization system needs to be employed.

To counter the afore mentioned curse of dimensionality, Parpinelli and Lopes proposed **Hybrid Bio-inspired Systems** (HBS) which are categorized into the natural computing's first area of investigation (that is computing inspired by nature). Here, an "*HBS employs several biological phenomena that are already usually implemented in bio-inspired optimization algorithms, but additionally it also adopts several ecological phenomena into building computational tools to solve complex problems*" (Parpinelli and Lopes, 2012).³⁶

As the HBS serves a distinction from the usual bio-inspired optimization algorithms while countering the afore mentioned *curse of dimensionality*, the dynamic, reciprocal, symbiotic relationship between the built form and the algorithm would be realized by creating a **Hybrid bio inspired stochastic optimization system**.

³⁵ Bellman, R.E. and Dreyfus, S.E., (1962) *Applied Dynamic Programming*, London: Oxford University, 362 pp.

³⁶ Parpinelli, R. S. and Lopes, H. S. (2012). Biological plausibility in optimisation: an ecosystemic view. *International Journal of Bio-Inspired Computation*, 4(6), pp. 345-358.

Considering **computing inspired by nature**, a very many optimization algorithms have been developed, tested and implemented by studying biological phenomena such as “*the evolution of the species, the behavior of social groups, the dynamics of the immune system, the strategies of search for food and in the ecological relationships of different populations.*” However, “*the ecological concepts of habitats, ecological relationships and ecological succession had not been explored computationally in a context of optimization*” (Parpinelli, 2013)³⁷ until the explorations of Parpinelli and Lopes to develop an “*ecology-inspired algorithm to solve numerical optimization problems termed as ECO*” (Parpinelli and Lopes, 2011).³⁸

With the proposal of ECO, Parpinelli and Lopes laid the theoretical foundations of Computational Ecosystems while conforming to the theoretical framework of natural computing. ECO was designed to be a cooperative search algorithm constituting of populations of individuals, where each population evolves based on its independent search strategy, while also interacting with other populations implementing the ecological concepts of habitats, ecological relationships and ecological successions. Thus, forming a cooperative search algorithm that characterizes homogenous and heterogenous cooperation similar to a PS²O algorithm (as mentioned in 2.1.2).

Hence, the afore mentioned HBS are developed with and defined by cooperative search concepts. “*These hybrid strategies are expected to provide more efficient and flexible approaches to solve complex problems that would be very difficult to solve with simple methods. With such diversity of search strategies and the advantages of applying them cooperatively, it is possible to establish an analogy with the dynamics of biological ecosystems.*” (Parpinelli and Lopes, 2012).³⁹

³⁷ Parpinelli, R. S. (2013). *An Ecosystemic View for Developing Biologically Plausible Optimization Systems*. PhD Thesis. Federal University of Technology Paraná.

³⁸ Parpinelli, R. S. and Lopes, H. S. (2011). An Eco-inspired Evolutionary Algorithm Applied to Numerical Optimization. In: *Third World Congress on, Nature and Biologically Inspired Computing*. [online] Salamanca: IEEE, pp. 466-471. Available at: <https://ieeexplore.ieee.org/document/6089631> [Accessed 12 Oct. 2020].

³⁹ Parpinelli, R. S. and Lopes, H. S. (2012). Biological plausibility in optimisation: an ecosystemic view. *International Journal of Bio-Inspired Computation*, 4(6), pp. 345-358.

By biological ecosystems, this research implies to the field of ecology and the definitions that emerge thereof (as mentioned in 2.1.1). In the analogy of an HBS the ECO was designed to be behaving as a biological ecosystem, thereby exhibiting natural computing inspired by nature. It essentially symbolizes an algorithm populated by hypothetical species which behave according to an optimization algorithm and the ecosystem on the whole consists of populations of species that respond to each other and the environment corresponding to aforementioned ecological concepts of habitats, ecological relationships, and ecological successions.

Considering the fact that the inhabitants of the above mentioned algorithm, which are the species (as independent agents) and their populations (as collective data sets), in their pursuit of an optimization state driven by the search algorithms strive to attain a state of equilibrium, and the HBS rely on ecological concepts to attain this state of equilibrium, and behave as biotic elements (the species and their populations) in an abiotic environment, it can be theorized that the algorithm seeks symbiosis amongst its biotic elements. Here, the symbiosis the research refers to is the “*symbiosis that is not just defined by persistent mutualistic interactions*” (de Bary, 1879)⁴⁰, but the “*symbiosis that also involves commensal interactions and parasitic interactions, however, excluding predatorial interactions*” (Douglas, 2010)⁴¹. Because “*symbiosis is now increasingly being considered as an important selective force behind evolution*” (Wernegreen, 2004)⁴², as “*many biological species have been found to have had a long history of interdependent co-evolution*” (Paracer and Ahmadjian, 2000)⁴³, for an HBS, symbiosis becomes an essential means for the achievement of an equilibrium state.

⁴⁰ De Bary, A. (1879) *Die Erscheinung der Symbiose: Vortrag, gehalten auf der Versammlung deutscher Naturforscher und Aerzte zu Cassel (In English - The Phenomenon of Symbiosis)*, Strassburg: Karl J. Trübner, 30 pp.

⁴¹ Douglas, A. E. (2010) *The Symbiotic Habit*, Princeton, NJ: Princeton University Press, 232 pp.

⁴² Wernegreen, J. J. (2004). Endosymbiosis: Lessons in Conflict Resolution. *PLoS Biology*, 2(3), pp. 345-358.

⁴³ Paracer, S. and Ahmadjian, V. (2010) *Symbiosis: An Introduction to Biological Associations*, Princeton: Oxford University Press, 304 pp.

Parpinelli and Lopes further expand on the behavior and ecological significance of these symbiotic relationships within an ECO to be “*both positive and negative relationships between individuals of the same species (intraspecific relationships or homotypic) or between individuals of different species (interspecific relations or heterotypic)*” (Parpinelli and Lopes, 2014).⁴⁴ The four different combinations of these relationships can be described as following:

- Positive Intraspecific relationships (positive homotypic) – Interactions that lead to the constitution of societies and colonies.
- Negative Intraspecific relationships (negative homotypic) – Interactions such as Competition and Cannibalism.
- Positive Interspecific relationships (positive heterotypic) – Interactions such as Mutualism, Proto cooperation, Inquilinism, and Commensalism.
- Negative Interspecific relationships (negative heterotypic) – Interactions such as Competition, Amensalism, Predatism, Parasitism, and Slavery.

Because Intraspecific relationships are pertaining to the interactions between the individuals of the same species, they lead to population level behaviors (properties defining a population), whereas, Interspecific relationships that are between individuals of different species can lead to an ecological community (group of species across habitats sharing similar properties). These relationships also have a crucial role in functioning of the search algorithm, “*where **Intraspecific relationships** are responsible for **intensifying** the search and **interspecific relationships** are responsible for **diversifying** the search thereby maintaining a healthy diversity in the optimization algorithm thus increasing the biodiversity*” (Ibidem).⁴⁵.

⁴⁴ Parpinelli R. S. and Lopes, H. S. (2014). A computational ecosystem for optimization: review and perspectives for future research. *Memetic Computing*, 7(1), pp. 29-41.

⁴⁵ Ibidem

To establish a canonical ECO, Parpinelli and Lopes developed the following pseudo code for the algorithm (Parpinelli and Lopes, 2011).⁴⁶:

1. Let $i = 1, \dots, NQ, j = 1, \dots, NH$ and $t = 0$;
2. Initialize each population $Q_i(t)$ with n_i random candidate solutions;
3. **while** stop criteria no satisfied **do** {Ecological succession cycles}
4. Perform evolutive period for each population $Q_i(t)$;
5. Identify the region of reference C_i for each population $Q_i(t)$;
6. Using the C_i values, define the NH habitats;
7. For each habitat $H_j(t)$ define the communication topology $CT_j(t)$ between populations $Q_i^j(t)$;
8. For each topology $CT_j(t)$, perform interactions between populations $Q_i^j(t)$;
9. Define communication topology $TH(t)$ between $H_j(t)$ habitats;
10. Perform interactions between $H_j(t)$ habitats according to $TH(t)$;
11. Increase t ;
12. **end while**

⁴⁶ Parpinelli, R. S. and Lopes, H. S. (2011). An Eco-inspired Evolutionary Algorithm Applied to Numerical Optimization. In: *Third World Congress on, Nature and Biologically Inspired Computing*. [online] Salamanca: IEEE, pp. 466-471. Available at: <https://ieeexplore.ieee.org/document/6089631> [Accessed 12 Oct. 2020].

This algorithm is an HBS displaying all the afore mentioned ecological relationships.

The pseudo code for the afore mentioned canonical ECO algorithm can be simplified as followed:

- Apart from the time instant (t), there are three important variables and thus participants of this algorithm, namely, the individuals (i), the populations (Q) and the habitats (j).
- The individuals are the basic elements of the algorithm, the populations denote all the elements sharing same fundamental properties analogical to the biological equivalent of a species, whereas the habitat is a set of populations reaching similar optimization states at a specific frame of time.
- This implies, that a population of individuals in a habitat for a given instance of time cannot be a part of another habitat in the same instance of time.
- After establishing on *line 1* that the total number of possible populations would be NQ and total number of possible habitats would be NH , *line 2*, initiates the algorithm at $t=0$, with a random organization.
- From *lines 3 to 12*, the biological concept of ecological succession is represented by a while loop that proceeds until the ecological succession cycles reach the maximum predefined value.
- Within the loop of ecological succession, *line 4* exploits the optimization within each population pertaining to its intensification and diversification criteria. After the end of this period the habitats of the system are identified, as shown on *lines 5 and 6*. Soon after, intra-habitat communications are established and performed in the form of an ecological mating relationship as shown on *lines 7 and 8*. Then, *lines 9 and 10* establish and perform inter-habitat communication in the form of an ecological migration relationship.

- As the time instant increases in line 11, the lines 3 to 12 are looped to fulfill the maximum predefined value of the ecological succession.
- Identifying the habitats of the system, $H_j(t)$, represented by lines 5 and 6 are very crucial steps in this algorithm. Here, the longest concentrations of individual populations are recognized, their centroids are found and a habitat is defined using a predetermined minimum distance threshold ρ .
- Intra-habitat communication $CT_j(t)$, in every habitat $H_j(t)$, between populations $Q_i^j(t)$, are achieved by establishing and performing the aforementioned ecological mating relationship. In this relationship an individual of each population, hereafter termed as *the best individual* is chosen using “*the tournament strategy*” (Blickle, 2000)⁴⁷ and genetic exchange between them is performed in order to generate a new individual. The new generated individual replaces an individual selected at random in their initial population, obviously excluding the best individual.
- Inter-habitat communication TH (t), in every habitat $H_j(t)$, is achieved by establishing and performing the aforementioned ecological migration relationship. In this relationship, for each habitat, a random population within the habitat is chosen. The best individual of this population is chosen and is subjected to migration to another habitat. In the destination habitat, it replaces an individual that is randomly chosen, obviously excluding the best individual of the destination habitat.

Thus, a canonical ECO is an HBS implementing all the different ecological criteria while optimizing the algorithm serving as a working prototype of a Computational Ecosystem.

⁴⁷ Blickle, T. (2000). Tournament Selection. In: T. Bäck, D. B. Fogel, and Z. Michalewicz, ed., *Evolutionary Computation 1: Basic Algorithms and Operators*, 1st ed. New York: Taylor & Francis Group, pp. 181-186.

Further expanding on the semantics and usage of an HBS, Parpinelli and Lopes have bifurcated the HBS into the following groups (Parpinelli and Lopes, 2012).⁴⁸:

- **Bio Plausible HBS** – Here, designers of the computational ecosystem generally aim to “*achieve biologically plausible functionalities in non-biological contexts, such as the optimization of engineering problems.*”
- **Engineered HBS** – Here, designers of the computational ecosystem have only one purpose and that is “*to combine more than one bio-inspired algorithm together to create a new algorithm.*”

This thesis, however, would eventually intend to develop, test, taxonomize, and prototype **Hybrid Bio Plausible Bio-inspired Stochastic Optimization Algorithms** as functioning Computational Ecosystems primarily in computational environments. These computational ecosystems would consist of individual participants exhibiting structural properties of **biotic** and **abiotic** agents programmed with the behavioral properties of their real-life counterparts.

Depending on the search landscape and driven by the optimization goals communicated across different species by means of **stigmergy**, the biotic agents would form populations and several populations across the iterations over time would form habitats. While exhibiting properties of **homogenous and heterogenous cooperation**, by means of **intraspecific** and **interspecific** relationships, the populations would experience **intensification** (to find the state of equilibrium and thus optimization) and **diversification** (to maintain a significant amount of biodiversity).

⁴⁸ Parpinelli, R. S. and Lopes, H. S. (2012). Biological plausibility in optimisation: an ecosystemic view. *International Journal of Bio-Inspired Computation*, 4(6), pp. 345-358.

In order to develop the aforementioned Bio Plausible Hybrid Bio-inspired Stochastic Optimization Algorithms as functioning Computational Ecosystems, the research needs to identify existing bio-inspired algorithms that are dormant and already implemented in the field of computation. Figure 2.1 illustrates a representation of the “taxonomy and nomenclature of various bio inspired optimization algorithms grouped by the area of inspiration” (Binitha and Sathya, 2012)⁴⁹.

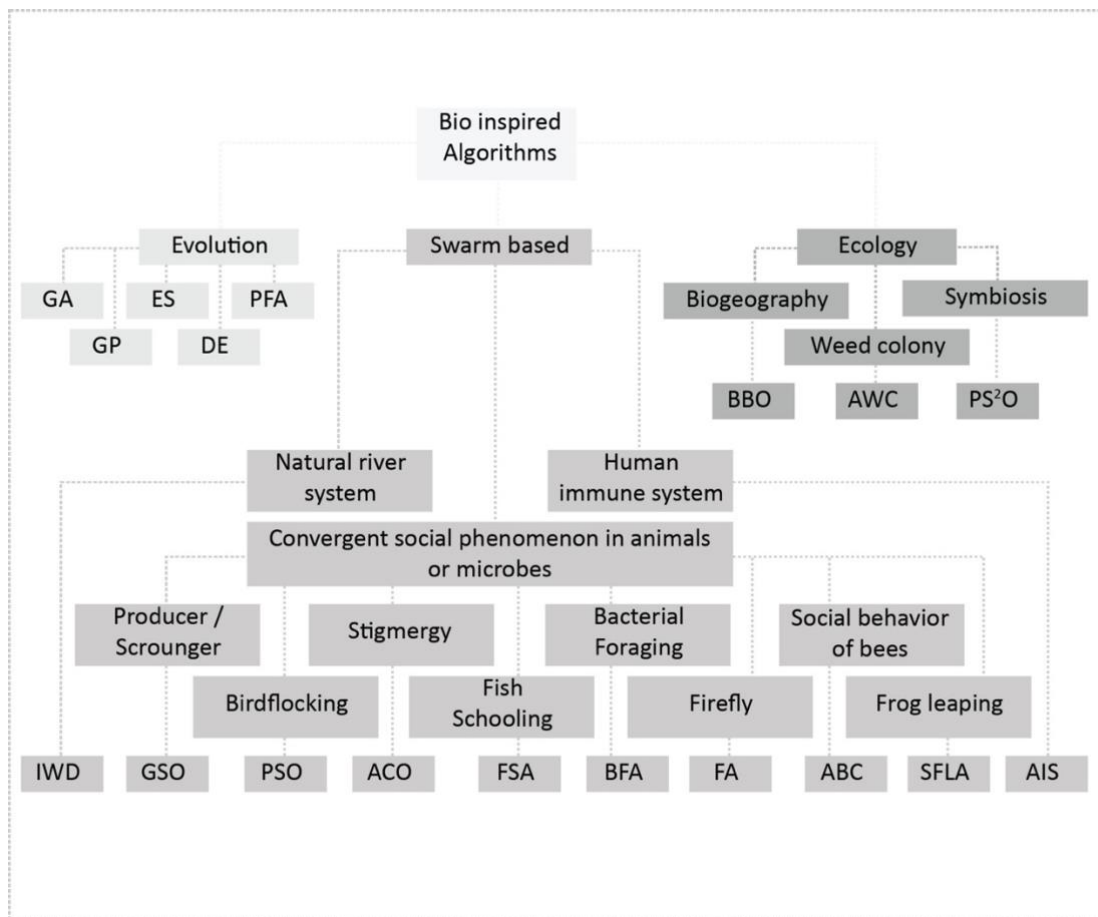


Figure 2.1 – Graphical representation of the taxonomy and nomenclature of various bio inspired optimization algorithms grouped by the area of inspiration. Original image by Binitha and Sathya (May 2012) Illustration and graphics by Author (January 2021).

⁴⁹ Binitha, S. and Sathya, S. S. (2012). A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSC)*, 2(2), pp. 137-151.

Abbreviations mentioned in fig. 2.1. –

- **GA – Genetic Algorithms** - Evolutionary based stochastic optimization algorithm with a global search potential.
- **GP – Genetic Programming** - An extension to Genetic algorithms differs from the latter in terms of representation of the solution.
- **ES – Evolutionary Strategy** - Global optimization algorithm inspired by the theory of adaptation and evolution by means of natural selection.
- **DE – Differential Evolution** - Similar to GA except in a DE mutation is the result of arithmetic combinations of individuals.
- **PFA – Paddy Field Algorithm** - Operates on reproductive principle based on proximity to global solution and population density akin plant populations.
- **IWD – Intelligent Water Drops Algorithm** - An innovative population-based method inspired by the processes in natural river systems.
- **GSO – Group Search Optimizer** - A population-based optimization algorithm, which adopts the producer–scrounger (PS) model metaphorically to design optimum searching strategies, inspired by animal foraging behavior.
- **PSO – Particle Swarm Optimization** - A computational intelligence oriented, stochastic, population-based global optimization technique inspired by the social behavior of bird flocking searching for food.
- **ACO – Ant Colony Optimization** - A meta heuristic inspired by the foraging behavior of ants in the wild based on stigmergy.

Theorizing, Taxonomizing and Prototyping operational Ecosystems in Computational environments

- **FSA – Fish Swarm Algorithm** - Swarm intelligent evolutionary computation technique inspired by the natural schooling behavior of fish.
- **BFA – Bacterial Foraging Algorithm** - Inherits the aspects of bacterial foraging patterns like chemo taxis, metabolism, reproduction and quorum sensing.
- **FA – Firefly Algorithm** - An unconventional swarm-based heuristic algorithm for constrained optimization tasks inspired by the flashing behavior of fireflies.
- **ABC – Artificial Bee Colony Algorithm** - A swarm intelligence algorithm based on swarming behavior of bees; further classified into - foraging and mating.
- **SFLA – Shuffled Frog Leaping Algorithm** - A population-based cooperative meta-heuristic algorithm with efficient mathematical function and global search capability.
- **AIS – Artificial Immune System Algorithm** - Based on clonal selection principle it is a population-based algorithm inspired by the human immune system which is a highly evolved, parallel and distributed adaptive system.
- **BBO – Biogeography Based Optimization** - A global optimization algorithm inspired by mathematical models of biogeography. Biogeography is the study of distribution of species in nature over time and space.
- **AWC – Invasive Weed Colony Optimization** - A numerical stochastic search algorithm inspired by the ecology of weed colonization and distribution.
- **PS²O - Particle Swarm Optimization with two hierarchies** – It extends the dynamics of the canonical PSO algorithm by adding a significant ingredient that takes into account the symbiotic co evolution between species.

2.1.4 About the Implementation of Computational Ecosystems

Fig. 2.1 helps the research in establishing two crucial observations in the concept of computational ecosystems:

- The concept of a recognized, well-researched taxonomy and nomenclature of existing bio-inspired algorithms that could serve as examples of computational ecosystems already exists.
- More such algorithms that take into account the implementation of either Bio Plausible or Engineered Hybrid Bio-inspired Stochastic Optimization Algorithms can be designated as computational ecosystems.

Having answered the question of what a computational ecosystem is and how an algorithm could be termed as a computational ecosystem by understanding a canonical ECO algorithm, its properties, and its relevance in the theoretical framework of this research, the semantics still have a few unanswered questions pertaining to the implementation of computational ecosystems to the relevance of this research (explained in 2.1.3), especially issues relating to the following:

- To establish a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm.
- To represent a fluid design workflow that performs modelling, analysis and fabrication simultaneously to generate form, structure, and enclosure for architectural intent.
- To make computational design more autonomous and digital fabrication more data-driven.

To elaborate on the above-mentioned concerns, the relationship of the built form and the algorithm excluding the role of design, that the research proposes to establish, needs to be analyzed and evaluated through the lens of biological and ecological interactions.

As elaborated in 1.1.4 *the built form and the algorithm*, and with the example of the “Urbach Tower project” (Wood et al, 2020)⁵⁰, the built form and design have seen an introduction of the algorithm in the AEC industry in the first couple of decades of the 21st century. The Architectural community which has been categorically bifurcated into the design architect and the delivery architect have also encountered a significant dominance of the algorithm in the form of adaptive design tools (such as Finch 3D) for the former and BIM software for the latter. However, with the advancements in the manufacturing, production and computation industries brought in by the fourth Industrial Revolution, the role of design can be projected to become redundant as the years go by. With the aforementioned proclamations (previously elaborated in 1.1.4) that :

- Algorithm in the form of computational design software can design the built form by performing modelling, analysis, and prototyping simultaneously with the use of data sets and rule sets pertaining to the structure, material, services, and equipment dynamically updated from a cloud source.
- Algorithm in the form of digital fabrication robots can fabricate the built form without the necessity of design documentation in the form of traditional plans, sections, elevations, and details, but with just a G-Code that involves a precise instruction set for tool paths and material usage.

⁵⁰ Wood, D., Grönquist, P., Bechert, S., Aldinger, L., Riggerbach, D., Lehmann, K., Rüggeberg, M., Burgert, I., Knippers, J., and Menges, A. (2020). From Machine Control to Material Programming Self-Shaping Wood Manufacturing of a High Performance Curved CLT Structure – Urbach Tower. In: *Fabricate 2020 Making Resilient Architecture*, London: UCL press pp. 50-57 Available at: <https://www.uclpress.co.uk/products/154646> [Accessed 15 Jun. 2020].

With the trivial use cases of design in the AEC industry maintained and supervised by the algorithm, both the design and the delivery architects can have more intellectual and intuitive control over the built form by controlling the algorithm in the form of computational design and digital fabrication respectively.

To perform the aforementioned trivial use cases of design by means of the algorithm in the form of computational design and digital fabrication, however, the algorithm would require a certain amount of autonomy in the system. This autonomy could be defined as the liberty to access important aspects such as access to data sets of quantitative characteristics, bye-laws and engineering properties of components related to structure, material, services, and equipment. Moreover, as the algorithm also would be performing analysis while modelling and prototyping, this autonomy could also be defined as the liberty to access data sets of a more macro level such as topographical data, climatological data, demographic data, data of energy sources, data of service sources, infrastructural data and thermal comfort data.

With the above-mentioned autonomy, the algorithm would be more intuitive, informed and conversant, perhaps not as much as the architect, although much better than the current design tools used by the architect. But with the power of computation, that involves data processing of higher quantities and at higher velocities, the algorithm could also be equipped to become autonomous with post construction data and post occupancy data. Here, the built form would be generating a massive amount of data sets such as structural performance, energy performance, Mechanical Electrical and Plumbing (MEP) efficiency, Heating Ventilation and Air-conditioning (HVAC) efficiency, sewage and water supply efficiency, functional efficiency, accessibility, climate control, comfort levels and the overall qualitative of space. If the algorithm gets an access to these data sets, there could be a tremendous enhancement in the way it deals with the subsequent built forms. Moreover, such an unprecedented access and autonomy over pre and post execution data sets would also allow the architect to truly and thoroughly master the qualitative and quantitative optimization of architecture.

However, it is not just the accessibility of data sets pertaining to the AEC industry that need to be established to form a relationship between the built form and algorithm that the research seeks (that's in fact the easy bit and quite elementary to be fair!). To truly serve as a better alternative to the traditional understanding and industry application of design (that is as a mere means of documentation and delivery of an AEC project), the algorithm needs to learn how to use these data sets, and more importantly how to communicate design decisions to and from the built form. To establish communication, a system would have to be installed that would create, maintain and evaluate a feedback loop between the **built form** and the **algorithm**. This means that, the feedback loop would have to be **autonomous** and **autopoietic**.

The research prefers to align itself with defining **autopoiesis** "*as a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components which* (Maturana and Varela, 1980)⁵¹:

- *through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them, and*
- *constitute it (the machine) as a concrete unity in space in which they (the components) exist by specifying the topological domain of its realization as such a network."*

"A unity, which is the core concept of the autopoietic machine, is essentially an entity distinct from a background and exists in a space defined by its components" (Maturana and Varela, 1980).⁵² Although, this means that an autopoietic system is in isolation, the canonical example of an autopoietic system, a biological cell, doesn't operate in isolation. Hence, the aforementioned feedback loop is not obligated to operate in isolation.

⁵¹ Maturana, H. R. and Varela, F. J. (1980). *Autopoiesis and Cognition: The Realization of the Living*. Berlin: Springer Science & Business Media, p.146.

⁵² Ibidem

In the process of generating a theoretical framework, it is keen to note that the research prefers autopoiesis over the generic allopoiesis as a core conceptual property of the feedback loop. Apart from the conceptual relevance of autopoiesis in establishing a recursive feedback loop within the communication logic between the built form and algorithm, the research also wants to abandon the traditional, functional rationale that design is an allopoietic, transformative process of creating something out of something else. By establishing the above theoretical construct of creating, maintaining and constantly evaluating an autonomous and autopoietic feedback loop between the **built form** and the **algorithm**, the research essentially seeks a constant recursive loop where the built form and the algorithm both undergo transformations using the data of the other while creating more data for further use.

To put it rationally, two structurally plastic composite unities (the built form and the algorithm) “interact with each other and thus operate as selectors of their individual paths of structural change, thereby forming a reciprocal structural coupling. As a result, the changes of state of one system trigger the changes of state of the other recursively, and a domain of coordinated conduct is established between the two mutually adapted systems” (Maturana, 2002).⁵³ An autonomous and autopoietic structural coupling as a means of communication between the built form and the algorithm both in the form of computational design and digital fabrication is exactly what this research has hypothesized in the last chapter (previously, in 1.2) as,

What if the **built form** was constructed, monitored and governed by an
autonomous, unbiased **algorithm**?

What if this **algorithm** was dynamically constructed, monitored and governed by
the **built form**?

⁵³ Maturana, H. R. (2002). Autopoiesis, Structural Coupling and Cognition: A history of these and other notions in the biology of cognition. *Cybernetics & Human Knowing*, 9(3-4), pp. 5-34.

A computational ecosystem as explained and elaborated (previously, in 2.1.3) clearly instills the fundamental concepts of a recursive, evolutionary algorithm in its workflow by defining itself as either a Bio Plausible or an Engineered Hybrid Bio-inspired Stochastic Optimization Algorithm.

A computational ecosystem by the virtue of the canonical ECO algorithm (previously elaborated in 2.1.3 About computational ecosystems) quite distinctly exhibits the following characteristics that are unique within the spectrum of bio-inspired optimization algorithms as seen in fig. 2.1:

- A computational ecosystem represents an autonomous, autopoietic feedback loop that is based upon the structural coupling of two entities (in a canonical ECO algorithm, these two structurally plastic composite unities can be clearly represented by the populations and habitats) which would be essential in establishing a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm.
- Moreover, the homogenous and heterogenous cooperation that is clearly represented by intraspecific and interspecific relationships, (driven by intensification and diversification within the evolutive period of a canonical ECO algorithm) in a computational ecosystem has all the key ingredients of representing a fluid design workflow that performs modelling, analysis and fabrication simultaneously to generate form, structure, and enclosure for architectural intent.
- And finally, the essential components of intra-habitat communication strategies and inter-habitat communication strategies (executed by applying the ecological mating and migration relationships respectively in a canonical ECO algorithm) would be instrumental in making computational design more autonomous and digital fabrication more data-driven.

2.2 Applicability of the semantics

As a corollary to the previous section (2.1 Establishing the semantics), which focused on addressing, understanding, and establishing key terminologies that form the theoretical framework in the pursuit of this research, this section (2.2 applicability of the semantics) explores the implementation of the theoretical construct of computational ecosystems into the existing fields of pure and applied sciences. The study and exploration of these implementations will be crucial for the research as it will further assist in instituting the definitions and use cases of most of the semantics that has been established until this point. As this entire chapter focuses on the first literary objective, that installs the **lexical semantics** (in the reference of setting up a vocabulary base), this section is vital in providing empirical evidence for the structure of thought that serves as a theoretical foundation for the rest of the research.

What is interesting to observe in the applications of the semantics is that different fields of science define a computational ecosystem differently, and thus it becomes easier for the research to form its definition of a computational ecosystem (in the context of forming a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm) while being relevant and comparable to the other applications, apart from the canonical computational ecosystem (explained and elaborated upon in section 2.1). However not acknowledged scientifically, many of these applications of computational ecosystems can be observed in the field of “*Artificial Life or A-Life*” (Langton, 1995)⁵⁴ where agent-based simulations and interactions are conducted in what is termed as “*Artificial Ecosystems*”.

Continuing on the above concepts of A-life, this thesis would illustrate some examples of the applicability of computational ecosystems in behavioral sciences and visual arts.

⁵⁴ Langton, C. G. (1995). *Artificial Life: An Overview*. Cambridge: MIT Press, p.341.

2.2.1 Applicability in biology, epidemiology and behavioral sciences

Most of the bio inspired optimization algorithms illustrated in fig. 2.1 in all the three distinctions (that is, evolution based, swarm based, and ecology based) could be quite essentially classified as computational ecosystems. Most of them, and especially the bio inspired optimization systems specifically associated to the swarm-based algorithms (for example, the PSO, ACO, BFA, FSA, and ABC) implement one or all of the crucial ingredients of a computational ecosystem such as:

- **Structural Coupling** – Which is quite efficiently demonstrated by the *GSO* algorithm (He, Wu and Saunders, 2006)⁵⁵, where the producers, scroungers and rangers enter into a 3-way structural coupling that allows the algorithm to solve many of its optimization goals.
- **Intensification and Diversification** – Which is a core concept in the functioning of a *BBO* (Simon, 2009)⁵⁶ algorithm (apart from the *PS²O* algorithm as elaborated in 2.1.2), where the species (participating individuals or agents) undergo homogenous and heterogenous co-operations to achieve their optimization objectives.
- **Intra-habitat and inter-habitat communication strategies** – Which is embedded in the function of chemotaxis or cell movement in a *BFA* (Passino, 2002)⁵⁷ where the bacterial movement is simulated by establishing a cell-to-cell communication mechanism (similar to the use of stigmergy in an *ACO* algorithm as explained in 2.1.2) to achieve the optimization goals.

⁵⁵ He, S., Wu, Q. H. and Saunders, J. R. (2006). A Novel Group Search Optimizer Inspired by Animal Behavioral Ecology. In: *IEEE International Conference on Evolutionary Computation*. Vancouver: IEEE, pp. 1272-1278.

⁵⁶ Simon, D. (2009). Bio-geography based optimization. *IEEE Transactions on Evolutionary Computation*. 12(6), pp. 702-713.

⁵⁷ Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*. 22(3), pp. 52-67

Although the aforementioned algorithms are examples of bio-inspired optimization algorithms and are theoretically situated in the first area of investigation in the field of Natural Computing (computing inspired by nature; previously explained and elaborated upon in 2.1.3), applications in the field of behavioral sciences are mostly based on the second area of investigation (simulation and emulation of nature by implementing computing). Quite often these applications in behavioral sciences which are theoretically termed as **artificial ecosystems** are performed by implementing Agent-based Modelling (ABM) or Individual-based Modelling (IBM). Although agent-based modelling is a mathematical concept that was introduced in the early 1940s, due to the lack of computational architecture it failed to pick up across the discipline of mathematics and computer sciences until the 1990s, after which it became quite ubiquitous as a technique of simulating natural forms and systems. On the whole, an ABM or IBM is a simulation system that helps in the emulation of natural forms and systems “*based on the actions and interactions of autonomous agents (both individual or collective entities) to understand the behavior of a system and what governs its outcomes*” (Railsback and Grimm, 2011)⁵⁸.

As ABMs and IBMs have been highly implemented **in the field of Biology**, their results have led studies into understanding and predicting some complex phenomena across the scientific disciplines, such as the complex phenomena of **emergence** (in the sense of macro-state changes emerging from micro-state agent behaviors) by mere simulation of some existing biological interactions in computational environments. The study and simulation of “*Population Dynamics*” (Caplat, Anand, and Bauch, 2007)⁵⁹ and “*Landscape Diversity*” (Wirth, Szabó, and Czinkóczy, 2016)⁶⁰ are some of the applications of ABMs and IBMs in biology.

⁵⁸ Railsback, S. F., & Grimm, V. (2011). *Agent-based and individual-based modelling: A practical introduction*. Princeton: Princeton University Press., p.329.

⁵⁹ Caplat, P., Anand, M., Bauch, C. (2007). Symmetric competition causes population oscillations in an individual-based model of forest dynamics. *Ecological Modelling* 211. 3(4), pp. 491-500.

⁶⁰ Wirth, E., Szabó, G., Czinkóczy, A. (2016). Measure of Landscape Heterogeneity by Agent-Based Methodology. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. III(8), pp. 145-151.

Computational Ecosystems in the form of ABMs and IBMs are also widely implemented ***in the field of Epidemiology*** almost as often and effectively as the conventional Compartmental Modelling technique (where agents/participants would move through computational compartments of S, I, and R; where S - Susceptible, I – Infectious and R - Recovered). ABMs and IBMs have often been employed in the analysis, information, and intervention of dangerous diseases such as the “*Avian Influenza which is the result of a mutant influenza virus infecting poultry chickens in Indonesia*” (Situngkir, 2004)⁶¹ and the more recent SARS-CoV-2 pandemic, commonly known as the COVID-19 pandemic where an “*ABM termed as CovidSim was developed by Neil Ferguson and his team of epidemiologists in 2020*” (David, 2020)⁶².

Computational Ecosystems in the form of ABMs have also been implemented ***in the field of behavioral sciences*** for varied purposes such as “*to understand the resource supply and demand relationships with the inclusion of human behavior in the nexus of the food-water-energy system in agriculture*” (Magliocca, 2020)⁶³. Whereas computational ecosystems in the form of A-Life (as explained in 2.2) have also been implemented in “*assessing population diversity, population density fluctuations, and socializing behaviors while developing human crowd simulators*” (Antunes, 2016)⁶⁴

All above implementations of Computational Ecosystems show structural couplings with intensification and diversification strategies performed with intra-habitat and inter-habitat communication strategies.

⁶¹ Situngkir, H. (2004). Epidemiology Through Cellular Automata: Case of Study Avian Influenza in Indonesia. [online]. Available at: <https://arxiv.org/abs/nlin/0403035> [Accessed 24 Apr. 2021].

⁶² David, A. (2020). Special report: The simulations driving the world’s response to COVID-19. *Nature*, [online]. Available at: <https://www.nature.com/articles/d41586-020-01003-6> [Accessed 12 Jan. 2021].

⁶³ Magliocca, N. R. (2020). Agent-Based Modelling for Integrating Human Behavior into the Food–Energy–Water Nexus. *Land* 2020[online] Volume 9(519), p. 25. Available at: <https://www.mdpi.com/2073-445X/9/12/519/> [Accessed 24 Apr. 2021].

⁶⁴ Antunes, R. F. (2016). Human Crowd Simulation: What can We Learn from ALife? In: *ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems*. [online] Cancun: MIT Press Direct, p. 8. Available at: <https://direct.mit.edu/isal/proceedings/alif2016/38/99500> [Accessed 26 Apr. 2021].

2.2.2 Applicability in visual arts and design

Computational Ecosystems in the form of Alife, ABM, and IBM have also experienced a considerable rise in the fields of visual arts, communication design, generative art, design engineering, and architectural design. While being driven by non-biological (that is not focused on survival, reproduction, or growth of a system) and/or non-optimization goals (that is not focused on the search algorithms for finding peak fitness conditions) these Computational Ecosystems have diverse subjective, and context-specific goals. Nonetheless, these algorithms can be deemed as Computational Ecosystems, as they exhibit structural couplings with intensification and diversification strategies performed with intra-habitat and inter-habitat communication strategies.

Because the focus of the algorithms in the field of Visual arts and communication design is on non-biological and non-optimization goals, a slightly non-computational and analogical definition for Computational Ecosystem has been derived as followed:

- Computational Ecosystems can be defined as *“computer programs that simulate interactions of agents inspired by life in nature. In a typical computational ecosystem, agents are organized in a hierarchical structure (food chain) and a community dynamic is promoted through the trade of token units of energy and biomass between these agents”* (Antunes, Leymarie, and Latham, 2016)⁶⁵.
- In ecology, *“Computational Ecosystems are used when modelling carbon-based contexts and can be considered part of the sub-domain of ABMs and IBMs”* (Ibidem)⁶⁶.

⁶⁵ Antunes, R. F., Leymarie, F. F., Latham, W. (2016). Computational Ecosystems in Evolutionary Art, and Their Potential for the Future of Virtual Worlds. In: Y. Sivan, ed., *Handbook on 3D3C Platforms*, 1st ed. Cham, Switzerland: Springer International Publishing, pp. 441-473.

⁶⁶ Ibidem

Although based strongly in Alife and ABM, the aforementioned research of Antunes, Leymarie, and Latham provides a profound insight into establishing the theoretical framework of Computational Ecosystems, for the application of generative arts while developing a thorough taxonomy of existing researches and artworks that implement computational ecosystems within the theoretical construct of the definition stated previously (not to be mistaken with the canonical computational ecosystem definition elucidated in 2.1.4). However, their theoretical assumptions possess some similarities to the canonical computational ecosystems, such as:

- “*Eden, a sonic system*” (McCormack, 2001)⁶⁷ which demonstrates that these computational ecosystems are **Autonomous** and to a certain extent **Autopoietic** as they exhibit certain properties of self-motivated communities of agents despite their significant biodiversity.
- “*A Genotype-Phenotype model is theorized to be existing in these computational ecosystems*” (Antunes, Leymarie, and Latham, 2016)⁶⁸ that further demonstrates the existence of **inter-habitat** and **intra-habitat** communication strategies often performed by homogenous, heterogenous, or chemo static means.

Their theoretical assumptions also add an important dimension of **Interactivity** to the realization of computational ecosystems. With this extension, the agents participating in a Computational Ecosystem could be added, deleted or their properties could be modified by external user input or abiotic factor, thereby making a computational ecosystem context-aware (Sommerer and Mignonneau, 1994).⁶⁹

⁶⁷ McCormack, J. (2001). Eden: an evolutionary sonic ecosystem. In: *Advances in Artificial Life, 6th European Conference*. Berlin: Springer - Verlag, p. 10.

⁶⁸ Antunes, R. F., Leymarie, F. F., Latham, W. (2016). Computational Ecosystems in Evolutionary Art, and Their Potential for the Future of Virtual Worlds. In: Y. Sivan, ed., *Handbook on 3D3C Platforms*, 1st ed. Cham, Switzerland: Springer International Publishing, pp. 441-473.

⁶⁹ Sommerer, C., Mignonneau, L. (1994). A-Volve: A real-time interactive environment. In: *ACM*

Apart from Generative Art, a diverse range of design disciplines have employed computational ecosystems in various forms. Here, the end goal is not as abstract as the implementation in visual or sonic arts but is often driven by the optimization of the more quantitative aspects of design. For instance, the “*application of computational ecosystems in the field of transport design as an ABM in the simulation and analysis of self-driving cars with respect to computationally generated traffic conditions, topographical conditions, and resource optimization applied to the scenario of the city of Lisbon, Portugal*” (Martinez and Viegas, 2017).⁷⁰ The model optimizes mobility outputs and CO2 emissions for two distinct transport systems. Computational Ecosystems in the form of ABM have also been implemented in the architectural and urban environments “*to assess greenhouse gas emissions employing pedestrian simulation in procedurally-generated 3D models*” (Aschwanden et al, 2009)⁷¹. Here, the model develops a stochastic search algorithm to simulate decisions that record the paths taken, while recording stress, effort and deviations.

Thus, Computational Ecosystems can be expressed with the canonical definition (2.1.4) or the non-canonical definition (2.2.2), however, the theoretical framework persists that a computational ecosystem represents a feedback loop (structural coupling) that is autonomous, autopoietic. Moreover, it undergoes homogenous and heterogenous co-operations performed by intra-habitat and inter-habitat communication strategies while allowing a degree of interactivity and become context-aware.

Siggraph Visual Proceedings. pp. 172–173.

⁷⁰ Martinez, L. M., Viegas, J. M. (2017). Assessing the impacts of deploying a shared self-driving urban mobility system: An agent-based model applied to the city of Lisbon, Portugal. *International Journal of Transportation Science and Technology*. [online] 6(2017), pp. 13-27. Available at: <https://reader.elsevier.com/reader/sd/pii/S2046043016300442?token=0D63F683EEF2FFDA49A010FD6D7BD77A18B8CAC02BDE12BCA11F14B0F0702DA369F02E3B46DAB8B7E279C516B24D40CC&originRegion=eu-west-1&originCreation=20210703000320> [Accessed 22 May 2020].

⁷¹ Aschwanden, G.D.P.A., Wullschleger, T., Müller, H., Schmitt, G. (2009). Agent based evaluation of dynamic city models: A combination of human decision processes and an emission model for transportation based on acceleration and instantaneous speed. *Automation in Construction*. [online] 22, pp. 81-89. Available at: <https://www.sciencedirect.com/science/article/pii/S0926580511001415> [Accessed 22 May 2020].

2.3 Cellular Automata as Computational Ecosystem

As defined, illustrated, and extensively exemplified in the two preceding sections of this chapter (2.1 and 2.2), Computational Ecosystems in the form of Alife, ABM, and IBM serve as widely accepted computational models across various fields, disciplines and industries not just to solve optimization problems, or to mimic natural systems, but also to generate art and to solve some critical global problems relating to climate change and pandemics. From a technical perspective, these Computational Ecosystems are realized by using toolkits that are based on applying programming languages such as JAVA, .NET, C++, J#, C#, and Logo. Some of these toolkits such as AnyLogic⁷², Repast⁷³, and NetLogo⁷⁴ also use their languages or modified versions of the aforementioned programming languages. All these toolkits and languages have a very strong (usually) open-source community that versions these toolkits often and well. However, they all have a major shortcoming with their very limited and sometimes even non-existent 3D modelling capabilities.

As the research hypothesizes to theorize, generate, taxonomize, and prototype computational ecosystems as autonomous, autopoietic, context aware feedback loops between the ***built form*** and ***algorithm***, it becomes very essential for the research to develop these computational ecosystems in an environment that anticipates modelling, analysis, and prototyping of 3-dimensional morphologies in 3-dimensional environments. Furthermore, additional software plugin capabilities required to create and use existing data sets (as explained in 2.1.4), would require 3-dimensional capabilities in the software and the technique employed to realize the Computational Ecosystem. Also, since the research has sensationally postulated to replace design, a 2-dimensional interface would be rudimentary and useless.

⁷² AnyLogic. (2000). France: The AnyLogic Company.

⁷³ Repast. (2006). Chicago: Repast HPC.

⁷⁴ NetLogo. (1999). Illinois: Northwestern University Center for Connected Learning and Computer-Based Modelling

On the whole, owing to the capabilities of the existing programming languages and available software, the Computational Ecosystem thus developed would have to conform to the theoretical definitions elaborated thus far and performs in a 3-dimensional Euclidian environment that exhibits the aforementioned capabilities.

Although never theorized to be an example of a Computational Ecosystem by ecologists or computational theorists, **Cellular Automata** are “*computational models that expand on the idea of self-replicating systems*”, as was developed conceptually in the mid 20th century by John Von Neumann (Von Neumann and Burks, 1966)⁷⁵. A Cellular Automaton consists of participating agents which are created, replicated and deleted based on pre-conceived rules in a computational environment. The rules are often rooted in the study of behaviors and interactions of biological organisms or systems. The participating agents are denoted by a grid-cell in the Euclidian environment, hence the name - **Cellular Automata**. They have been applied in a diverse range of disciplines across pure and applied sciences since their original inception by John Von Neumann in the 1950s, however, “*the Von Neumann model of cellular automata did not gain popularity until John Conway, came up with the cellular automaton - Game of Life*” in 1970 (Gardner, 1970)⁷⁶. In the early 2000s however, Stephen Wolfram, deeply theorized Cellular Automata into different typologies as an independent field of computational sciences in his book – A New Kind of Science.

To base the theoretical framework of establishing the lexical semantics, the thesis would now be elaborating on the different models of Cellular Automata – by Von Neumann, Conway, and Wolfram to understand their distinct properties and theoretical implications to the idea of Computational Ecosystems within the relevance of the research.

⁷⁵ Von Neumann, J. and Burks, A. W. (1966). *Theory of Self-Reproducing Automata*. Illinois: University of Illinois Press, p.387.

⁷⁶ Gardner, M. (1970). MATHEMATICAL GAMES - The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. [online] 223(4), pp. 120-123. Available at: <https://web.stanford.edu/class/sts145/Library/life.pdf> [Accessed 31 May 2020].

2.3.1 Cellular Automata – John Von Neumann model

John Von Neumann with his colleague Stanislaw Ulam proposed a “*simple self-reproducing kinematic automaton which would serve as a proof-of-concept for his hypothetical claims of a non-biological self-replicating system during his university lectures in 1948 and 1949*” (Von Neumann and Burks, 1966)⁷⁷. This time in the history of mathematics and computation was filled with speculations by mathematicians and computer scientists around the world of an inevitable logical barrier that could not accept a self-replicating system in any shape or form. Most of these theories predated the computational architecture as we know it today (it also belongs to the time when the conceptualization and construction of a Universal Turing Machine by Alan Turing was not a reality). For this reason, the Von Neumann model of cellular automata, which could be considered as the canonical cellular automata was originally conceptualized and constructed by Von Neumann with the use of pencil and graph paper.

“Von Neumann evidently was dissatisfied with his original kinematic model because of its seeming mathematical inelegance. The kinematic model, while qualitatively sound, appeared not easily susceptible to mathematically rigorous treatment and so might not serve to convince a determined skeptic” (Freitas Jr. and Merkel, 2004)⁷⁸. Ulam, then suggested Neumann that the *“notion of a self-replicating machine would be amenable to rigorous treatment if it could be described in a “cell space” format that is, in the form of a geometrical grid or tessellation, regular in all dimensions”* (Ibidem)⁷⁹. This important intervention helped Neumann develop the automaton in its simplest form.

⁷⁷ Von Neumann, J. and Burks, A. W. (1966). *Theory of Self-Reproducing Automata*. Illinois: University of Illinois Press, p.387.

⁷⁸ Freitas Jr., R. A., Merkle, R. C. (2004). *Kinematic Self-Replicating Machines*. [online] www.molecularassembler.com. Available at: <http://www.molecularassembler.com/KSRM/2.1.3.htm> [Accessed 18 Nov. 2017].

⁷⁹ Ibidem

Thus, the basic rules for the configurations, neighbourhoods and states were laid down for the first operating cellular automaton. In this canonical version, the cellular automaton is modelled as a **finite state automaton** (FSA – as a computational model where the model can exist in only one state at a given time. These states of existence are predetermined and finite). These FSA, after the intervention of Ulam are termed as **Cells** laid out in an infinite square grid. To describe the canonical cellular automata rule set, however, some semantics would have to be established, as following:

- **Cell** – The unit square of a square grid. The aforementioned FSA exists in each cell and all these cells make their transitions synchronously, in correspondence with a universal "clock" as in a synchronous digital circuit depending on the configuration of their neighbourhoods (in this case, a Von Neumann neighbourhood).
- **States** – Any states of existence as defined in the initial rule-set. The state of a cell is determined by the states of its neighbourhood. In the canonical cellular automaton, a total of 29 possible different states are predetermined.
- **Von Neumann neighbourhood** – The group of 4 neighbouring cells surrounding a central cell on its cardinal directions (i.e. up, down, left and right) in a square grid. The neighbourhood consists of the central cell itself, and it's four neighbours that are at a unit distance from the central cell.

As previously mentioned, the Von Neumann model of cellular automata was initially conceived without a computer. However, because of the current state of advanced computation, many reiterations of the canonical version have been created. With it, lots of computational tools have been developed for the ease of modelling of these basic and yet powerful cellular automata. One such platform is the open source software Golly⁸⁰.

⁸⁰ Golly. (2005). England: Andrew Trevorrow and Tom Rokicki.

The basic distinctions of these states can be described as followed:

- **The unexcited state** – This could be considered as the ‘0’ state or the false state of the cell.
- **The sensitized state** – This could be considered as the transition state from the ‘0’ state to the ‘1’ state.
- **The excited state** - This could be considered as the ‘1’ state or the true state of the cell. It carries one bit (as in binary) of data for every sensitized step.
- **The quiescent state** – This could be considered as the transition state from the ‘1’ state to the ‘0’ state.
- **The transmission state** – This could be considered as states that help in transmitting information in the entire neighbourhood. Each of these could be in the excited or quiescent states.
- **The confluent state** – This is activated if a signal is received from the entire neighbourhood. If activation occurs, after two moments of time they emit signals outward toward any cell in their neighbourhood which does not have a transmission directed toward it. The confluent states have the property of a one-cycle delay; thus, they carry two bits of data at any instance of time.

Thus, considering the above basic distinctions of possible states, a “*Von Neumann model of Cellular Automata can have 29 different predetermined states*” (Freitas Jr. and Merkel, 2004)⁸¹ depending on their interspecific combinations which can be considered as the rule sets to initiate the Automaton.

⁸¹ Freitas Jr., R. A., Merkle, R. C. (2004). *Kinematic Self-Replicating Machines*. [online] www.molecularassembler.com. Available at: <http://www.molecularassembler.com/KSRM/2.1.3.htm> [Accessed 18 Nov. 2017].

They can be described in Golly by means of RGB values (red, green, blue) as followed:

1. *U (Ground state or Unexcited state) (48,48,48)*
2. *the transition or sensitized states (in 8 sub states)*
 - a. *S (newly sensitized) (255, 0, 0)*
 - b. *S0 – (sensitized, no input for 1 cycle) (255, 125, 0)*
 - c. *S00 – (sensitized, no input for 2 cycles) (255, 175, 50)*
 - d. *S000 – (sensitized, no input for 3 cycles) (251, 255, 0)*
 - e. *S01 – (sensitized, no input for 1 cycle, then input for 1 cycle) (255, 200, 75)*
 - f. *S1 – (sensitized, input for one cycle) (255, 150, 25)*
 - g. *S10 – (sensitized, input for 1 cycle, then no input for 1 cycle) (255, 255, 100)*
 - h. *S11 – (sensitized, input for 2 cycles) (255, 250, 125)*
3. *the confluent states (in 4 states of excitation)*
 - a. *C00 – (quiescent and will also be quiescent next cycle) (0, 255, 128)*
 - b. *C01 – (now quiescent, but will be excited next cycle) (33, 215, 215)*
 - c. *C10 – (excited but will be quiescent next cycle) (255, 255, 128)*
 - d. *C11 – (currently excited and will be excited next cycle) (255, 128, 64)*

Theorizing, Taxonomizing and Prototyping operational Ecosystems in Computational environments

4. *The ordinary transmission states (in 4 directions, excited or quiescent)*
 - a. *North-directed (excited and quiescent) (36, 200, 36) (106, 106, 255)*
 - b. *South-directed (excited and quiescent) (106, 255, 106) (139, 139, 255)*
 - c. *West-directed (excited and quiescent) (73, 255, 73) (122, 122, 255)*
 - d. *East-directed (excited and quiescent) (27, 176, 27) (89, 89, 255)*

5. *the special transmission states (in 4 directions, excited or quiescent)*
 - a. *North-directed (excited and quiescent) (191, 73, 255) (255, 56, 56)*
 - b. *South-directed (excited and quiescent) (203, 106, 255) (255, 89, 89)*
 - c. *West-directed (excited and quiescent) (197, 89, 255) (255, 73, 73)*
 - d. *East-directed (excited and quiescent) (185, 56, 255) (235, 36, 36)*

The confluent states consider the following (Von Neumann and Burks, 1966)⁸²:

- *They do not communicate data, and they take input from one or more ordinary transmission states, or deliver output to transmission states, that are not directed towards them.*

- *They implement **AND** operator to inputs.*

- *Data held by them is lost if that state has no adjacent transmission state that is also not pointed at them. Thus, they can be used as "bridges".*

⁸² Von Neumann, J. and Burks, A. W. (1966). *Theory of Self-Reproducing Automata*. Illinois: University of Illinois Press, p.387.

Furthermore, the model is based on the following **construction rules** (Von Neumann and Burks, 1966)⁸³:

- *Initially, the cells are in the ground state **U**. When given an input excitation from a neighbouring ordinary or special transmission state, the cell in the **U** state becomes **sensitized**, transitioning through a series of states before finally **resting** at a quiescent transmission or confluent state. The choice of which destination state the cell will reach is determined by the sequence of input signals. Thus, the transition/sensitized states can be thought of as the nodes of a bifurcation tree leading from the **U** state to each of the quiescent transmission and confluent states. In the following tree, the sequence of inputs is shown as a binary string after each step:*
- *a cell in the **U**, given an input, will transition to the **S** state in the next cycle (1)*
 - *a cell in the **S**, given no input, will transition into the **S0** state (10)*
 - *a cell in the **S0**, given no input, will transition into the **S00** (100)*
 - *a cell in the **S00**, given no input, will transition into the **S000** (1000)*
 - *a cell in the **S000**, given no input, will transition into the east-directed ordinary transmission state (10000)*
 - *a cell in the **S000**, given an input, will transition into the north-directed ordinary transmission state (10001)*

⁸³ Von Neumann, J. and Burks, A. W. (1966). *Theory of Self-Reproducing Automata*. Illinois: University of Illinois Press, p.387.

Theorizing, Taxonomizing and Prototyping operational Ecosystems in Computational environments

- *a cell in the **S00**, given an input, will transition into the west-directed ordinary transmission state (1001)*
- *a cell in the **S0**, given an input, will transition into the **S01** (101)*
 - *a cell in the **S01**, given no input, will transition into the south-directed ordinary transmission state (1010)*
 - *a cell in the **S01**, given an input, will transition into the east-directed special transmission state (1011)*
- *a cell in the **S** state, given an input, will transition into the **S1** (11)*
 - *a cell in the **S1**, given no input, will transition into the **S10** (110)*
 - *a cell in the **S10**, given no input, will transition into the north-directed special transmission state (1100)*
 - *a cell in the **S10**, given an input, will transition into the west-directed special transmission state (1101)*
 - *a cell in the **S1**, given an input, will transition into the **S11** (111)*
 - *a cell in the **S11**, given no input, will transition into the south-directed special transmission state (1110)*
 - *a cell in the **S11**, given an input, will transition into the quiescent confluent state C00 (1111)*

And, on the following ***Destruction rules*** (Von Neumann and Burks, 1966)⁸⁴:

- *If an input is sent from a special-transmission state cell into a confluent-state cell, the latter will be reduced back to U.*
- *Similarly, if an input is sent from a special-transmission state cell into an ordinary transmission-state cell, the latter will be reduced back to U.*
- *Conversely, if an input is sent from an ordinary-transmission state cell into a special transmission-state cell, the latter will be reduced back to U.*

Thus, Von Neumann demonstrated that his “*cellular model of machine replication possessed the sufficient logical properties including logical universality, construction capability, and constructional universality , thus enabling self-replication and the empirical evidence of a Universal Constructor*” (Freitas Jr. and Merkle, 2004)⁸⁵. Interestingly, the model was implemented in the calculation of liquid motion, where “*the driving concept of the method was to consider a liquid as a group of discrete units and calculate the motion of each based on its neighbours' behaviors*” (Białynicki-Birula and Białynicka-Birula, 2004).⁸⁶

However, the important aspect of the Von Neumann model that could be implemented in the context of Computational Ecosystems is the intraspecific relationship in a cell depending on its state, and interspecific relationship with its neighbourhood in a given instance of time. Moreover, the structural coupling between the cells, states and the neighbourhood is analogous to that of a GSO algorithm (2.1.3).

⁸⁴ Von Neumann, J. and Burks, A. W. (1966). *Theory of Self-Reproducing Automata*. Illinois: University of Illinois Press, p.387.

⁸⁵ Freitas Jr., R. A., Merkle, R. C. (2004). *Kinematic Self-Replicating Machines*. [online] www.molecularassembler.com. Available at: <http://www.molecularassembler.com/KSRM/2.1.3.htm> [Accessed 18 Nov. 2017].

⁸⁶ Białynicki-Birula, I., Białynicka-Birula, I. (2004). *Modelling Reality - How Computers Mirror Life*. Oxford: Oxford University Press, p.188.

2.3.2 Cellular Automata – John Conway model

Although the Von Neumann model was quite accomplished at performing its function of a self-replicating machine in the form of a Universal Constructor, it was quite laborious and complex in the form of its diverse range of states and their combinations. After Von Neumann, very many mathematicians have established their versions of Cellular Automata. Most of them attempting to reduce the number of finite states determined before setting up the Automaton. Notable examples during this time include the “Codd model with eight states which was created to emulate the Von Neumann model with a lesser amount of states” (Codd, 1968)⁸⁷; or the “Greenberg-Hastings cellular automaton (GH model) with three states which was designed to model excitable media” (Greenberg and Hastings, 1978)⁸⁸.

As the computational resources evolved, many such Automata were developed by mathematicians and computer scientists through the decades of the late 20th century further demonstrating that simpler machines than von Neumann's can be shown to be capable of reproducing themselves. The question then arose: “How simple can a machine become while still retaining the capacity to reproduce itself?” (Langton, 1984)⁸⁹. In 1970, John Conway’s game of life, which was “a cellular automaton with just 2 states was designed to serve as a solitary math game expanding on the idea of mathematical simulation games” (Gardner, 1970)⁹⁰ answered the above question to its very extreme. The simplification in Conway’s model was so remarkable, that it breathed a new life into the study and research of cellular automata in the fields of mathematics and computer sciences.

⁸⁷ Codd, E. F. (1968). Cellular Automata. PhD Thesis. Academic Press, New York.

⁸⁸ Greenberg, J. M., Hastings, S. P. (1978). Spatial Patterns for Discrete Models of Diffusion in Excitable Media. *SIAM Journal on Applied Mathematics*, 34(3), pp. 515–523.

⁸⁹ Langton, C. G. (1984). Self-Reproduction in Cellular Automata. *In: Physica 10D*. Amsterdam: Elsevier Science Publishers B.V. (North-Holland Physics Publishing Division), pp. 135-144.

⁹⁰ Gardner, M. (1970). MATHEMATICAL GAMES - The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. [online] 223(4), pp. 120-123. Available at: <https://web.stanford.edu/class/sts145/Library/life.pdf> [Accessed 31 May 2020].

The simplicity of the Conway model, can be seen quite clearly in its rule sets. Moreover, its analogies to biological optimization goals and physiological behaviors of its cells, demonstrates the anthropomorphic nature of its semantics. The game of life considers the same base semantics of **Cells** and **States** (as defined in 2.2.3) to establish its basic rule set. However, it doesn't use the Von Neumann neighbourhood, and instead employs the Moore neighbourhood, which can be considered as:

- **Moore neighbourhood** – The group of eight neighbouring cells surrounding a central cell in all directions of a square grid. The neighbourhood has the central cell itself, and it's eight neighbours, like pixels in computer graphics.

The Conway model is defined by the following rule sets (Gardner, 1970)⁹¹ :

- *The environment of the Game of Life is an infinite, two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead, (or populated and unpopulated, respectively).*
- *Every cell interacts with its eight neighbours, conforming to the Moore neighbourhood. At each step, in time, the following transitions occur:*
 - *Any live cell with less than 2 live neighbours dies, by underpopulation.*
 - *Any live cell with 2 or 3 live neighbours lives on to the next generation.*
 - *Any live cell with more than 3 live neighbours dies, by overpopulation.*
 - *Any dead cell with exactly 3 live neighbours becomes a live cell, by reproduction.*

⁹¹ Gardner, M. (1970). MATHEMATICAL GAMES - The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. [online] 223(4), pp. 120-123. Available at: <https://web.stanford.edu/class/sts145/Library/life.pdf> [Accessed 31 May 2020].

Theorizing, Taxonomizing and Prototyping operational Ecosystems in Computational environments

- *These rules, which bear similarities between the automaton and real life, can be summarised into the following:*
 - *Any live cell with 2 or 3 live neighbours (in its Moore neighbourhood) survives.*
 - *Any dead cell with 3 live neighbours (in its Moore neighbourhood) becomes a live cell.*
 - *All other live cells die in the next generation.*
 - *Similarly, all other dead cells stay dead.*
- *The **initial pattern** constitutes the seed of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a **tick**.*
- *Each generation is a pure function of the preceding one. The rules continue to be applied repeatedly to create further generations.*

The minimal and yet anthropomorphic rule sets and the simplicity of a two-state automaton coupled with the availability of higher processing powers and the option of keeping the automaton running (because it's a zero-player game, and fits the definition of being an automaton, i.e., a machine that doesn't require human intervention except for the first step) the Conway model was used by a lot of people (not just mathematicians or computer scientists, but also just computer owners) who ran the automaton enough times to discover very many variations merely generated by choosing different starting conditions. Game of life has thus been found to be capable of creating huge biodiversity of different patterns and cellular organisms that have been extensively studied and classified according to their behavior.

The primary categories of these different cellular organisms are:

- Still Lives – Which remain unchanged across generations or ticks. (These include the Block, Bee-hive, Loaf, Boat and Tub).
- Oscillators – Which oscillate between two configurations for a set duration of ticks. (These include the Blinker, Toad, Beacon, Pulsar and Penta decathlon).
- Spaceships – Which seem to move across the grid. (These include the Glider, Light Weight Spaceship – LWSS, Middle Weight Spaceship – MWSS, and Heavy Weight Spaceship – HWSS).

It is both fascinating and computationally intelligent how none of these cellular organisms were intended or designed while establishing the rule sets in the Cellular Automata, but they were just discovered by different users. It is also quite interesting that the combinations of these above-mentioned cellular organisms also create interesting patterns. Thanks to a very strong user base and community, the most recent being Sir Robin, which was the first truly elementary knight ship (which is a Spaceship that moves two squares left for every one square it moves down, like a knight in chess, instead of moving orthogonally or along the diagonal), discovered by Adam Goucher in 2018.⁹² However, many patterns in the Game of Life eventually become a combination of the primary categories mentioned above; other patterns may be called chaotic but many of these patterns or so-called cellular organisms that emerge out of the Conway model of Cellular Automata termed as ***Methuselah***, which are patterns of fewer than ten live cells which take longer than 50 generations to stabilize (Gardner, 1970).⁹³

⁹² www.Conwaylife.com, (2018). *Forums for Conway's Game of Life*. [online] Available at: <https://www.conwaylife.com/forums/viewtopic.php?f=2&t=3303> [Accessed 17 May. 2019].

⁹³ Gardner, M. (1970). MATHEMATICAL GAMES - The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. [online] 223(4), pp. 120-123. Available at: <https://web.stanford.edu/class/sts145/Library/life.pdf> [Accessed 31 May 2020].

Following are some notable Methuselahs:

- R-pentomino – A cellular automata that starts with 5 cells which stabilizes with 116 cells after 1103 generations.
- The Acorn – A cellular automata that starts with 7 cells which stabilizes with 633 cells after 5206 generations.

Of all the different cellular organisms generated by a Cellular Automata of the Conway model, the **Gliders** are the most peculiar, because they can be synthesized quite easily and minimally, they can be collided with each other to form several other complicated cellular organisms, and they can be used to communicate information over long ranges across the grid just like other Spaceships (the glider possesses specialized ability, as it is the smallest spaceship and thus has very less chance of unintended collisions with other organisms). Gliders have also been collided with several other cellular organisms to generate interesting patterns and useful results.

With the help of different combinations of gliders, it is possible to construct logic gates such as AND, OR, and NOT. It is possible to build a pattern that acts like a finite-state machine connected to two counters, which has the same computational power as a Universal Turing machine, so the Game of Life with unlimited memory and no time constraints is theoretically as powerful as any computer; in other words, “*it is Turing complete*” (Berlekamp, Conway and Guy, 2001).⁹⁴ The Conway model is so versatile, that it has already been implemented into creating an “*open-source 8bit programmable computer*” using Golly⁹⁵ by Nicolas Loizeau in 2016 (Loizeau, 2016)⁹⁶ thereby proving that it’s Turing complete.

⁹⁴ Berlekamp, E. R., Conway, J. H. and Guy, R. K. (2001). *Winning Ways for Your Mathematical Plays*. 2nd ed. Wellesley, Massachusetts: A K Peters, Ltd., p. 276.

⁹⁵ Golly. (2005). England: Andrew Trevorrow and Tom Rokicki.

⁹⁶ Loizeau, Nicolas (2016). *Building a computer in Conway's game of life*. [online] www.nicolasloizeau.com. Available at: <https://www.nicolasloizeau.com/gol-computer> [Accessed 05 May 2020].

The Conway model of Cellular Automata thus demonstrates that it (the concept of cellular automata and the model itself) can be implemented into using not just as a computational model to serve the first and second fields of investigation in Natural Computing (as in computing inspired by nature; simulation and emulation of nature by means of computing) but also in the near future, to serve the third field of investigation (in Natural Computing) – Computing with natural materials, thereby making Cellular Automata as a primary tool to perform Natural Computing.

Owing to the phenomenal computational success of the Conway model, lots of variant algorithms have cropped up thereby proving the versatility of Cellular Automata in the field of Computational Modelling. Many of these variants have alterations in the shape and size of the cells, neighbourhoods, and states. Although the Conway model is essentially based on a 2-dimensional environment, many 3-dimensional variants have also been conceptualized, developed, and tested. Although none of these variants have yet been identified as being Turing complete, many of them have successfully generated healthy biodiversity of cellular organisms similar to the Conway model.

The important aspect of the Conway model that could have direct applicability in the context of Computational Ecosystems in the form of a computational modelling tool to perform natural computing. Moreover, the potential of slight variations in its lexical semantics and rule sets could help the research in generating a strong 3-dimensional variant in order to achieve the research goals. The flexibility of the Conway model, and its sheer ingenuity in reducing the finite states to the basic two, also demonstrates its potential for developing a computational ecosystem. In fact, it is quite by itself, already a performing computational system, owing to the fact that it already works on the principles of a feedback loop (structural coupling between the cell and its neighbours) that is autonomous, autopoietic, and with sufficient context-awareness. Moreover, it further augments the similar parameters previously seen in the Von Neumann model (2.2.3).

2.3.3 Cellular Automata – Stephen Wolfram model

The use, study, and research of Cellular Automata, after the Conway model, has been absolutely different as compared to its original Von Neumann days. Very many researchers ventured into applying Cellular Automata in the theoretical lexicon of mathematics and computational science. Some, however, tried to expand its impact and impressions onto physics and philosophy.

Stephen Wolfram, initiated a research on Cellular Automata by working extensively with one-dimensional Cellular Automata (or Elementary Cellular Automata, explained in detail over the next page) while establishing that these computational models were not just to be studied as computational systems, but as discrete systems. He further deduced that *“simple, natural, questions concerning the limiting behavior of cellular automata are often undecidable, and the consequences of their evolution could not be predicted, but could effectively be found only by direct simulation or observation”* (Wolfram, 1984)⁹⁷.

In the obsession of his hypothesis, that in the way that complicated patterns could arise in natural systems, Cellular Automata could explain this complexity in nature by means of their complexity and undecidability. He also attempted to model neural networks and self-gravitating gases with Cellular Automata to prove his hypothesis, *“but later found that Cellular Automata was an unsuitable model for the simulation of these systems”* (Wolfram, 2002).⁹⁸ However, continuing on this hypothesis, Wolfram went on to sensationalize his claims of a paradigm shift in Science with Cellular Automata at the heart of his claims. Although, he studied Cellular Automata empirically and systematically, which this research intends to employ for establishing its theoretical framework in terms of the lexical semantics.

⁹⁷ Wolfram, S. (1984). Computation Theory of Cellular Automata. *Communications in Mathematical Physics*, 96 (1984), Pp.15-57.

⁹⁸ Wolfram, S. (2002). *A New Kind of Science*. Champaign, IL: Wolfram Media.

For his empirical research, Wolfram established an Elementary Cellular Automata with a specific rule set and a numbering system (called as the Wolfram code) to define various rule sets, which can be defined as followed (Wolfram, 2002):⁹⁹

- **Elementary Cellular Automata** - A one-dimensional cellular automaton where there are **two possible states** (labeled 0 and 1) and the rule to determine the state of a cell in the next generation depends only on the current state of the cell and its **two immediate neighbours**.
- There are **$8 = 2^3$ possible configurations** for a cell and its two neighbours.
- The rule defining the cellular automaton must specify the resulting state for each of these possible configurations, so there are **$256 = 2^8$ possible elementary cellular automata**.
- Each of these rules must be given an ID from 0 to 255 as per the Wolfram code.
- **Wolfram Code** - The code is based on the observation that a table specifying the new state of each cell in the automaton, as a function of the states in its neighbourhood, may be interpreted as a **k-digit number in the S-ary positional number system**.
 - **S** – The number of states that each cell in the automaton may have.
 - **$k = S^{2n+1}$** – The number of neighbourhood configurations,
 - **n** – The radius of the neighbourhood.
- Thus, the Wolfram code for a particular rule is a number in the range from 0 to $S^k - 1$, converted from S-ary to decimal notation.

⁹⁹ Wolfram, S. (2002). *A New Kind of Science*. Champaign, IL: Wolfram Media.

- *It may be calculated as followed:*
 1. *List all the k state configurations of the **neighbourhood** of a given cell.*
 2. *Interpreting each configuration as a number as described above, sort them in descending numerical order.*
 3. *For each configuration, list the state which the given cell will have, according to this rule, on the next iteration.*
 4. *Interpret the resulting list of states again as an S -ary number, and convert this number to decimal. The resulting decimal number is the Wolfram code.*
- The Wolfram code for the numbering system does not specify the size (nor shape) of the neighbourhood, nor the number of states — these are assumed to be known from context.
- Although every Wolfram code in the valid range defines a different rule, some of these rules are isomorphic and usually considered equivalent. By convention, each such isomorphism class is represented by the rule with the lowest code number in it.

Following the above rule sets, the Wolfram Model has helped in generating some interesting Cellular Automata out of the 256 possible rule sets, such as:

- Rule 30 – Displaying non-periodic, chaotic behavior.
- Rule 110 – Turing complete (like the Conway and Von Neumann models).
- Rule 184 – Simultaneously describes many, seemingly different, particle systems.

What's unique about the Wolfram model, however, is the meticulous and thorough study into classifying the rules sets in different categories of complexities, which was not found in the previous models, that emphasized on classifying the different patterns or outcomes based on their behaviors for a specific rule set.

Based on all the rule sets of a Wolfram model, it was found that the behavior of the Elementary Cellular Automata was found to be *“similar to the behavior observed in continuum dynamic systems, with simple rules yielding steady-state behaviors consisting of fixed points or limit cycles, and complex rules giving rise to behaviors that are analogous to deterministic chaos.”* However, all these Elementary Cellular Automata following the Wolfram model evolving from disordered initial states were found to be under one of these classes (Ilachinski, 2001):¹⁰⁰

- **Class 1** – *Nearly all initial patterns evolve quickly into a stable, homogeneous state. Any randomness in the initial pattern disappears.*
- **Class 2** – *Nearly all initial patterns evolve quickly into stable or oscillating structures. Some of the randomness in the initial pattern may filter out, but some remains. Local changes to the initial pattern tend to remain local.*
- **Class 3** – *Nearly all initial patterns evolve in a pseudo-random or chaotic manner. Any stable structures that appear are quickly destroyed by the surrounding noise. Local changes to the initial pattern tend to spread indefinitely.*
- **Class 4** – *Nearly all initial patterns evolve into structures that interact in complex and interesting ways, with the formation of local structures that are able to survive for long periods of time.*

¹⁰⁰ Ilachinski, A. (2001). *Cellular Automata A Discrete Universe*. Singapore: World Scientific Publishing Co. Pte. Ltd. Pp. 808.

Wolfram had conjectured that many of class 4 are capable of universal computation, although it has only been proven for Rule 110. Wolfram in his seminal book, *A New Kind of Science* went above and beyond to prove that the study of Cellular Automata should initiate a new field in science (Physics and Chemistry), and researchers of Cellular Automata should pursue to understand and characterize the computational universe. In the book, Wolfram also attempted to demonstrate “*simple programs that exhibit phenomena like phase transitions, continuum behavior, and thermodynamics that are familiar from traditional science*” (Wolfram, 2002).¹⁰¹

However, his theory and claims were highly criticized to be abrasive and arrogant. The lack of scientific methodology is the main theme for most of these criticisms. The research agrees that the Wolfram model provides deep insight into the study and classification of the rule sets, but “*just because the patterns of cellular automata can resemble those of the natural world does not mean that nature must work that way*” (Gad-el-Hak, 2003).¹⁰²

In the relevance of the research, the four classes identified by the Wolfram Model could be considered very helpful in categorizing Computational Ecosystems according to predetermined lexical, logical, and behavioral parameters as part of the Operational Objectives I (Taxonomies, as previously explained in 1.2.1 Objectives for Computational Ecosystems). It would be also astute to determine a system that would perform analogous to the Wolfram Code (that establishes semantics for the rule-set number assigning system). This system would also be helpful in further simplifying the nomenclature system for each ruleset, as in the case of the research, a 3-dimensional grid system would be considered as a base for any neighbourhood, and that would directly lead to an exorbitant amount of results, no matter the total number of determined states.

¹⁰¹ Wolfram, S. (2002). *A New Kind of Science*. Champaign, IL: Wolfram Media.

¹⁰² Gad-el-Hak, M. (2003) *A New Kind of Science - Review*. *A New Kind of Science*, by S. Wolfram. *Applied Mechanics Reviews*, 56 (2), pp. B18-B19.

2.3.4 Cellular Automata – applications in the AEC Industry

Yet again, the AEC industry was slow to catch up on the phenomenal revolution Cellular Automata was creating in a wide range of scientific and artistic disciplines, by almost five decades. But eventually, it happened in the form of a didactic approach in architectural design. In 1995, John Frazer (an architectural academic) in a lecture at the AA (Architectural Association School of Architecture, London) quoted -

*I'm dedicating this lecture, to the first building intelligent enough to understand and appreciate the gesture. I confidently expect that such a **building** will have **designed and constructed itself** in response of the needs of its users and acting in harmony with its environment. It will be **self-sustaining**, it will exhibit metabolism, it will derive from all of its environment and be **controlled by a symbiotic relationship between its inhabitants and all of that environment**. And when it has outlived its usefulness, it will **self-destruct and redistribute its resources**. This lecture is to that first building, and I hope that it won't be too long (AA School of Architecture, 2015, 02:07).¹⁰³*

With this proclamation, Frazer intended to introduce a new branch of science concerned with creative morphology and intentionality, wherein he proposed the architectural design industry to “*search for a design theory based on form-generation developed for architectural purposes*” (Frazer, 1995).¹⁰⁴ Driven by the computational advances of the Turing Machine, and Von Neumann’s universal constructor, Frazer developed his own version of a universal constructor (in deference of von Neumann) was able “*to respond on a rule-based system of constructing a 3D self-replicating automata in response to an obstacle*” (Frazer, 2001).¹⁰⁵

¹⁰³ AA School of Architecture (2015) *John Frazer - An Evolutionary Architecture*. 02 May. Available at: <https://www.youtube.com/watch?v=58ZUhdKaRC8> (Accessed: 18 Dec. 2017).

¹⁰⁴ Frazer, J. H. (1995). *Themes VII: An Evolutionary Architecture*. London: Architectural Association, p. 127.

¹⁰⁵ Frazer, John H. (2001) *The Cybernetics of Architecture: A Tribute to the Contribution of Gordon Pask*. *Kybernetes*. The International Journal of Systems & Cybernetics. 30(5/6). pp. 641-651.

Inspired by the Von Neumann, Conway and Wolfram models, Frazer's universal constructor had the following predeterminations (Frazer, 1995)¹⁰⁶:

- **Cell** – *A unit cube of unit dimensions, owing to its self-similarity could represent anything, and could be modelled at any scale.*
- **States** – *Each cube could have any of the 256 states which were displayed by means of LED lights that were embedded in these cubes. The 8-bit code could be used to map the state of the cell to any form or structure: to environmental conditions such as wind; to sound, or even to dance. The stack of cells could communicate information for the LED display, where the displays would have specific codes depending on their state – like blinking patterns for adding or deleting cells.*
- **Neighbourhood** – *A 12x12x12 cell array would be considered as the entire system or termed as a **landscape**, however the cell considered a neighbourhood similar to a Moore neighbourhood.*

The universal constructor was a truly successful proof of concept, that capitalized on the technology based on the late 20th century electronics. The constructor served as a physical modelling tool that was embedded with its own ruleset (serving as an intelligent 3D extension of a 2D cellular automata). Although not autonomous or autopoietic (as the system merely blinked LEDs), the constructor was surely context aware. Many other such applications were attempted including an evolutionary model and later *“compiled as part of architectural discourse in order to explore beyond an algorithmic approach of generative and self-organizing architecture and to investigate systems which learned on the basis of feedback”* (Frazer, 2001).¹⁰⁷

¹⁰⁶ Frazer, J. H. (1995). *Themes VII: An Evolutionary Architecture*. London: Architectural Association, p. 127.

¹⁰⁷ Frazer, John H. (2001) *The Cybernetics of Architecture: A Tribute to the Contribution of Gordon Pask*. *Kybernetes*. The International Journal of Systems & Cybernetics. 30(5/6). pp. 641-651.

Another notable experimentation implementing cellular automata as a rule-based system of mathematical constructions and concepts to investigate the process of generating architectural forms is “*Krawczyk’s Architectural Interpretation of Cellular Automata*” (Krawczyk, 2002)¹⁰⁸. Krawczyk expanded on the applications of a 3D cellular automata that was proposed by Schrandt and Ulam based on the Von Neumann model as “*empirical results obtained by experiments on computing machines*” (Schrandt and Ulam, 1967)¹⁰⁹. Krawczyk’s results and conclusions are quite elementary as compared to those done by means of extensive, and thorough experimentations by Frazer. But while experimenting with the rule sets (based on the precedents of the cells, states, and neighbourhoods set up for a 3D Cellular Automata based on the Schrandt-Ulam model), Krawczyk makes the following observation that could be quite essential for the theoretical foundations of this research –

- *The pure mathematical translation of a cellular automata into architectural form includes a number of issues that **do not consider built reality**.*
- *The interpretation or translation to a possible built form can be dealt with after the form has evolved or it **can be considered from the very beginning**.*

As the early 2000s saw a rise in implementing novel computational techniques in a diverse range of design disciplines, such as urban design and urban planning. Herr and Kwan made a similar approach of implementing Cellular Automata as a “*generative architectural design strategy for high-density residential architecture*” (Herr and Kvan, 2005)¹¹⁰.

¹⁰⁸ Krawczyk, R. J. (2002). Architectural Interpretation of Cellular Automata. *Generative Art 2002*. pp. 7.1-7.8.

¹⁰⁹ Schrandt, R. G., Ulam, S. M. (1967). *On Recursively Defined Geometrical Objects and Patterns of Growth*. [online] Los Alamos, New Mexico: Los Alamos Scientific Laboratory of the University of California, p. 19. Available at: https://digital.library.unt.edu/ark:/67531/metadc1027179/m2/1/high_res_d/4573212.pdf [Accessed 05 May 2018].

¹¹⁰ Herr C.M., Kvan T. (2005) Using Cellular Automata to Generate High-Density Building Form. In: Martens B., Brown A. (eds) *Computer Aided Architectural Design Futures 2005*. Dordrecht: Springer, p. 10.

Apart from implementing Cellular Automata “for a very large scale, large density architectural purpose” (Moreno and Grinda, 2004)¹¹¹, Herr and Kvan propose a novel methodology inspired by classical Cellular Automata with enhanced properties that is described as followed –

- To accommodate both generative and traditional design procedures, the implemented cellular automata may be used in phases, with ***intermittent stages of manual design interventions***.
- ***Cell behaviors can be assigned dynamically*** during the design process, such that elements within the modelling environment can change their behavior over time.
- In contrast to classical cellular automata, where cells are uniform and cell states do not affect cell geometry, functions can be assigned to any element in the modelling environment, with ***cells able to change their geometry in response to their states***.

“Compared to a conventional generic high-resolution approach, this non-uniform solution greatly limits the number of cells required in modelling architectural geometries and avoids the restrictions imposed by the compulsory use of additive approximation based on homogeneous grids of elements” (Herr and Kvan, 2005)¹¹². These key improvements in traditional cellular automata ruleset, that involve dynamically modifying the cell states, depending on any external condition or requirement would be very essential in making the Cellular Automata and consequently the Computational Ecosystem truly context-aware.

¹¹¹ Moreno, D., Grinda, E. G. (2004). Soft Metropolitanism [Apartments in Micro-Skyscrapers]. In: F. Marquez Cecilia, and R. Levene, ed., *EL CROQUIS 118: CERO 9, ABALOS & HERREROS, NO.MAD*, 1st ed. Madrid: El Croquis, pp. 140-147.

¹¹² Herr C.M., Kvan T. (2005) Using Cellular Automata to Generate High-Density Building Form. In: Martens B., Brown A. (eds) *Computer Aided Architectural Design Futures 2005*. Dordrecht: Springer, p. 10.

Cellular Automata, in 2010, has also been implemented “to simulate how simple rules could emerge a highly complex architectural designs of some Indonesian heritages” (Situngkir, 2004)¹¹³. Here, Situngkir implements Cellular Automata as an “exploratory tool based upon the 3D cellular automata constructed within the totalistic 2D cellular automata with 9 neighbours” (Packard & Wolfram, 1985).¹¹⁴ He also observes that some of the 9th and 15th century Indonesian temples resemble Class I cellular automata (explained in 2.3.4). Although these observations do not help the research in establishing any lexical semantics or theoretical framework, it certainly helps the research in understanding that even the Class I Cellular Automata (the one without any complexity or the one that is absolutely devoid of Turing completeness) could serve as an interesting computational blueprint to design a built form that is structurally stable, functionally efficient and aesthetically inspiring.

Furthermore, in 2018, design researchers at the Bartlett School of Architecture, UCL London explored the application of machine learning to combinatorial design-assembly from the scales of building to urban form. “Connecting the historical lines of discrete automata in computer science and formal studies in architecture their research contributed to the field of additive material assemblies, aggregative architecture and a possible upscaling to urban design” (Koehler et al, 2018).¹¹⁵ Their research implements a term Mereology (set of recursive assembly strategies, integrated into the design aspects of the building parts), stating that architectural arrangements can be described as chaining and nesting of multiple discrete systems, which can be used to nest discrete patterns scaled to an urban form.

¹¹³ Situngkir, H. (2010). Exploring Ancient Architectural Designs with Cellular Automata. [online]. Available at:

https://www.researchgate.net/publication/2146550_Epidemiology_Through_Cellular_Automata_Case_of_Study_Avian_Influenza_in_Indonesia [Accessed 31 Aug. 2019].

¹¹⁴ Wolfram, S. and Packard, N. H. (1985). Two-Dimensional Cellular Automata. *Journal of Statistical Physics*, 38, pp. 901-946

¹¹⁵ Koehler, D., Saleh, S. A., Li, H., Ye, C., Zhou, Y., Navasaityte, R., (2018). Mereologies - Combinatorial Design and the Description of Urban Form. In: *GENERATIVE DESIGN - Volume 2 - eCAADe 36*. Łódź, Poland: eCAADe, Faculty of Civil Engineering, Architecture and Environmental Engineering Lodz University of Technology, cop. 2018. pp. 85-94.

2.4 Theoretical Assumptions for Computational Ecosystems

Although thus far Cellular Automata has been established as a potential computational modelling environment for this research, some key observations are necessary to be highlighted before proceeding to establish the logical semantics and the foundation for the theoretical assumptions for the research.

(As seen in 2.2.3,) John Von Neumann was able to exhibit a universal Turing machine embedded in a cellular array using 29-states per cell and the 5-cell neighbourhood. Although, Langton makes a remarkable observation in 1984, that “*Von Neumann’s Turing machine is suitably modified so that, as output, it can **construct** in the array any configuration which can be described on its input tape. His machine (the universal constructor) would thus be programmed to construct any machine described on the input tape, and create a copy of the input tape and attach it to the machine created. This shows two levels of automaton in the Von Neumann model*”(Langton, 1984)¹¹⁶ –

1. The **cellular automaton itself** which is embedded on the input tape.
2. **The input tape** with the constructor which in **itself is an automaton**.

Langton refers to this input tape as a cyclic storage loop that is capable of representing the universal constructor (while serving the predetermined functions of the constructor) as well as creating variant (or identically similar) offspring of the parent constructor. He also identifies these loops to be simple structures, (which are not necessarily exhibiting universality), employing transcription and translation in their reproduction capabilities. Langton’s conclusions can be implemented in the research to identify that the feedback loop generated as part of the Cellular Automaton could be considered autonomous and autopoietic.

¹¹⁶ Langton, C. G. (1984). Self-Reproduction in Cellular Automata. *In: Physica 10D*. Amsterdam: Elsevier Science Publishers B.V. (North-Holland Physics Publishing Division), pp. 135-144.

Thus, Cellular Automata can be implemented in the research to serve as a computational modelling environment, that has vast computational capabilities, research precedence, and several examples of applications in the AEC industry. It can also be established that Cellular Automata can provide a versatile domain to craft ***Bio-plausible Hybrid Bio-inspired Stochastic Optimization***, which can serve as an autonomous, autopoietic, context-aware feedback loop that develops a dynamic, reciprocal, symbiotic relationship between the ***built form*** and the ***algorithm***.

However, Cellular Automata have been identified for some computational issues. Expanding on all the three models explained in this thesis thus far (i.e. Von Neumann model, Conway model and Wolfram model), it can be observed that all these models are undecidable, meaning given an initial pattern and an outcome, no algorithm can determine whether the outcome is possible or probable. *“This is a corollary of the halting problem: the problem of determining whether a given program will finish running or continue to run forever from an initial input”* (Berlekamp, Conway and Guy, 2001).¹¹⁷ This makes the unpredictable nature of the algorithm quite computationally expensive (requiring higher processor speeds and memory cores), and requires the halting problem to be tackled with some creative computational ingenuity.

Moreover, Oxman observed in her doctoral thesis that, *“when cellular automata and genetic algorithms are combined with some finite-element method, there is tremendous disparity between the actual form generated by the script or the program, and its material properties and behavior relative to the anticipated functions.”* She thus suggests, *“as we aim to unite between generation, evaluation and eventually fabrication, we must look beyond current approaches in design computation that support and promote seamless integration between the digital and the physical domains”* (Oxman, 2010).¹¹⁸

¹¹⁷ Berlekamp, E. R., Conway, J. H. and Guy, R. K. (2001). *Winning Ways for Your Mathematical Plays*. 2nd ed. Wellesley, Massachusetts: A K Peters, Ltd., p. 276.

¹¹⁸ Oxman, N. (2010). *Material-based Design Computation*. PhD Thesis. Massachusetts Institute of Technology.

Several previous applications of Cellular Automata in the AEC industry have also been observed to have programmed in very limited generative capacity, and especially for top-down developmental control and for pragmatic rule-breaking (Herr and Ford, 2015).¹¹⁹ The authors argue the following issues –

- *While other fields have developed their own Cellular Automata models to fit their subjects of study more appropriately, Architecture (and the AEC industry on the whole) has not yet established a common theoretical framework outlining the potential of Cellular Automata in architectural design processes.*
- *A generative design tool stems from its capability to perform tasks that rely on numerically formalized dimensional or relational constraints, design decisions however require more holistic, experienced, and yet intuitive context-based understanding and decision making that is difficult to be translated in a binary logic, and hence has been hitherto unexplored.*
- *Generally, the cellular automata rule sets are typically used as fully automated systems that do not allow for much user intervention during run-time, limiting the designer's role to select from a range of potential solutions once the generative process has finished.*
- *Fully automated generative design processes rely on formalizable evaluation methods to distinguish appropriate solutions from others automatically in order to produce meaningful results in their respective design contexts.*

Thus, the implementation of Cellular Automata in the context of the research should be done while considering the above observations.

¹¹⁹ Herr, C. M., Ford, R. C. (2015). Adapting Cellular Automata as Architectural Design Tools. In: *Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2015*. Hong Kong: The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), pp. 169-178.

To conclude on the theoretical assumptions for Computational Ecosystems, it is essential to summarize the lexical semantics in the context of the hypothesis, methodologies, and objectives (as per chapter |1). Thus (as per 2.1 and 2.2), a Computational Ecosystem would be defined as a **Hybrid Bio-inspired Stochastic Optimization** that is –

- **A context-aware system** that represents an autonomous, autopoietic feedback loop that is based upon the structural coupling of two or more entities which would be essential in establishing a dynamic, reciprocal, symbiotic relationship between the **built form** and the **algorithm**.
- **A discrete system of elements**, that experience homogenous and heterogenous cooperation clearly represented by their intraspecific and interspecific relationships, thus creating a fluid design workflow that performs modelling, analysis, and fabrication simultaneously to generate form, structure, and enclosure for architectural intent.
- **A symbiotic system**, that is equipped with communication strategies at both the inter-habitat and intra-habitat levels which would be instrumental in making computational design more autonomous and digital fabrication more data driven.
- Realized **by employing Cellular Automata**,
 - Which is not just based on one or on a combination of some or all the three models explained in this thesis thus far, but is also sufficiently modified, taxonomized and versioned so as to accommodate the requirements of the architectural intent it serves.
 - Which offers an organizational solution to the halting problem by ingesting the concept of **decay** in a manner relevant to the ecosystem.

Theorizing, Taxonomizing and Prototyping operational Ecosystems in Computational environments

- Which simplifies intuitive design decisions in a non-binary logic, by allowing user intervention during run-time and thus providing seamless integration between the digital and the physical domains.
- Which offers a step-wise methodology serving as a blue-print for the **Architecture** of potential **Computational Ecosystems**.

Objectives

3| On the semantic syntax for the Computational Ecosystems.

3.1 Lexical Semantics from theoretical assumptions

After establishing the Literary Objectives-I, related to the lexical semantics (as in 1.2.1, and explained in thorough detail in 2|) the definitions for Computational Ecosystem along with their theoretical frameworks and operational precedents have been outlined. The intent of implementing Cellular Automata as the primary computational modelling environment to architect Computational Ecosystems has also been sufficiently theorized. However, to set up the Computational Ecosystems (and the Cellular Automata), the research needs to establish a strong semantic syntax that could ease the functionalities of the operational objectives. The Semantic Syntax (analogous to a language) can be seen as an amalgamation of the lexical semantics (analogous to the vocabulary of a language) and the logical semantics (analogous to the grammar of a language). Moreover, as the analogous vocabulary will be used to establish the analogous grammar in this case, understanding and setting up the logical semantics for the research would be based on the lexical semantics.

Although a theoretical foundation has been quite effectively and thoroughly laid in the previous chapter (2|), the precise semantics that would be applied in the research have not been clearly identified so far. It thus becomes essential to extract the lexical semantics from the theoretical assumptions. The Computational Ecosystem (pertaining to this research) for all purposes beyond this point, would be denoted by its abbreviation **CE**. Similarly, for all purposes beyond this point the Cellular Automata (pertaining to this research) will be denoted by its abbreviation **CA**. Although both these abbreviations CA and CE are widely accepted, the thesis is introducing them now, firstly, to avoid losing relevance to tags (as in tags required for search optimization), and secondly to establish significant distinction to the canonical use of these abbreviations.

As elaborated in the previous chapter, **classical Computational Ecosystems** which basically represent an optimization algorithm are constituted of –

- Participating individuals called as **Species** – These could be Bacteria, Ants, or Fish as in a BFA, ACO or FSA respectively (as illustrated in 2.1.3); These could also be represented by agents that serve as computational counterparts of the species.
- A **Community dynamic** within the species – For example by means of Chemotaxis in a BFA, or by Stigmergy in an ACO, or by the search-swarm-follow mechanisms in an FSA. It can also be promoted through the trade of token units of energy and biomass between these agents (as per 2.2.2).

Also, as explained and illustrated previously, **classical Cellular Automata** which basically represent a modelling environment are constituted of –

- **Cells** – As seen in previous examples, these could be considered as a varied form of geometrical arrays, from basic orthogonal shapes to discrete or combinatorial geometries (such as polyhedral, polytopes and packing).
- **States** – As already defined, these could be considered as a combination of rule sets which could either be distinguished on the basis of a binary logic, or on the basis of RGB values (like in the Von Neumann Model as per 2.3.1).

(As introduced in 1.2.2,) the components of a **CE** realized by a **CA** would be –

- **Elements** – The **CE** equivalent of **Species** and **Cells**.
- **Economy** – The **CE** equivalent of **Community dynamics** and **States**.

The significance of **Elements** and **Economies** pertaining to **Ecosystems** will be explained in further sections.

3.1.1 Element

To set up a methodological framework, for the construction of **CE** based on **CA**, it is important to establish the lexical semantics in such a way that the terminologies serve as free variables or **placeholders** (symbols that will later be replaced by a value) for a diverse range of applications.

The simplest definition of **Elements** for the **CE** would be the equivalent of **Species** from a traditional Computational Ecosystem (more specifically, a canonical computational ecosystem - ECO) or an equivalent of the **Cells** of a traditional Cellular Automaton (that could be from any cellular automata model discussed in 2|). To install this terminology in the research as a placeholder, however, some more definition, explanation and exemplification is required. Thus, previous use of the term element in a diverse range of disciplines, would be an ideal direction to start this exploration. Apart from providing insights into the use of the terminology as a placeholder, it would also help the research in determining a definition for the placeholders that is based on first principles (or **ab initio**), rather than the highly analogous (or **empirical**) definitions mentioned in 3.1.

The term element has been repeatedly mentioned in a wide range of ancient cultures in Greece, Ancient Egypt, Persia, Babylonia, Japan, Tibet, and India to explain the nature and complexity of all matter in terms of simpler substances (the elements considered as the building blocks of the universe). Although there are varied inclusions of different elements, the most commonly found are – earth, water, wind and fire. These philosophical constructs have also been later expanded upon in all the above-mentioned cultures to explain the concepts of consciousness through theology. However, one metaphysical branch of Hinduism – **Sankhya**, stands out and postulates only two forms of elements in the universe – **Purusha** (consciousness) and **Prakriti** (matter). This duality can also be translated into the ecosystem components of the **biotic** and the **abiotic**.

This understanding of the term element as a constituent building block of the universe was later expanded into Chemistry as terminology for the 118 distinct building blocks of the observable universe which cannot be broken down into simpler substances by chemical means (as introduced in 1.2.2). It is also defined as a substance that is made entirely from one type of atom (as in, the element hydrogen is made from atoms containing a single proton and a single electron and if you change the number of protons an atom has, you change the type of element it is). The concept of an element as a representative of a particular kind of constituent parts (in this case atoms) is similar to the concept of a **Species** that is representative of a class of participants that share the same genetic make-up.

In a more mathematical sense, i.e., in the context of the Set Theory in Mathematics, an element is like a member of a set and is any one of the **distinct objects** of that set (here, distinct objects are those numbers, sets, functions, expressions, geometry, mathematical transformations, and spaces which are not mathematically equal). Thus, elements are seen as the constituent building blocks of a set. Moreover, in the notational system implemented in the set theory, while denoting that a certain numerical entity **is an element of** a set, the symbol \in is used. Perhaps, the research can also use the symbol \in to represent an element in a **CE**.

The term element, however, is already used as a placeholder in the field of Computational sciences. A term similar to the one used in Set Theory is implemented in the Unified Modelling Language (UML) (which is a generalized, universal notational system to represent a pseudo-code graphically while it is in its developmental stages), which represents a similar notion of an abstract class that has no superclass (as in an unprecedented semantic). In Metadata (a branch of data sciences, where data is embedded in existing data), also, the term element is used to define an atomic unit of data that has precise meaning or precise semantics. As metadata is also implemented in a wide range of industries such as imaging, telecommunications, videography, geospatial mapping, data warehousing, and cloud applications amongst many other, it has a significant definition for a **data element**.

From all the usages of element in the different fields of study such as philosophy, chemistry, mathematics, and computational sciences stated previously, the research would focus on the placeholder of an **Element** summarized as follows –

- The **Element** is a distinct object, that would represent different components of a **CE** (similar to the Species of a canonical ECO algorithm).
- The **Element** could represent both biotic or abiotic components of the **CE**, and this characteristic would serve as distinctions amongst different elements.
- The **Element** would also be identified by a name, and a clear definition. If required, further categorization of elements should also be done.
- To avoid confusion between different terminologies within the context of this research, the **Element** would be abbreviated as ϵ , and various distinctions could be symbolized by using appropriate subscripts, such as ϵ_b and ϵ_a (for biotic and abiotic elements respectively).
- (As introduced in 1.2.2) Examples of an ϵ could be Platonic solids, Archimedean solids, point clouds, passive agents, active agents, cognitive agents, service equipment, structural members or fabrication material.
- As during the course of the operational objectives of this research, a **CE** could have multiple ϵ , all the particular elements would have to be classified, named, defined, and symbolized based on a wide range of factors.
- To make the **CE** context-aware, some elements would also have to be capable of being introduced by external factors (such as the user, or a conditional, or depending on context-specific factors), which will have to be classified, named, defined, and symbolized based on the predetermined external factor.

3.1.2 Economy

With the placeholder, **element** (ϵ) thoroughly defined and established, as part of the literary objectives of stating the logical semantics, the thesis proceeds to define and establish the placeholder of **economy**. Unlike the element placeholder, the economy placeholder does not have any similar precedents in the fields of computational science or mathematics. Thus, understanding the implementation of the placeholder becomes more essential in this case.

(As theorized in 2.1.1,) The biotic and abiotic inhabitants (or elements) of an ecosystem are inadvertently striving for a certain state of dynamic equilibrium and while attempting to attain this state of dynamic equilibrium, the ecosystem needs a specific predetermined currency within the system to evaluate, govern, and maintain its state of equilibrium. A **CE** (as theorized in 2.4) performs as a discrete system of elements, that experience homogenous and heterogenous cooperation that is represented by intraspecific and interspecific relationships.

However, to establish these relationships, the **CE** would need a system of rules, that would help in performing a system of checks and balances on its states of equilibria. That is, to evaluate if the equilibrium has a deficit or an excess (as in if the state of equilibrium was not achieved, was it underdone or overdone and by how much), while exploring what other factors could help in maintaining a steady equilibrium (like exploring if the rule-set needs to modify to accommodate the state of equilibrium), and finally to document all possible states, of equilibria or otherwise (as in maintaining a temporal ledger of all the states across the runtime of the **CE**). These rule sets would have to be assigned to all the ϵ (predetermined or user added) of the **CE**, moreover the rule sets would serve as a global currency of the **CE** making sure that all the ϵ have the same goals of equilibria across the runtime of the **CE**.

Thus, in the simplest terms, these rule sets or **Economy**, would serve as a logical equivalent of **Community dynamics** and **States** (as described in 3.1).

Although, as mentioned previously, the economy placeholder is not as widely formulated as the element placeholder, the precedents of the term **Economy** are found in Greek history, to mean household management (as in the management of household resources). Even though the modern meaning of economy – “as a social domain that emphasizes the practices, discourses, and material expressions associated with the production, use, and management of resources” (James, 2015)¹²⁰ – essentially deals with larger test sets than households, the definition still revolves around its etymological roots of resource management. This consistency in the essence of the term economy over the ages, could serve as an astute reference for the ab initio definition of Economy for the **CE**, that is, the placeholder would be considered as a global currency for the resource management within the system to evaluate, govern, and maintain its state of equilibrium.

In economics (concerning modern economies of nations or cultures), the following degrees of precedence are considered for the economy to survive and progress –

- Primary stage – Extraction of raw materials from their natural sources.
- Secondary stage – Transforming the raw materials into consumer goods.
- Tertiary stage – Providing services to businesses and consumers.
- Quaternary stage – Research and development required to perform the above stages.

In the context of the research, the above stages can serve as an example for the different stages of deployment for different **Economies** during the runtime of the **CE**.

¹²⁰ James, P. (2015). *Urban Sustainability in Theory and Practice – Circles of Sustainability*. New York: Routledge, pp. 260.

From all the precedents of economy in history and economics stated previously, the research would focus on the placeholder of an **Economy** summarized as follows –

- The **Economy** is a distinct rule set, that would represent different states for the different $\mathbf{\epsilon}$ of a **CE** (similar to the States of a cellular automaton).
- The **Economy** could represent a wide range of rules pertaining to the evaluation, maintenance, and governance of the state of equilibrium of a **CE**, and this **Economy** or **Economies** could be assigned to one or many $\mathbf{\epsilon}$.
- The **Economy** should also be identified by a name, and clear definition. If required, further categorization of economies should also be done. Just like the $\mathbf{\epsilon}$, some context aware $\mathbf{\Psi}$, which would be introduced externally would have to be named, defined, and symbolized accordingly.
- To avoid confusion between different terminologies within the context of this research, the **Economy** would be abbreviated as $\mathbf{\Psi}$, and various different distinctions could be symbolized by using appropriate subscripts, such as $\mathbf{\Psi}_p$ and $\mathbf{\Psi}_s$ (for primary and secondary economies respectively).
- Because the $\mathbf{\Psi}$ represent the cell states of the **CE**, the $\mathbf{\Psi}$ could also be further elaborated to accommodate a specific notation system based on Wolfram code (as described in 2.3.3).
- The stages of deployment of the $\mathbf{\Psi}$ should also be encoded into the notation system by adding superscripts such as $\mathbf{\Psi}_p^1$ and $\mathbf{\Psi}_p^2$ (mentioning the correct order of deployment in a numerical order).
- (As introduced in 1.2.2) Examples of an $\mathbf{\Psi}$ could be static structural stability, kinetic structural stability, functional adequacy, functional compatibility, contextual compatibility, climatic optimization, and fabrication constraints.

3.1.3 Ecosystem

An ecosystem has been amply theorized in this thesis so far. Moreover, by establishing the syllogism through Computation – Ecosystem – Computational Ecosystem, the terminology has also had sufficient analogical precedents (in a wide range of industries and research domains). However, the inclusion of the lexical semantics pertaining to the Elements (ϵ) and the Economy (Ψ), make it absolutely essential to expand the definitions of **CE** in order to accommodate the same format. Thus, in spite of establishing a theoretical definition for a **CE** (as per 2.4), and identifying the components of a **CE** (as per 3.1), it is essential for the thesis to establish an operational ab initio definition. While researching Computational Ecosystems (as per the canonical ECO algorithm), Parpinelli and Lopes propose that the following features can still be explored in the ecological framework (Parpinelli and Lopes, 2014).¹²¹ –

- 1. The environment can be explored with the insertion of abiotic components biasing the behavior of populations.*
- 2. By using some source of feedback from the optimization process during its course, the habitats formation can be better distributed, as well as the intra and inter-habitat communication topologies, can be better defined.*
- 3. Flow of information (stigmergy) & energy (trophic structures) can be explored.*
- 4. Strategies and metrics for maintaining the diversity of solutions both at micro and macro levels can be applied.*
- 5. The Computational Ecosystem framework can be explored asynchronously.*

¹²¹ Parpinelli R. S. and Lopes, H. S. (2014). A computational ecosystem for optimization: review and perspectives for future research. *Memetic Computing*, 7(1), pp. 29-41.

Apart from the aforementioned proposals, an illustrative map of the elements of Computational Ecosystems was also developed by Parpinelli and Lopes, as shown in figure 3.1.

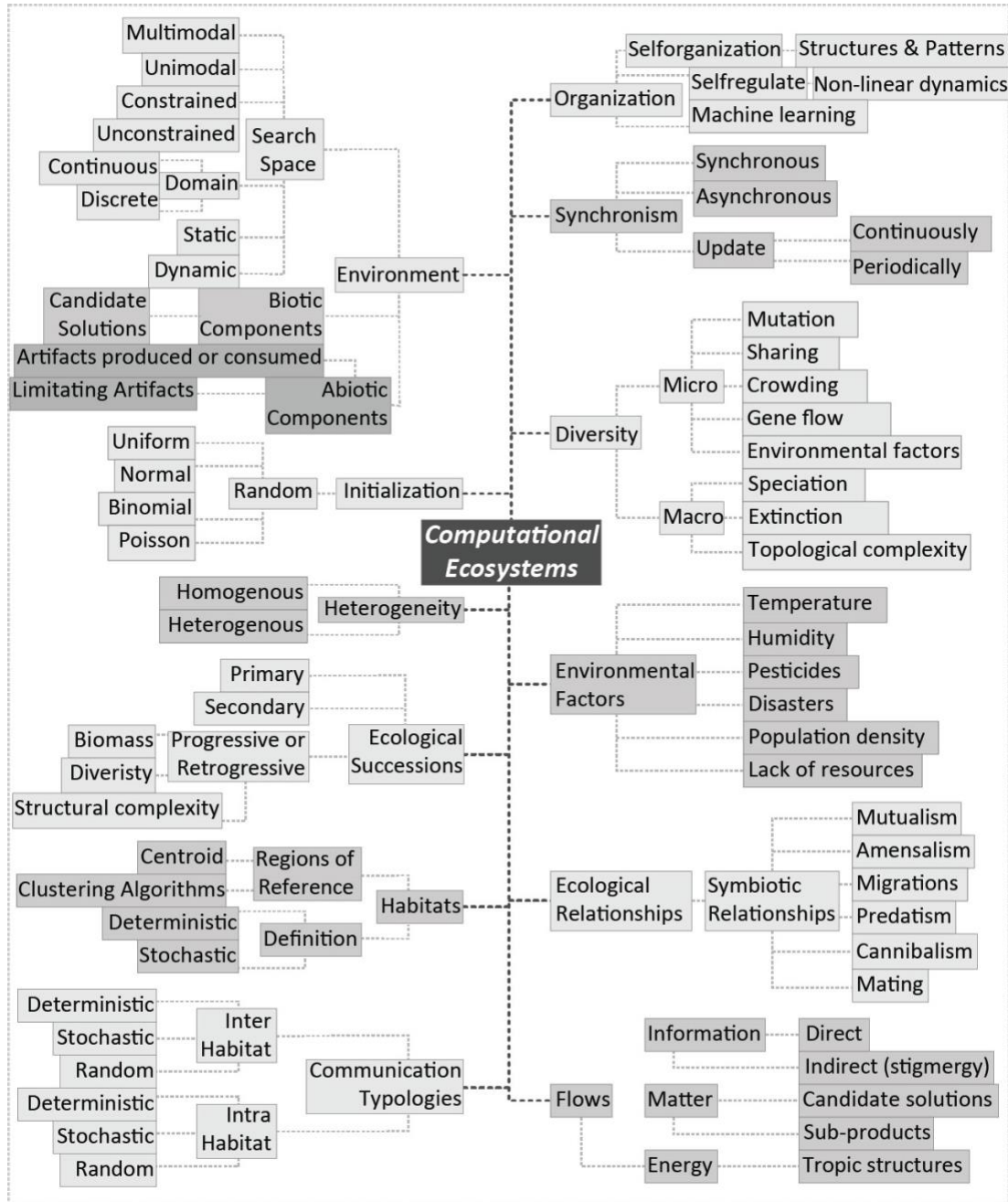


Figure 3.1 – Graphical representation of extended illustrative map for the elements of a computational ecosystem. Original image by Parpinelli and Lopes (December 2014) Illustration and graphics by Author (January 2021).

Considering all the precedents of ecosystem pertaining to all the explorations in the field of computer sciences illustrated previously, the research would focus on the placeholder of a **CE** summarized as follows –

- The **CE** is the modelling environment where the ϵ interact with each other depending on the goals set up by their respective Ψ (similar to the functioning of a **CA**) while it performs as a **Hybrid Bio-inspired Stochastic Optimization Algorithm** (as per 2.4) with its constituent attributes (as per 3.1).
- Owing to the possibility of multiplicity in the ϵ and the Ψ , several numerous **CE** can exist at the same time. These **CE** could also be nested into each other while performing at different scales or different stages of deployment in the runtime of their parent **CE**, or both.
- The **CE** should also be identified by name, and clear definition. If required, further categorization of **CE** should also be done. Just like the ϵ and the Ψ , some context aware **CE**, which would be introduced externally (if that happens) would have to be named, defined, and symbolized accordingly by adding superscripts such as **CE¹** and **CE²** (mentioning the correct order of deployment in a numerical order).
- Following the proposals by Parpinelli and Lopes (mentioned in 3.1.3), apart from recursive algorithms, genetic algorithms should also be used to explore points 2 and 5 in the **CE**.
- Although points 1 and 3 have been sufficiently covered in the definitions of ϵ and Ψ respectively, 4 could be tackled smartly by implementing the initialization properties of the **CE** as illustrated in fig. 3.1.
- Throughout the introduction of new **CE** and the corresponding ϵ and Ψ , fig. 3.1 should be referred to, illustrated, and updated accordingly.

3.2 Establishing Logical Semantics for operational objectives

As the lexical semantics of the research – Elements (ϵ), Economies (Ψ) and Ecosystems (CE) are established on the basis of the theoretical assumptions in the previous sections, some essential explanations are yet to be made before setting up the logical semantics. Moreover, as these logical semantics will be implemented undeviatingly to perform the operational objectives, it becomes essential to identify and theorize all aspects of anomalies and deviations from the definitions mentioned thus far. One important aspect that exemplifies the anomalies in the theoretical logic is the proposed multiplicity of the ϵ and Ψ (as per 3.1.1 and 3.1.2). This potential multiplicity can pose problems of confusion and redundancy throughout the research methodology. Moreover, it can't be merely solved by adding superscripts in the notation system (although the notation system for the ϵ , Ψ , and CE with the use of subscripts and superscripts to show multiplicity will be sustained and implemented).

This problem needs to implement a solution that already exists in the field of computer organization and resource hierarchies introduced by Michael J. Flynn (As introduced in 1.2.2). He developed “a hierarchical model for computer organizations commonly identified as Flynn’s Taxonomy which is still used as a design tool for modern processors and their functionalities.” It is as followed (Flynn, 1972)¹²² –

- *SISD – Single Instruction stream, Single Data stream.*
- *SIMD – Single Instruction stream, Multiple Data streams.*
- *MISD – Multiple Instruction streams, Single Data stream.*
- *MIMD – Multiple Instruction streams, Multiple Data streams.*

¹²² Flynn, M-J. (1972). Some Computer Organizations and their Effectiveness. *IEEE Transactions on Computers*, C-21(9), pp. 948-960.

Elaborating on Flynn's Taxonomy, the problem with the multiplicity of the ϵ , Ψ , and thus consequently the multiplicity of CE , could be solved by concluding that the placeholders of ϵ and Ψ are the parameters which can be considered as variables. Furthermore, the capacity of ϵ and Ψ can be expanded (as introduced in 1.2.2) by considering the following assumptions based on the lexical semantics established thus far –

- A CE can only exist if the ϵ and the Ψ are established.
- A CE can exist with a single ϵ and a single Ψ .
- A CE can exist with multiple ϵ and a single Ψ .
- A CE can exist with a single ϵ and multiple Ψ .
- A CE can exist with multiple ϵ and multiple Ψ .

Thus, it can be stated that there can be only four possibilities of variations or deviations considering the potential of multiplicity of ϵ and Ψ . Moreover, based on the theoretical groundwork laid by Flynn's taxonomy, the research can thus establish the following possible combinations of the procedural sequences –

- CE^{SESE} – Single Element, Single Economy.
- CE^{MESE} – Multiple Elements, Single Economy.
- CE^{SEME} – Single Element, Multiple Economies.
- CE^{MEME} – Multiple Elements, Multiple Economies.

This adoption of Flynn's Taxonomies helps in avoiding the anomalies that could be related to the multiplicity of CE as a combinatorial product of ϵ and Ψ .

However, implementing **CA** as a computational framework to perform **CE** (as in all the procedural sequences) still has a major shortcoming that has already been highlighted in this thesis. Due to the undecidability of most of the Cellular Automata models described in this thesis so far the reliability of Cellular Automata as a computational framework to perform **CE** becomes very dysfunctional. This constitutional deficiency in any Cellular Automata model has been theoretically proven as a common phenomenon in the theory of Computation (as part of the computability theory) termed as the Halting Problem. For the **CE** to perform flawlessly, the halting problem needs to be overcome.

Although, it had been proved by Turing in 1937, that “a general algorithm to solve the halting problem for all possible program-input pairs cannot exist”(Turing, 1937)¹²³, Minsky stated in 1967 that, any finite-state machine, “if left completely to itself, will fall eventually into a perfectly periodic repetitive pattern and the duration of this repeating pattern cannot exceed the number of internal states of the machine” (Minsky, 1967)¹²⁴ thereby stating that theoretically, “the halting problem is decidable, because the machine has finite computational power” (with at least $2^{1,000,000}$ possible states, which is a cosmic equivalent of eons of galactic evolution).

Although not directly helpful in solving the halting problem, Minsky’s statement can help the research in theorizing that to make sure that a **CE** is halted, a concept of **decay** can be included in the rule sets of the **Ψ**. This condition of decay would act as a counterpart to the condition of growth in the **CE** meaning that the **Ψ** would become truly context aware, and not just be ruthlessly driving towards an equilibrium state. Moreover, with the introduction of the growth-decay dichotomy, the **€** and **Ψ** can perform structural coupling in a more anthropomorphic way. However, each taxon will have a specific concept of decay, which will be relevant to its **€** and **Ψ**.

¹²³ Turing, A. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42), pp. 230-265.

¹²⁴ Minsky, M. (1967). *Computation: Finite and Infinite Machines*. New Jersey: Prentice-Hall Inc.p. 334.

3.2.1 General Assumptions for all Procedural Sequences (CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME})

The previous explanation on CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME} , and the subsequent examples (as per 1.2.2) provide a clear illustration of all the individual taxon and their relationship within the entire taxonomy. However, as the concepts pertaining to the lexical semantics have already been established in the previous section (as in 3.1), the research requires a more ab initio approach in establishing the logical semantics as well. It also becomes essential to establish the logical semantics in this approach, because all the individual taxa as part of the procedural sequences will be implemented in fulfilling the operational objectives of this research (As explained in 1.2.2 and illustrated in fig. 1.10).

Before independently defining all the distinct taxa, some general assumptions can be considered for all the procedural sequences as a collective unit. These assumptions mainly involve the understanding of the notation systems required to perform the following operations as per the operational objectives –

- **Taxonomy** –
 - Although the independent taxa could have their distinguishing superscripts and subscripts notational systems as identified and defined for the specific taxa, all the procedural sequences must follow the classification system termed as **Folksonomy** (Peters, 2009)¹²⁵ which is a system that classifies content based on user tags, similar to citation indexing in Mendeley¹²⁶, and hashtags on Instagram.

¹²⁵ Peters, I. (2009). *Folksonomies. Indexing and Retrieval in Web 2.0*. Berlin: De Gruyter Saur, pp. 445.

¹²⁶ Mendeley. (2008). London: Elsevier.

- Folksonomies would be better suited for the entire procedural sequences, because the taxonomies generated in the operational objectives would be rich in graphical information, and would not require to be taxonomized based on hierarchies.
 - Moreover, the taxonomy can reflect the actual usage of the specific **CE** in terms of its implementation in the component of a built form, and can thus be used to create a robust data set.
 - And finally, because the tags for the taxa can be generated by the user, they can be simplified with user vocabulary or notational systems.
- **Evaluation –**
 - This part of the operational objectives exclusively performs the simulations section of the primary objectives, and thus involves the bulk of programming that would be performed in this research. Moreover, it also involves testing, bug-fixing, troubleshooting, and versioning of the **CE** for all the taxa in the aforementioned procedural sequences. This means, a lot of S.O.D. (Systems Oriented Design), pseudo-codes and computational strategies will be formulated in this section. With their many aberrations, the procedural systems could disarray the notation systems. Thus, to avoid any deviations and discrepancies in the communication strategies adopted for the evaluation, all the procedural sequences must follow the visualization system – UML version 2.5.1 (OMG, 2017)¹²⁷ which is the latest version of UML, “*the standard way to visualize the design of a computational system.*”

¹²⁷ OMG – Object Management Group (2017). *OMG® Unified Modelling Language® (OMG UML®) Version 2.5.1*. Milford, Massachusetts: OMG Group, pp. 754

- All the procedural sequences would be using a combination of the Structural UML Diagrams - Class Diagrams and Package Diagrams (within the system of UML 2.5.1), to define and identify the different variations of ϵ and ψ , their attributes (their physical properties), and their behaviors (their potential interactions with other ϵ and ψ).
 - Moreover, all the variations of all the procedural systems would be using a combination of the Behavioral UML Diagrams (within the system of UML 2.5.1), to illustrate the functioning of a specific CE (be it CE^{SESE} , CE^{MESE} , CE^{SEME} , or CE^{MEME}) while introducing the role of actors (as in the user who would have their role across the runtime of the CE to provoke the ϵ and ψ as per the design of the CE) and determining the role and activation of individual ϵ and ψ throughout the CE .
 - All the different above-mentioned UML diagrams will also be used to identify and define the different user-tests, bugs, troubleshooting, and versioning performed throughout the individual CE across the procedural sequences to perform the operational objectives.
 - However, individual taxa of the CE could have modification of their respective UML diagrams owing to the variations in the cardinality of their ϵ and ψ .
- **Prototyping –**
 - Although the prototyping objectives will be specific to the individual taxa within the procedural sequences (as in maybe one of the ψ of a specific CE demands the construction of a bridge, while another CE demands the construction of a tower), all the initial independent tests will be performed by means of FDM printing with PLA filament, and evaluated as per the predetermined requirements of the specific CE .

3.2.2 Single Element Single Economy Ecosystem (CE^{SESE})

As explained previously with an example of a system of cubes stacked on top of each other by considering the rules of structural stability (described in 1.2.2 and illustrated in fig. 1.6) the CE undergoes optimization for one and only one type of species (in the case of the example, a cube) by considering one and only one type of rule set (in the case of the example, structural stability). Thus, it is termed as the Single Element, Single Economy Ecosystem – CE^{SESE} .

Considering the above example (originally explained in 1.2.2 and illustrated in fig. 1.6), a standard UML based on the assumptions of 3.2.1 can be established for a CE^{SESE} as illustrated in the fig. 3.2 below.

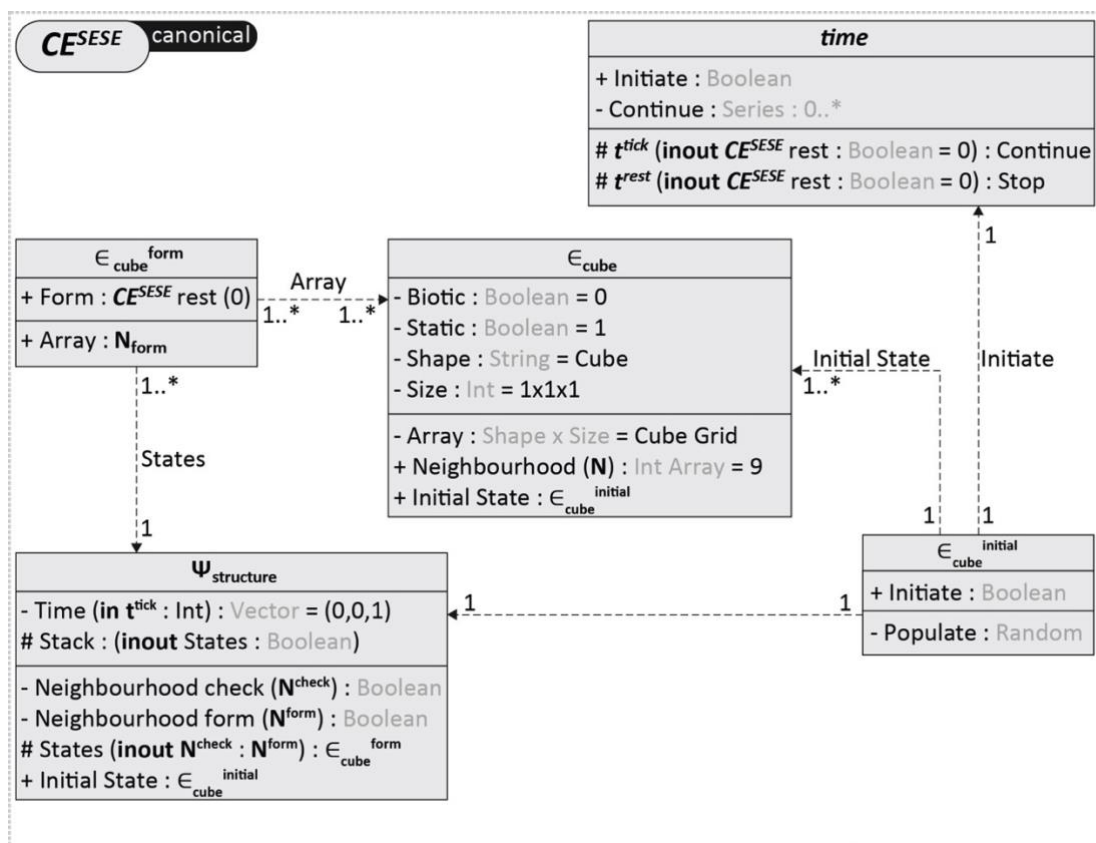


Figure 3.2 – Example of a standard UML for a CE^{SESE} considering an example case for the ϵ and Ψ . Illustration and graphics by Author (February 2018).

Considering the lexical semantics established in the previous sections (as per 3.1.1, 3.1.2, and 3.1.3) the following logical and empirical definitions could be established for a CE^{SESE} –

- The CE^{SESE} could be populated by one and only one species of ϵ . These ϵ could be represented by Platonic solids, Archimedean solids, point clouds, passive agents, active agents, cognitive agents, service equipment, structural members or fabrication material.
- The CE^{SESE} could be governed by one and only one equilibrium condition of Ψ . These Ψ could be represented by static structural stability, kinetic structural stability, functional adequacy, functional compatibility, contextual compatibility, climatic optimization, and fabrication constraints.
- All the procedural sequences and pseudo codes that would be formulated and theorized to achieve the operational objectives for the CE^{SESE} must follow the canonical CE^{SESE} UML diagram as shown in Fig. 3.2.
- The specific considerations for the ϵ and Ψ (pertaining to the geometrical properties of the cells, neighbourhoods and the specific considerations for the cell states), however, can be made while establishing the particular CE^{SESE} and must be named, defined and sufficiently illustrated.
- All the results achieved after performing the CE^{SESE} following all the procedural sequences must be taxonomized by following the canonical CE^{SESE} folksonomy system (as illustrated previously in 3.2.1).
- As the results will be demonstrated in this thesis in a graphical format, the visual programming script (conceptualized, designed and illustrated in Grasshopper3D) will be added in the annexure section of the thesis. Thus, it must clearly reflect the nomenclature as per the CE^{SESE} taxonomy system.

3.2.3 Multi Element Single Economy Ecosystem (CE^{MESE})

As explained previously with an example of a system of cubes and icosahedra stacked on top of each other by considering the rules of structural stability (described in 1.2.2 and illustrated in fig. 1.7) the CE undergoes optimization for multiple types of species (in the case of the example, a cube and an icosahedron) by considering one and only one type of rule set (in the case of the example, structural stability). Thus, it is termed as the Multiple Elements, Single Economy Ecosystem – CE^{MESE} .

Considering the above example (originally explained in 1.2.2 and illustrated in fig. 1.7), a standard UML based on the assumptions of 3.2.1 can be established for a CE^{MESE} as illustrated in the fig. 3.3 below.

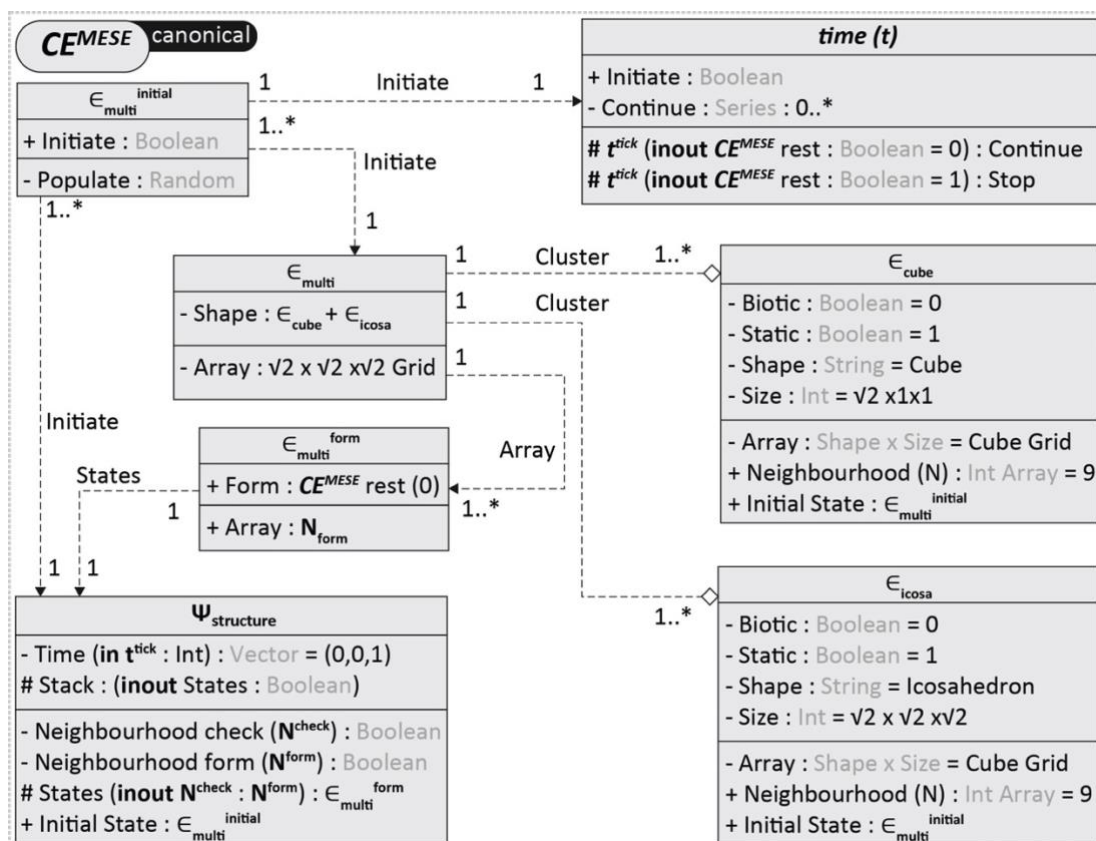


Figure 3.3 – Example of a standard UML for a CE^{MESE} considering an example case for the E and Ψ . Illustration and graphics by Author (February 2018).

Considering the lexical semantics established in the previous sections (as per 3.1.1, 3.1.2, and 3.1.3) the following logical and empirical definitions could be established for a CE^{MESE} –

- The CE^{MESE} could be populated by multiple species of ϵ . These ϵ could be represented by Platonic solids, Archimedean solids, point clouds, passive agents, active agents, cognitive agents, service equipment, structural members or fabrication material.
- The CE^{MESE} could be governed by one and only one equilibrium condition of Ψ . These Ψ could be represented by static structural stability, kinetic structural stability, functional adequacy, functional compatibility, contextual compatibility, climatic optimization, and fabrication constraints.
- All the procedural sequences and pseudo codes that would be formulated and theorized to achieve the operational objectives for the CE^{MESE} must follow the canonical CE^{MESE} UML diagram as shown in Fig. 3.3.
- The specific considerations for the ϵ and Ψ (pertaining to the geometrical properties of the cells, neighbourhoods and the specific considerations for the cell states), however, can be made while establishing the particular CE^{MESE} and must be named, defined and sufficiently illustrated.
- All the results achieved after performing the CE^{MESE} following all the procedural sequences must be taxonomized by following the canonical CE^{MESE} folksonomy system (as illustrated previously in 3.2.1).
- As the results will be demonstrated in this thesis in a graphical format, the visual programming script (conceptualized, designed and illustrated in Grasshopper3D) will be added in the annexure section of the thesis. Thus, it must clearly reflect the nomenclature as per the CE^{MESE} folksonomy system.

3.2.4 Single Element Multi Economy Ecosystem (CE^{SEME})

As explained previously with an example of a system of cubes stacked on top of each other by considering the rules of structural stability and buoyancy (described in 1.2.2 and illustrated in fig. 1.8) the CE undergoes optimization for one and only one type of species (in the case of the example, a cube) by considering multiple types of rule sets (in the case of the example, structural stability and buoyancy). Thus, it is termed as the Single Element, Multiple Economies Ecosystem – CE^{SEME} .

Considering the above example (originally explained in 1.2.2 and illustrated in fig. 1.8), a standard UML based on the assumptions of 3.2.1 can be established for a CE^{SEME} as illustrated in the fig. 3.4 below.

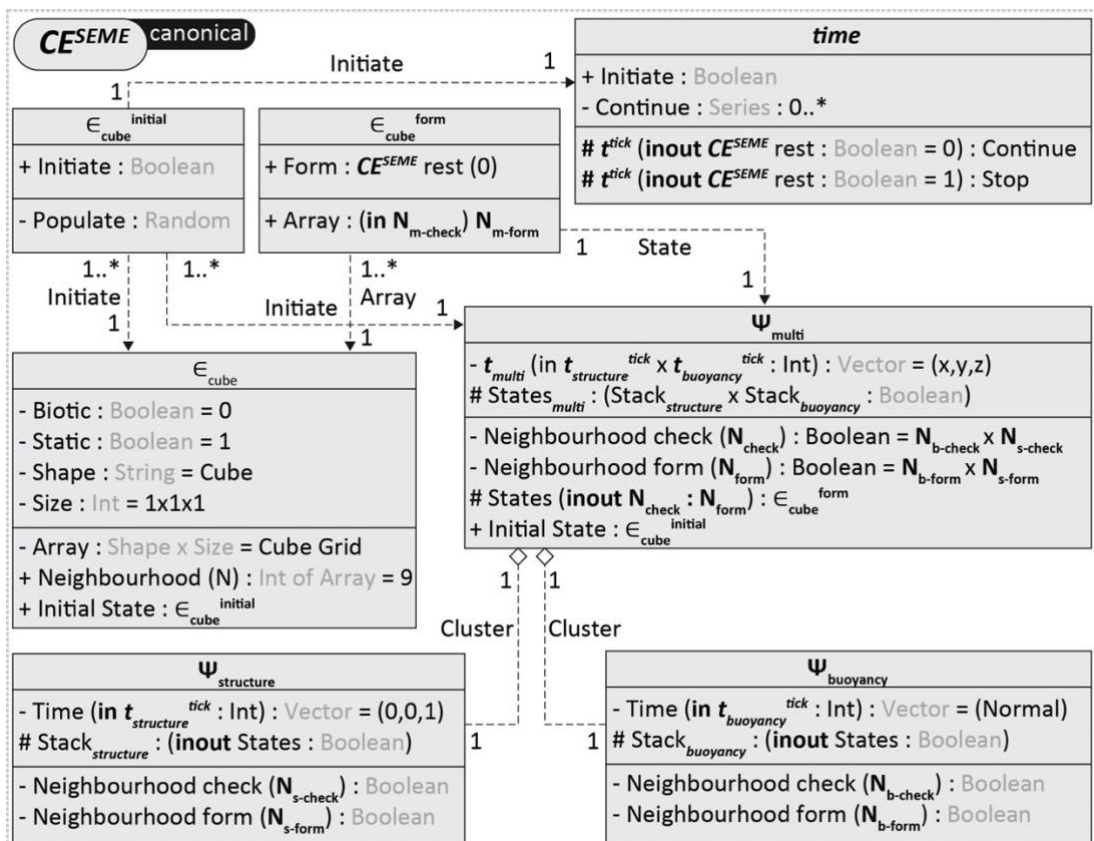


Figure 3.4 – Example of a standard UML for a CE^{SEME} considering an example case for the \in and Ψ . Illustration and graphics by Author (February 2018).

Considering the lexical semantics established in the previous sections (as per 3.1.1, 3.1.2, and 3.1.3) the following logical and empirical definitions could be established for a CE^{SEME} –

- The CE^{SEME} could be populated by one and only one species of ϵ . These ϵ could be represented by Platonic solids, Archimedean solids, point clouds, passive agents, active agents, cognitive agents, service equipment, structural members or fabrication material.
- The CE^{SEME} could be governed by multiple equilibrium conditions of Ψ . These Ψ could be represented by static structural stability, kinetic structural stability, functional adequacy, functional compatibility, contextual compatibility, climatic optimization, and fabrication constraints.
- All the procedural sequences and pseudo codes that would be formulated and theorized to achieve the operational objectives for the CE^{SEME} must follow the canonical CE^{SEME} UML diagram as shown in Fig. 3.4.
- The specific considerations for the ϵ and Ψ (pertaining to the geometrical properties of the cells, neighbourhoods and the specific considerations for the cell states), however, can be made while establishing the particular CE^{SEME} and must be named, defined and sufficiently illustrated.
- All the results achieved after performing the CE^{SEME} following all the procedural sequences must be taxonomized by following the canonical CE^{SEME} folksonomy system (as illustrated previously in 3.2.1).
- As the results will be demonstrated in this thesis in a graphical format, the visual programming script (conceptualized, designed and illustrated in Grasshopper3D) will be added in the annexure section of the thesis. Thus, it must clearly reflect the nomenclature as per the CE^{SEME} folksonomy system.

3.2.5 Multi Element Multi Economy Ecosystem (CE^{MEME})

As explained previously with an example of a system of cubes and cuboctahedra stacked on top of each other by considering the rules of structural stability and buoyancy (described in 1.2.2 and illustrated in fig. 1.9) the CE undergoes optimization for multiple types of species by considering multiple types of rule sets. Thus, it is termed as the Multiple Elements, Multiple Economies Ecosystem – CE^{MEME} .

Considering the above example, a standard UML based on the assumptions of 3.2.1 can be established for a CE^{MEME} as illustrated in the fig. 3.5 below.

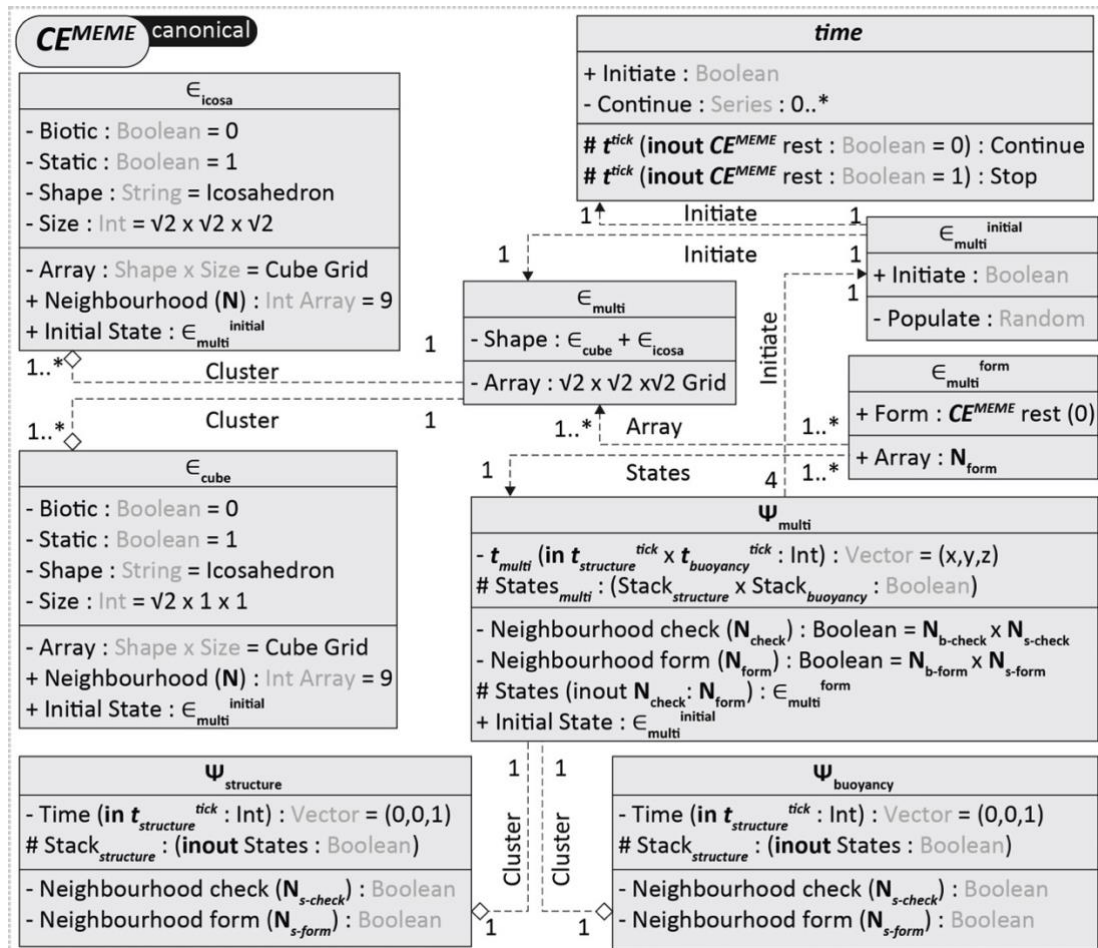


Figure 3.5 – Example of a standard UML for a CE^{MEME} considering an example case for the ϵ and ψ . Illustration and graphics by Author (February 2018).

Considering the lexical semantics established in the previous sections (as per 3.1.1, 3.1.2, and 3.1.3) the following logical and empirical definitions could be established for a CE^{MEME} –

- The CE^{MEME} could be populated by multiple species of ϵ . These ϵ could be represented by Platonic solids, Archimedean solids, point clouds, passive agents, active agents, cognitive agents, service equipment, structural members or fabrication material.
- The CE^{MEME} could be governed by multiple equilibrium conditions of Ψ . These Ψ could be represented by static structural stability, kinetic structural stability, functional adequacy, functional compatibility, contextual compatibility, climatic optimization, and fabrication constraints.
- All the procedural sequences and pseudo codes that would be formulated and theorized to achieve the operational objectives for the CE^{MEME} must follow the canonical CE^{MEME} UML diagram as shown in Fig. 3.5.
- The specific considerations for the ϵ and Ψ (pertaining to the geometrical properties of the cells, neighbourhoods and the specific considerations for the cell states), however, can be made while establishing the particular CE^{MEME} and must be named, defined and sufficiently illustrated.
- All the results achieved after performing the CE^{MEME} following all the procedural sequences must be taxonomized by following the canonical CE^{MEME} folksonomy system (as illustrated previously in 3.2.1).
- As the results will be demonstrated in this thesis in a graphical format, the visual programming script (conceptualized, designed and illustrated in Grasshopper3D) will be added in the annexure section of the thesis. Thus, it must clearly reflect the nomenclature as per the CE^{MEME} folksonomy system.

3.3 Semantic Syntax for the Procedural Sequences

The lexical semantics (which were thus obtained from the theoretical assumptions thoroughly established previously in 2|) and the logical semantics (which were based on the lexical semantics and comprehensively established throughout the sections 3.1 and 3.1) quite evidently form the theoretical foundations for this research while serving a database for the semantic syntax. This semantic syntax is required to undertake the procedural sequences that form the major bulk of the operational objectives and primary objectives for this research (as illustrated in 1.10) thus helping the research in empirically accomplishing the objectives laid down by the hypothesis.

As introduced (in 1.2.1), illustrated (in fig. 1.10), explained, and theorized (in 2| and 3|), this chapter thus concludes the literary objectives as explained –

- Literary objective I (***Lexical semantics***) – The terminologies and diction from the fields of Biology, Ecology, Genetics, Computing, Applied Mathematics, Applied Mechanics, Fabrication, Manufacturing and Economics relevant to the research as a literary aid were theorized to establish the structure of thought.
- Literary objective I (***Logical semantics***) – The terminologies and diction in the fields of Biology, Ecology, Genetics, Computing, Applied Mathematics, Applied Mechanics, Fabrication, Manufacturing and Economics relevant to the research as a literary aid were repurposed to establish the structure of a process.

The semantic syntax established by accomplishing the literary objectives will now aid in pursuing the operational objectives for all the procedural sequences in the further chapters (as per the structure explained in 1.3.2).

Apart from the semantic syntax that is derived from the theoretical assumptions made by pursuing the aforementioned Literary Objectives, the research is also based on the semantic syntax derived from the computational environment (in the form of software) that is being used to perform all the procedural sequences. Following is a list of all the digital tools that are employed in this research.

- Rhinoceros 3D¹²⁸ – Or Rhino 3D, which is a 3D modelling and CAD software, has been used as the primary computational environment to perform all the primary objectives for the procedural sequences (refer 1.2.1). Throughout the empirical advancements of this research, the latest available versions of Rhino 3D (as product names Rhino 5, Rhino 6, and Rhino 7) have been used depending on their respective release dates. All the files used to generate illustrations for this research have been visualized using Rhino 7, which is the latest version as of the publication of this thesis. Thus, the file formats are not compatible with the previous versions of the software.
- Grasshopper 3D¹²⁹ – Or Grasshopper, which is a visual programming language and environment, and (since Rhino 6) is packaged as a built-in plug-in for Rhino 3D, has been used as the programming interface to perform all the primary objectives for the procedural sequences. Grasshopper has not had any significant change throughout this research, apart from being directly integrated within the Rhino 3D interface. Thus, as of the publication of this thesis, all the Grasshopper definitions (which are those scripts that are written and performed in the Grasshopper environment, and saved as .gh file format) are and will be compatible across different versions.

Thus, the procedural sequences will be accomplished by implementing Rhino 7 and Grasshopper.

¹²⁸ Rhinoceros, version 1 (1998). Seattle: Robert McNeel & Associates.

¹²⁹ Grasshopper 3D (2007). Seattle: David Rutten, Robert McNeel & Associates.

Methodology

4| On the procedural sequences for the Computational Ecosystems

4.1 Primary objectives through Procedural sequences

An explicit roadmap regarding the objectives (the primary and secondary, as explained in 1.2.1) was clearly laid out in Chapter 1 (as illustrated in fig. 1.10 in 1.2.3) and the previous chapters (2| and 3|) have quite distinctively defined all the required theoretical foundation for the research which hypothesizes –

What if the **built form** was constructed, monitored and governed by an
autonomous, unbiased **algorithm**?

What if this **algorithm** was dynamically constructed, monitored and governed by
the **built form**?

And, theorizes to create, taxonomize, and prototype a dynamic, reciprocal, symbiotic feedback loop in the form of Computational Ecosystems (**CE**) built on the computational framework of Cellular Automata (**CA**). As theorized and defined in the previous chapters, the procedural sequences which form the basis of the operational objectives of the research have been performed in the form of the four taxa - **CE^{SESE}**, **CE^{MESE}**, **CE^{SEME}**, and **CE^{MEME}**. All these four taxa have undergone the three stages of operational objectives – **Taxonomies**, **Evaluation**, and **Fabrication**. As these operational objectives have had overlapping contributions to the primary objectives, all the taxa have been individually explained in this thesis by following a structure derived from the primary objectives – **Case studies**, **Simulations**, and **Prototyping** in this chapter. Moreover, as this entire chapter mostly focuses on all the processes undertaken to theorize, program, prototype, and iterate all the 4 independent taxa, the significance of the results of these taxa are discussed and elaborated upon in chapter 5.

Because of the impetus put on the actual performance of all the theoretical inputs and implementation of all the semantics described thus far, this chapter, which focuses on the methodology, has a lot of graphical information that helps in illustrating the different morphologies acquired by performing a wide range of combinations and variations of ϵ and ψ , and their initial states. All these morphologies have been programmed, generated and visualized by using Rhino 7 and Grasshopper (as elaborated in 3.3). However, to perform the distinguished computational operations that are required by the assumptions made for the specific ϵ and ψ , have been performed by implementing additional plugins. These plugins are available on major opensource scripting platforms (like GitHub, grasshopper3D and food4rhino) and can be downloaded and installed with absolute ease. The specific plugins have been mentioned while performing a specific attribute attached to the ϵ and ψ in question in the specific taxon.

Some illustrations have also been created by implementing V-Ray¹³⁰ for Rhino (which is a biased CGI rendering software built for geometry generated in Rhino). The rendering software was employed to simulate the visual properties of a wide range of materials that are used in the AEC industry. Because this research is not directly related to material engineering and purely focuses on computational design and digital fabrication, the use of the rendering engine as opposed to photography of actual material is justified. Thus, a lot of graphic-heavy imagery with near-photo-realistic quality becomes empirical evidence for most of the methodological framework in the form of the operational objectives.

However, to showcase the grasshopper definitions in a visual format, while not overpowering the visual imagery (which provides the empirical evidence with relative ease), the research prefers to add the definitions to the annexure section of this thesis.

¹³⁰ V-Ray for Rhino (1997). Sofia, Bulgaria: Chaos Group.

(As elaborated in 3.2) The pseudo-codes and computational strategies for all the procedural sequences have been based upon the canonical UML Class Diagrams for the respective taxon (as illustrated in fig. 3.2, fig. 3.2, fig. 3.4, and fig. 3.5) which were set up considering the examples in chapter 1| On the relevance of Computational Ecosystems (as in fig. 1.6, fig. 1.7, fig. 1.8, and fig. 1.9). However, a computational strategy needs a Behavioral UML Diagram to understand the role, interaction and runtime of all its components (in this case the ϵ and ψ) and cannot be defined by establishing a Structural UML Diagram that merely identifies and defines the attributes and operations of a class (again, in this case the ϵ and ψ).

On the contrary, the use of a Behavioral UML diagram cannot be used as a canonical version for all the **CE** that can be part of a specific procedural sequence. For example, it would be counter-intuitive to create a canonical **CE^{SESE}** UML with the help of a Behavioral UML diagram that could possibly address all the various properties that could be attributed to ϵ and ψ , while defining their roles, interaction and runtime. Therefore, it is clever to establish and illustrate the individual Behavioral UML diagrams for the specific taxon at the time of their introduction in this thesis. These specific and distinctive Behavioral UML Diagrams would be a part of the prototyping section of the **CE** taxon in question and will be categorized under the same criteria as the procedural sequences (as in the form of the four already established taxa).

All the subsequent Behavioral UML Diagrams (just like all the canonical UML Class Diagrams) have been created using a web-based tool - Lucidchart¹³¹ which is widely used by companies to create and share UML Diagrams in the software industry.

Thus, after establishing the variations, additions, extensions and specific modifications to the semantics, the individual taxa of the procedural sequences will be elaborated in the next pages through the primary objectives while fulfilling the operational objectives (as per 1.2.1).

¹³¹ Lucidchart (2008). Utah: Lucid Software Inc.

4.2 Single Element Single Economy Ecosystem (CE^{SESE})

The conceptual framework of a CE^{SESE} has already been sufficiently illustrated in the previous chapters (introduced in 1.2.2, illustrated in fig. 1.6, defined in 3.2.2). Moreover, a Structural UML Diagram establishing the attributes and operations of a canonical version, the ϵ and ψ has also been illustrated (in fig. 3.2 as part of sub section 3.2.2). However, to actually construct, taxonomize and prototype a procedural sequence, the research would have to consider specific tangible parameters which could be derived from real-life examples and constraints.

As the first CE to be established under the theoretical framework, it was prudent to start with an elementary form (not elementary in the sense as Wolfram's Elementary Cellular Automata) of CE which could be based on one of the first examples of a CE and the consequent conceptual frameworks introduced in this thesis. In short, the CE illustrated in 1.2.2 to introduce the concept of an $SESE$ (hereafter termed as the example scenario) would be an ideal place to start with the procedural sequences for a CE^{SESE} . This decision of starting with the example case would also make it easier to refer to the canonical Structural UML Diagram and base the pseudo-code on it, albeit with some modifications to better suit the exact parameters of the CE^{SESE} .

Thus, before specifically defining each parameter, we can roughly state the following overall components of the CE^{SESE} –

- ϵ – As illustrated in the example scenario, the ϵ would be represented by hexahedrons or cubes, with unit dimensions.
- ψ – As illustrated in the example scenario, the ψ would be represented by structural stability in the form of axial loads.

Regardless of the above generalization, the CE^{SESE} would require to fulfill all the primary objectives while pursuing the operational objectives for this taxon.

4.2.1 Case Studies

As a computational construct, it is quite straightforward and obvious to theorize an ecosystem that is inhabited by one and only one kind of species and those species are constructed, monitored, and governed by one and only one rule set. However, it is very difficult to find a real-life ecosystem that reflects similar constituent parts. Most ecosystems have complexities and multiplicities in either or both of the types of species or the types of purposes that bring them together (that is in terms of positive or negative intraspecific or interspecific cooperation as elaborated in 2.1.3).

On the contrary, owing to the relative ease and simplicity of computation, most of the bio-based algorithms that have been mentioned in this thesis (such as ACO, BFA, FSA, ABC as mentioned in 2.1.4 in fig. 2.1) are experimented and theorized with a single type of species constructed, monitored, and governed by a single type of rule set. Although the computation is not precise or realistic, the relative simplicity of the computational methodology is quite instrumental in generating required research observations and results. This very advantage has been one of the key factors in the significance of this taxon (CE^{SESE}) in providing straightforward results by means of performing relatively simplified methodologies.

This means, that the assumptions for the ϵ and Ψ for the CE^{SESE} must conform to a system that follows the aforementioned criteria (as per the example scenario) but can also have a considerable advantage if it was based on a real-life ecosystem that reflects similar constituent parts. However, the hexahedrons or cubes considered as ϵ , are lifeless blocks of an arbitrary dimension and can only be classified as $\epsilon_{\text{abiotic}}$, which makes it very laborious to analogically characterize it with a real-life ecosystem. Thus, the following addition must be made to the assumptions –

- ϵ – Hexahedrons or cubes, with unit dimensions, which can be programmed as sentient elements, that are aware of their existence, and thus can be considered as ϵ_{biotic} .

Eciton hamatum, which is a species of army ants found in parts of Mesoamerica and South America, are known for their large swarms which raid forests to forage for food sources. While in their pursuits, the swarms cross very many natural hurdles by forming collective assemblages out of their own bodies to perform a variety of functions that benefit the entire colony. Fig. 4.1 illustrates army ants building and using a living bridge to span gaps in the colony's foraging trail.



Figure 4.1 – Ant Bridge (Panama). Original Image from the fieldwork on Army ants (*Eciton hamatum*) induced to form a very large bridge over a wide gap by Chris R. Reid. Source: <https://chrisreid.wordpress.com/fieldwork/>

In field studies conducted on ant bridges, it has been found that, “*the ants continuously modify their bridges, such that these structures lengthen, widen, and change position in response to traffic levels and environmental geometry.*” It has also been observed that “*the bridge construction is influenced by a cost-benefit trade-off at the colony level, where the benefit of increased foraging trail efficiency is balanced by the cost of removing workers from the foraging pool to form the structure*” (Reid, et al).¹³²

¹³² Reid, C. R., Lutz, M. J., Powell, S., Kao, A. B., Couzin, I. D., and Garnier, S. (2015). Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. In: *Proceedings of the National Academy of Sciences of the United States of America*. [online] Washington DC: PNAS, p 6. . Available at: <https://www.pnas.org/content/pnas/early/2015/11/18/1512241112.full.pdf>

If a rudimentary computational model was to be made for the simulation of the bridging behavior of the aforementioned army ants it would translate to a **CE** as has been already defined as **Hybrid Bio Plausible Bio-inspired Stochastic Optimization Algorithm**. Say, this proposed **CE**, following a basic nomenclature system, could then be termed as **CE_{eciton-bridge}** (as the CE reflects bridging properties of the eciton hamatum species), and would then consist of the following constituent parts:

- **€** – The army ants. Acting as biotic, ambulatory agents that are self-aware of their physical properties such as their weight, weight-carrying capacity, movement speed, and gripping abilities. Thus, **€_{eciton}**.
- **Ψ** – The bridging ability. Forming collective assemblages by means of connection techniques, span-depth ratios for optimum bridge structures, and load calculations (considering the live loads and dead loads). Thus, **Ψ_{bridge}**.

Although, a more realistic **CE_{eciton-bridge}** would consist of a few more **€** (such as the abiotic food sources, abiotic food fragments, and the abiotic gaps in the foraging trail) and **Ψ** (such as shortest foraging trails and cost-benefit analysis for the colony dynamics) to precisely simulate the foraging behavior of the army ants and to understand the relations between the cost-benefit trade-off, however, to simulate and analyze the living bridge structures as collective assemblages created by the army ants, the above components could be considered sufficient. A **CE_{eciton-bridge}** with the abovementioned **€** and **Ψ** would then be analogous to a **CE^{SESE}** with a single species of **€** and a single ruleset of **Ψ**. The proposed **CE^{SESE}** can reflect a computational adaptation of the **CE_{eciton-bridge}**, albeit the significant divergence in the **€_{eciton}** and the **€_{biotic}** for the proposed **CE^{SESE}** (where the former is an ambulatory agent, and the latter is a static cube conceptualized to be sentient and context-aware). Therefore, a **CE_{eciton-bridge}** can be considered as a real-life case study to establish the **CE^{SESE}**.

Thus, the similarities between a **CE^{SESE}** and a **CE_{eciton-bridge}** as compared to the example scenario can be illustrated as shown in fig. 4.2. below.


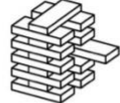



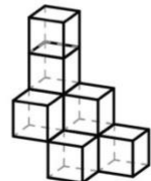
SESE		
The Example Scenario		
	Element - Cube	Economy - Structural Stability
CE_{eciton-bridge}		
	Element - ϵ_{eciton}	Economy - Ψ_{bridge}
CE^{SESE}		
	Element - ϵ_{cube}	Economy - Ψ_{tower}

Figure 4.2 – Comparing the example scenario, a **CE^{SESE}** and a **CE_{eciton-bridge}**. Illustration and graphics by Author (May 2018).

Considering the aforementioned assumption, the proposed **CE^{SESE}** can now be treated as a **Hybrid Bio Plausible Bio-inspired Stochastic Optimization Algorithm**, that seeks to simulate a computational adaptation of the collective assemblages achieved by eciton hamatum or army ants. This assumption can be employed as a direct analogy to develop the next stages of this taxon. However, the ϵ and Ψ must be clearly established as followed –

- ϵ – Or ϵ_{cube} (derived from already established ϵ_{biotic} as a cube), can be considered as a computational substitute for the ϵ_{eciton} established for the construction of **CE_{eciton-bridge}**. However, it needs to be considered with the following additional properties:

- The $\mathbf{\epsilon}_{\text{cube}}$ should be considered as a static agent that can be spawned or culled by the $\mathbf{CE}^{\text{SESE}}$ at the expense of the equilibrium state which will be determined by the $\mathbf{\Psi}$.
- The $\mathbf{\epsilon}_{\text{cube}}$ should be made to be self-aware of itself, in the sense that it should be able to identify and quantify its position, orientation, weight, and bounds.
- The $\mathbf{\epsilon}_{\text{cube}}$ should also be made to be self-aware of its surroundings and their physical properties, such as cardinality of neighbours, the array of neighbours, and distance from the initial plane.
- $\mathbf{\Psi}$ – can be considered as the computational substitute for the $\mathbf{\Psi}_{\text{bridge}}$ established for the $\mathbf{CE}_{\text{eciton-bridge}}$. However, it needs to be considered with the following additional properties:
 - Instead of a bridging logic which has major dependencies on the span of the gap (to be bridged), a tower logic which has no such constraints apart from the height can be considered as the primary attribute of the $\mathbf{\Psi}$. Thus, $\mathbf{\Psi}_{\text{tower}}$ can be introduced.
 - $\mathbf{\Psi}_{\text{tower}}$ should be considered as a simple vertical stacking logic serving as a computational substitute to the collective assemblages proposed for the $\mathbf{CE}_{\text{eciton-bridge}}$.
 - $\mathbf{\Psi}_{\text{tower}}$ should be designed primarily as a two-state rationale that depends on if the cube is spawned or culled while conforming to the understanding of vertical stacking.
 - $\mathbf{\Psi}_{\text{tower}}$ should seek an equilibrium state that spawns $\mathbf{\epsilon}_{\text{cube}}$ when the tower is under-structured, and culls $\mathbf{\epsilon}_{\text{cube}}$ when the tower is over-structured, thus maintaining a reciprocal coupling.

After establishing and defining the $\mathbf{\epsilon}$ and $\mathbf{\Psi}$ as $\mathbf{\epsilon}_{\text{cube}}$ and $\mathbf{\Psi}_{\text{tower}}$, the $\mathbf{CE}^{\text{SESE}}$ could be specifically defined under the computational modelling guidelines of \mathbf{CA} (as elaborated in 3 |) as followed –

- The $\mathbf{CE}^{\text{SESE}}$ which owing to its specific $\mathbf{\epsilon}$ and $\mathbf{\Psi}$ combination can be termed as $\mathbf{CE}_{\text{cube-tower}}$ will be defined as a functioning ecosystem constituting of sentient, context aware $\mathbf{\epsilon}$ entities that interact with each other considering the rulesets assigned by $\mathbf{\Psi}_{\text{tower}}$ to spawn or cull $\mathbf{\epsilon}_{\text{cube}}$ until the runtime of the $\mathbf{CE}_{\text{cube-tower}}$.
- The $\mathbf{CE}_{\text{cube-tower}}$ will perform solitarily without any additional internal components or partial runtimes. Moreover, there won't be context aware \mathbf{CE} that would have to be added externally. Thus, in this taxon, one and only one \mathbf{CE} that is $\mathbf{CE}_{\text{cube-tower}}$ would be performed for its entire runtime.
- Although derived from the intricate bio-inspired behavior of army ants creating and maintaining living bridges for foraging trails, the $\mathbf{CE}_{\text{cube-tower}}$ is basically an elementary vertical cube stacking algorithm. Therefore, the rule sets for $\mathbf{\Psi}_{\text{tower}}$ can be adapted from the Conway model of Cellular Automata (as elaborated in 2.3.2).
- The $\mathbf{\Psi}_{\text{tower}}$ would differentiate from the Conway model of Cellular Automata in the parameter of time. In the Conway model, with increased intervals of time (\mathbf{t}_{tick}) we find new cells being added to or removed from the 2D grid. However, in the case of the $\mathbf{CE}_{\text{cube-tower}}$, the new cells (in the form of $\mathbf{\epsilon}_{\text{cube}}$) will be spawned or culled on the upper levels in the 3D grid.

Considering all the above assumptions, definitions, illustrations and examples for the $\mathbf{CE}_{\text{cube-tower}}$, the simulations for this taxon can now be performed. However, some information still needs to be considered to establish and illustrate the empirical derivations of all the case studies. For example, the $\mathbf{\epsilon}_{\text{cube}}$ and $\mathbf{\Psi}_{\text{tower}}$ still need to be dimensionally, and geometrically defined to consider them for computational use.

4.2.2 Simulations

Thus, to begin defining the ϵ_{cube} and Ψ_{tower} to be employable in a computational context, it is important to reassess and repurpose the canonical CE^{SESE} Structural UML Diagram (as per fig. 3.2) for the $CE_{\text{cube-tower}}$. Fig. 4.3, as shown below, illustrates the modifications to the canonical version.

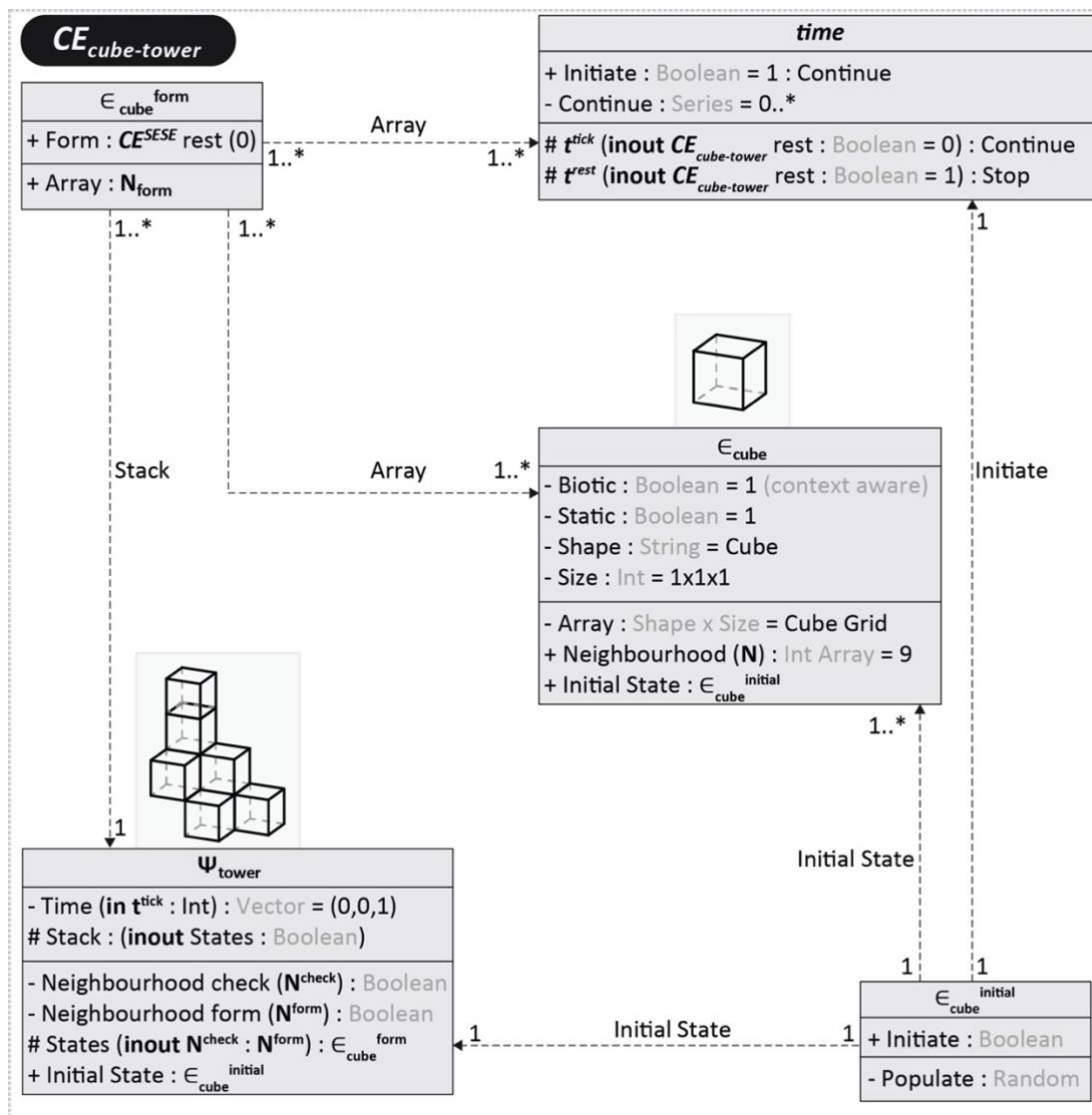


Figure 4.3 – Modifications made to the canonical CE^{SESE} UML Diagram to accommodate the parameters required for establishing the $CE_{\text{cube-tower}}$. Illustration and graphics by Author (May 2018).

Elaborating on the terminologies introduced in fig. 4.3, the attributes and operations of the classes can be defined as followed –

- t^{tick} – The periodic increment of time, which ensures that the $CE_{cube-tower}$ runs.
- t^{rest} – If the condition $CE_{cube-tower}$ rest is fulfilled, the t^{rest} will be activated. This means that the time increment will stop and the $CE_{cube-tower}$ will
- $CE_{cube-tower}$ rest – Is the situation where an E_{cube} array for the last 3 t^{tick} intervals is equal or repetitive.
- E_{cube} array – Is the collective assemblage of E_{cube} for a certain t^{tick} interval.
- E_{cube}^{form} – Is the formation of an E_{cube} array depending on the binary input from N^{form} .
- N^{form} – Is the binary input for the cells to be either spawned (input = 1) or culled (input = 0) depending on the N^{check} conditional.
- N^{check} – Is the binary output from the cells of the neighbourhood (N) to check if the cells in the N of the previous t^{tick} interval are either spawned (input = 1) or culled (input = 0).
- N – The neighbourhood of the cell in question (as in every E_{cube}) depending on how many neighbouring E_{cube} from the previous t^{tick} interval are to be considered to determine the N^{check} and the N^{form} routines that determine **Stack** operation.

Thus (as seen above in reverse chronology), determining the neighbourhood (N) becomes the most important step before considering to explain the other concepts under the inheritance field. Moreover, the determining of the initial states and the **initiate** conditions also require an understanding of the neighbourhood (N).

However, before establishing the **N**, the computational environment and interface need to be established. All the modelling has been performed in Rhino 7 (as explained in 3.3), the programming has been performed in Grasshopper for Rhino 7 (as explained in 3.3) and visualizations have been performed in V-Ray for Rhino (as explained in 4.1). Thus, all the assumptions that are mentioned here after, take place not only in the conceptual hypothetical realm of the structure of thought but also in terms of the software interfaces mentioned above.

The computational universe of a **CE_{cube-tower}** in terms of Euclidian geometry is an infinite, 3D orthogonal grid of cubes, and can be realized by the 1x1x1 dimensions (in the X, Y, and Z axes). However, the infinity of the grid would be treated rather differently in this case, because the motive of a **CE_{cube-tower}** is to construct a tower, and it requires a fixed, immobile ground plane. Considering this ground plane would be the coordinate XY-plane located at the origin (0,0,0), the infinity of the x and y axes can stretch in the +X, -X and +Y, -Y directions respectively, however, the infinity of the Z-axis can only stretch in the +Z direction. Thus, the Array operation in the **E_{cube}** class can be defined as an infinite 3D cube array conformed to a (0,0) base plane. Thus forming a **hypothetical revised infinite 3D Square grid**.

As the grid on all the axes is strictly prescribed to the dimensions 1x1x1, when cubes are stacked, it will appear as if the cubes are stacked on top of each other on the Z-axis. This would be particularly helpful in defining the **Stack** operation which is controlled by the **N^{check}** and the **N^{form}** routines, which correspond to **E_{cube}^{form}** that reflects if the **E_{cube}** is spawned or culled in the specific **t^{tick}** interval. The spawning or culling also corresponds to the terminology which determines if **E_{cube}** exists or does not exist, respectively. However, before determining the State conditions, which form the computational core of **Ψ_{tower}** (and thus consequently of the **CE_{cube-tower}**), the **N** for the **E_{cube} Array** defined above needs to be explained and illustrated. The concept of the **N** would also be instrumental in illustrating all the **N^{check}** and the **N^{form}** routines, as it would serve as a continuation to the same graphical vocabulary, with the same examples.

As the concept of neighbourhood (\mathbf{N}) depends directly on the morphology of the \mathbf{E}_{cube} , fig. 4.4 illustrates a solitary $\mathbf{E}_{\text{cube-0}}$, surrounded by its \mathbf{N} , say \mathbf{N}_0 .

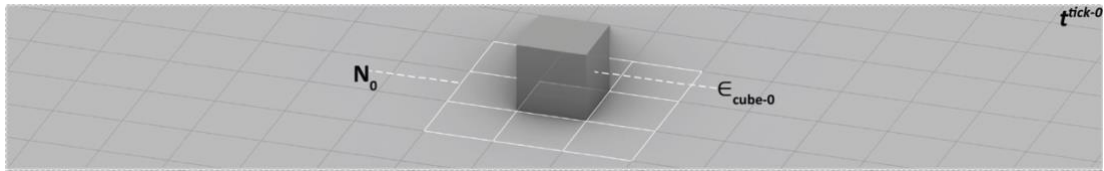


Figure 4.4 – A solitary $\mathbf{E}_{\text{cube-0}}$ surrounded by its \mathbf{N}_0 . Model and graphics by Author (May 2018).

Following the UML $\mathbf{CE}_{\text{cube-tower}}$ class diagram in fig. 4.3, the $\mathbf{N}^{\text{check}}$ for $t^{\text{tick-0}}$ would determine the \mathbf{N}^{form} for $t^{\text{tick-1}}$ and place a new $\mathbf{E}_{\text{cube-1}}$ on top of the existing $\mathbf{E}_{\text{cube-0}}$ as shown in fig. 4.5 below.

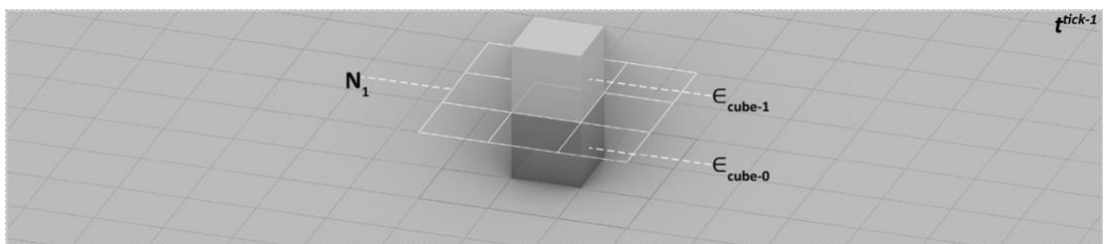


Figure 4.5 – \mathbf{E}_{cube} and their \mathbf{N} across consecutive t^{tick} . Model and graphics by Author (May 2018).

Thus, $\mathbf{E}_{\text{cube-1}}$ would now have its own \mathbf{N}_1 , and the process would go on until t^{rest} is achieved at $t^{\text{tick-2}}$, as illustrated in fig. 4.6 below.

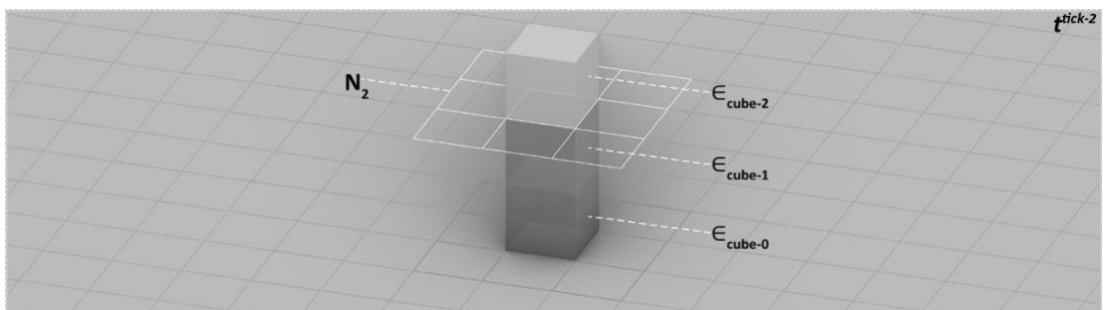


Figure 4.6 – \mathbf{E}_{cube} and their \mathbf{N} until t^{rest} is achieved. Model and graphics by Author (May 2018).

Thus, the $\mathbf{E}_{\text{cube-}n+1}$ for every corresponding $t^{\text{tick-}n+1}$ condition, would depend on the position of the $\mathbf{E}_{\text{cube-}n}$ at the specific $t^{\text{tick-}n}$ condition. However, to understand and determine the $\mathbf{N}^{\text{check}}$ and the \mathbf{N}^{form} routines to establish which \mathbf{E}_{cube} is spawned (or exists) and which is culled (or does not exist), it is essential to draw inspiration from Cellular Automata models mentioned in this thesis. As mentioned in 4.2.1, the construction of the $\mathbf{CE}_{\text{cube-tower}}$ would be following the **Conway Model**. (As already established in 2.3.2) The Conway model is a two-state **CA**, and every cell interacts with its eight neighbours, conforming to the Moore neighbourhood (\mathbf{N}_M). At each step, in time, the following transitions occur (Gardner, 1970)¹³³:

- *Any live cell with two or three live neighbours (in its \mathbf{N}_M) survives.*
- *Any dead cell with three live neighbours (in its \mathbf{N}_M) becomes alive.*
- *All other live cells die in the next generation.*
- *Similarly, all other dead cells stay dead.*

However, (as per 4.2.1) the state conditions of the Conway model cannot be adopted in the $\mathbf{CE}_{\text{cube-tower}}$, as the latter has a major deviation in terms of the computational environment. As Conway's game of life is based on an infinite 2d square grid, and the $\mathbf{CE}_{\text{cube-tower}}$ conforms to the aforementioned **hypothetical revised infinite 3D Square grid**, the transitions mentioned above (in the Conway model) that occur in time, will have to occur in the +Z axis in the $\mathbf{CE}_{\text{cube-tower}}$. Meaning, the above rules that help the *game of life* to be visualized as moving, will help the **CE** to be visualized as growing (in the +Z axis). But first, the state conditions will have to be modified and adapted to the **N**, while every t^{tick} condition would perform the \mathbf{E}_{cube} array until the t^{rest} condition is achieved.

¹³³ Gardner, M. (1970). MATHEMATICAL GAMES - The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*. [online] 223(4), pp. 120-123. Available at: <https://web.stanford.edu/class/sts145/Library/life.pdf> [Accessed 31 May 2020].

Thus, the modified state conditions for the $CE_{cube-tower}$ drawn from the Conway model can be stated as followed –

- Every $E_{cube-n+1}$ at the $t^{tick-n+1}$ interval interacts with its N of the E_{cube-n} at the t^{tick-n} interval, thus performing N^{check} . For the $t^{tick-n+1}$ interval, it then performs the N^{form} routine based on the following conditions –
 - Any existing (spawned) E_{cube-n} with two or three existing E_{cube-n} in its N continues to exist and is not culled at the $t^{tick-n+1}$ interval, as if it was being perfectly supported by its counterparts on the floor below.
 - Any existing (spawned) E_{cube-n} with other than two or three existing E_{cube-n} in its N stops existing and is culled from the $t^{tick-n+1}$ interval, as if it was being over-supported by its counterparts on the floor below.
 - Any non-existing (culled) E_{cube-n} with three existing E_{cube-n} in its N exists and is spawned for the $t^{tick-n+1}$ interval, as if it was being perfectly supported by its counterparts on the floor below.
- These rules, which compare the behavior of the automaton to real life, can be summarised into the following:
 - Any existing E_{cube-n} at the t^{tick-n} interval with two or three existing E_{cube-n} at the t^{tick-n} interval in its N continues to exist at the $t^{tick-n+1}$ interval.
 - Any non-existing E_{cube-n} at the t^{tick-n} interval with three existing E_{cube-n} at the t^{tick-n} interval in its N is spawned at the $t^{tick-n+1}$ interval.
 - All other existing E_{cube-n} at the t^{tick-n} interval stop existing at the $t^{tick-n+1}$ interval. Similarly, all other non-existing E_{cube-n} at the t^{tick-n} interval remain nonexistent at the $t^{tick-n+1}$ interval.

Thus, considering the aforementioned state conditions, the condition for the existing state of the \mathbf{E}_{cube} array, which can be either continued existence (as in not culled) or newly existing (as in spawned) can be illustrated as shown in fig. 4.7 below.

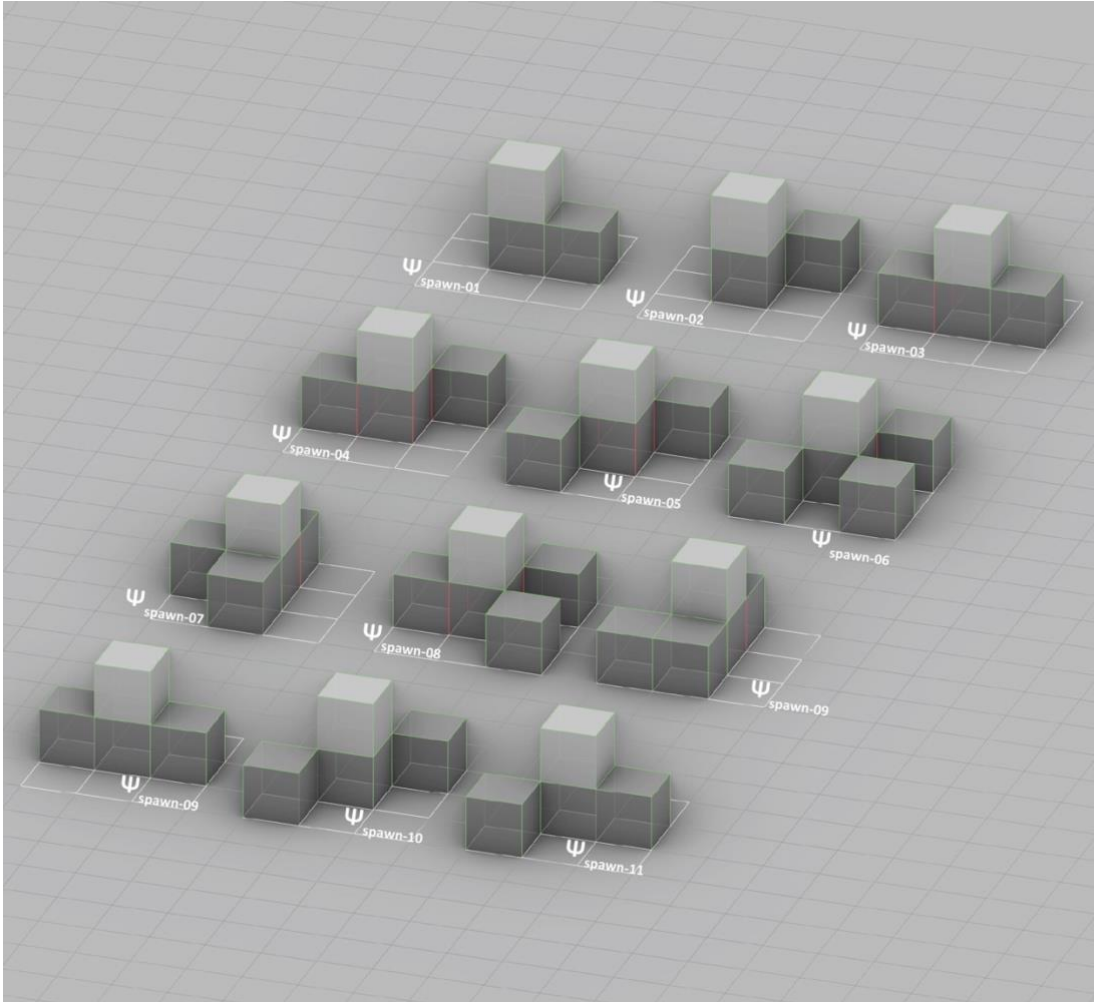


Figure 4.7 – $\mathbf{E}_{\text{cube}-n}$ array and their corresponding $\mathbf{E}_{\text{cube}-n+1}$ array considering the state conditions for the existing states of \mathbf{E}_{cube} array. Model and graphics by Author (May 2018).

The $\mathbf{E}_{\text{cube}-n+1}$ at the $t^{\text{tick}-n+1}$ interval is thus denoted by ■ (R,G,B – 204,204,204) colored cube, and the corresponding $\mathbf{E}_{\text{cube}-n}$ at the $t^{\text{tick}-n}$ interval is denoted by ■ colored cube (R,G,B – 129,129,129). Also, the existing cubes are denoted by ■ (R,G,B – 57,188,40) colored cube, and the non-existing cubes are denoted by ■ (R,G,B – 255,21,21) colored cubes.

And, considering the aforementioned state conditions, the condition for the non-existing state of the $\mathbf{\epsilon}_{\text{cube}}$ array, which can be either discontinued existence (as in culled) or not newly existing (as in not spawned) can be illustrated as in fig. 4.8 below.

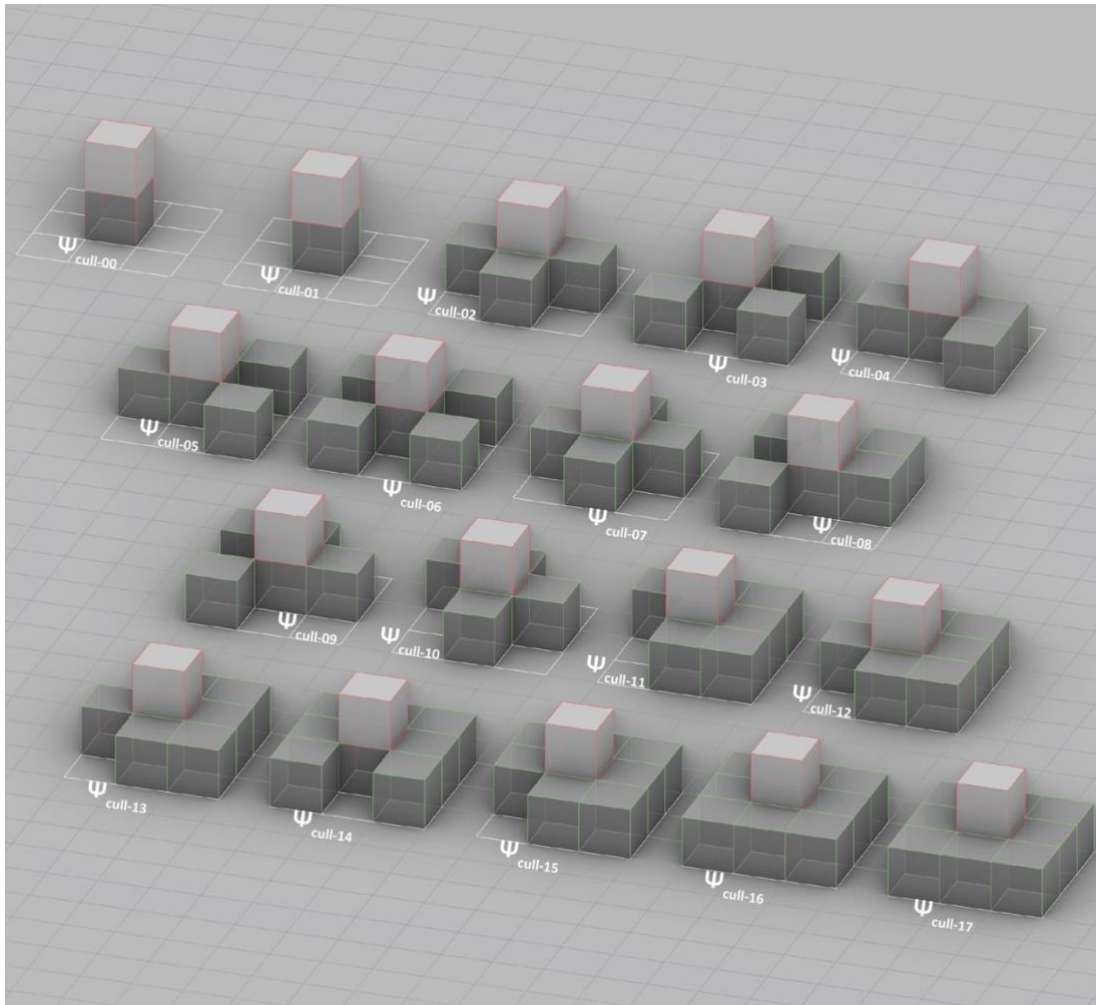


Figure 4.8 – $\mathbf{\epsilon}_{\text{cube-n}}$ array and their corresponding $\mathbf{\epsilon}_{\text{cube-n+1}}$ array considering the state conditions for the non-existing states of $\mathbf{\epsilon}_{\text{cube}}$ array. Model and graphics by Author (May 2018).

The $\mathbf{\epsilon}_{\text{cube-n+1}}$ at the $t^{\text{tick-n+1}}$ interval is thus denoted by ■ (R,G,B – 204,204,204) colored cube, and the corresponding $\mathbf{\epsilon}_{\text{cube-n}}$ at the $t^{\text{tick-n}}$ interval is denoted by ■ colored cube (R,G,B – 129,129,129). Also, the existing cubes are denoted by ■ (R,G,B – 57,188,40) colored cube, and the non-existing cubes are denoted by ■ (R,G,B – 255,21,21) colored cubes.

4.2.3 Prototyping

The state conditions in terms of the $\mathbf{N}^{\text{check}}$ and the \mathbf{N}^{form} routines of the Ψ_{tower} that determine the **Stack** operation for the \mathbf{E}_{cube} array have been sufficiently demonstrated and illustrated in terms of the Ψ_{spawn} and Ψ_{cull} rules (as shown in fig. 4.7 and fig. 4.8). The illustrations show all possible conditions for the Ψ_{spawn} and Ψ_{cull} rules, however, the computational logic for stacking is quite simple. If the cardinality of the \mathbf{N} is equal to 2 or 3, the \mathbf{E}_{cube} survives (i.e. not culled), or if not present already, it spawns. However, if the cardinality of the \mathbf{N} is equal to 0, 1, 4, 5, 6, 7, 8, or 9, an existing \mathbf{E}_{cube} is culled, or if not present, it is not spawned.

The above logic had to be tested to verify if the cubes (in the form of \mathbf{E}_{cube}) really experience structural stability (as directed by Ψ_{tower}) if stacked on top of each other in this way and eventually make a tower that is sufficiently structured (to develop a functioning $\mathbf{CE}_{\text{cube-tower}}$). Such a test would support establishing an early instance of empirical evidence that a \mathbf{CE} could emulate a bio-inspired optimization algorithm that helps to develop a built form.

The test would also serve as a prototype to develop a relationship between the built form and the algorithm without being dependent on design. Here, the role of design would be portrayed by the highly transparent and uncomplicated set of rules (in the form of Ψ_{spawn} and Ψ_{cull} rules) that are actually derived from rules of Cellular Automata (the Conway model) and modified to reflect basic intuitive structural rules.

Although a visual examination of all the rules, in the form of testing the rules on the base Rhino 7 environment (i.e. without conducting any computation) gives a clear idea that the Ψ_{spawn} and Ψ_{cull} rules are robust, physical tests and trials had to be conducted additionally by involving testers in the form of participants who would be willing to evaluate if the rules really work. These evaluators would have to possess a basic understanding of architecture, and thus students and practitioners in the AEC industry were considered to be ideal candidates.

The Author conducted a student workshop that was titled 'Designing Ways of Designing' in June 2018 at the IES (Indian Education Society's) College of Architecture in Mumbai, India. It was attended by 20 candidates – 16 students pursuing the B.Arch. degree and 4 recent graduates and practicing architects. For the testing, participants were first introduced to the concept of Computational Ecosystems and its research and were provided with a lecture on Cellular Automata and its implementation in the research. The participants were also given all the Ψ_{tower} rules in the form of an instruction manual so they would find it convenient to understand, examine and exercise the rule sets. After being thoroughly informed, the candidates were divided into four groups of five participants, and were tasked with testing the $CE_{\text{cube-tower}}$.

Each group was provided with a 50mm x 50mm grid as the base XY plane, made out of a 5mm Acrylic Sheet, and scored with a laser cutting machine to be able to display the grid lines. The groups were also provided with 5mm x 5mm x 5mm cubes (made out of the same Acrylic sheet mentioned above, by cutting with a laser cutter) and super glue to put the entire system together. The groups were encouraged to consider a random starting position with the only limit, that it should have no more than five cubes in the **initial state**. Thus, the testing groups were each developing individual $CE_{\text{cube-tower}}$ with the Acrylic cubes serving as the ϵ_{cube} of the system, and the instruction manual serving as the Ψ_{tower} rules.

Although the participants found the results to look quite stunning, and they found it absolutely entertaining to refer to a rule set that was generating random but logical form, owing to the tediousness of constantly consulting a ruleset and gluing tiny acrylic cubes to each other, the groups were asked to go for no more than ten iterations. This means, there was no rule for the termination of the $CE_{\text{cube-tower}}$ in this case, apart from the physical constraint of stopping after ten t_{tick} intervals.

Regarding the fact that all five individual $CE_{\text{cube-tower}}$ had different starting conditions in the form of $\epsilon_{\text{cube}}^{\text{initial}}$, all of them had different combinatory assemblages (governed by their specific N_{search} and N_{form} routines) and different morphologies.

The 5 different $CE_{cube-tower}$ and the different assemblages which can be considered as the empirical evaluation of the Ψ_{tower} rules are illustrated as a picture collage in fig. 4.9 below.

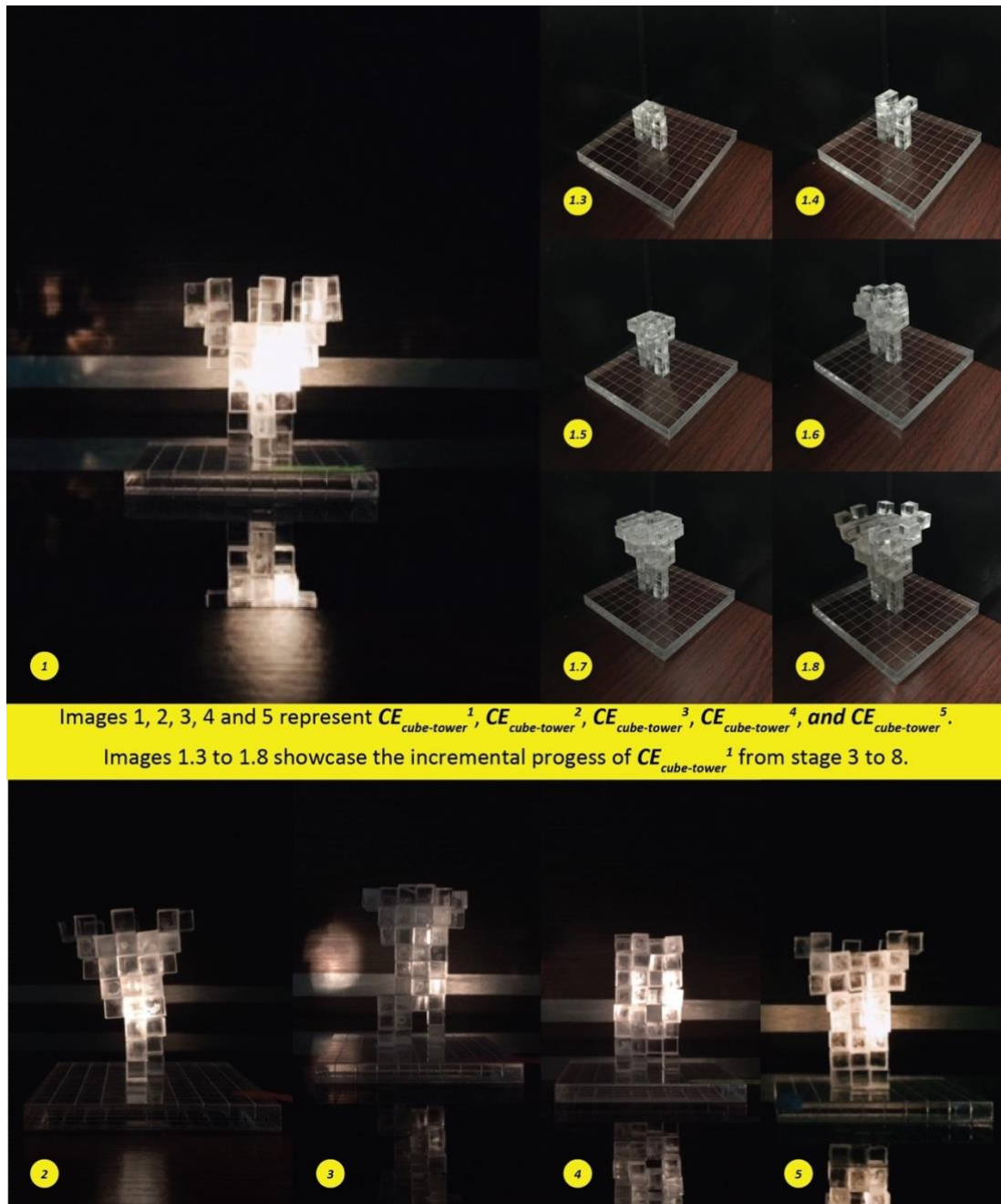


Figure 4.9 – Results and conclusions of the Designing ways of Designing workshop, conducted at IES College of Architecture, Mumbai India held to validate and prototype a $CE^{SESE} - CE_{cube-tower}$.
Photographs by Author (June 2018).

The evaluation of the $CE_{cube-tower}$ by means of physical modelling in a workshop for students and professionals in the AEC industry provided a deeper understanding of the validation of the concept of a CE as a link between the built form and the algorithm and the validation of the specific components of the CE . However, the limitations of physical modelling such as the aforementioned tediousness of consulting an instruction manual and building with small pieces restricted the runtime of a CE . Additionally, there was also a threat of the Ψ_{tower} rules being misread or misinterpreted at occasions leading to incorrect assemblages, which had to be thoroughly cross-checked. This counterintuitive behavior compelled the research to perform digital prototyping methods.

Thus, after validating the Ψ_{tower} rules of the $CE_{cube-tower}$, the natural progression in the prototyping stage was to develop a $CE_{cube-tower}$ computationally. However, although all the components of the CE were sufficiently defined, illustrated, exemplified and tested, in terms of computational semantics, the CE only existed in the form of the UML Class diagram (as illustrated in fig. 4.3). Therefore, a UML Sequence Diagram had to be set up to understand the role, interaction and runtime of all the classes to establish the $CE_{cube-tower}$ computationally. The UML Sequence Diagram for the construction of a $CE_{cube-tower}$ would have the following assumptions –

- The Actor would be considered as the designer who initiates the $CE_{cube-tower}$.
- $CE_{cube-tower}$ is considered as one of the objects, as it has its own runtime.
- The other objects in the diagram would be $E_{cube}^{initial}$, Ψ_{spawn} , E_{cube}^{form} , Ψ_{cull} in the chronological order of their application in the $CE_{cube-tower}$.
- The Diagram also has an **early rest** condition, that tests for early anomalies.
- The remaining algorithm however continues on a while loop that considers the $CE_{cube-tower}$ **rest** condition (mentioned in 4.2.2) to end the algorithm.

Fig. 4.10 shows the implementation of a UML Sequence Diagram to better understand and the role, interaction and runtime of the Ψ_{tower} , $\epsilon_{cube}^{initial}$, and their dependencies that determine the outcome of several different $CE_{cube-tower}$.

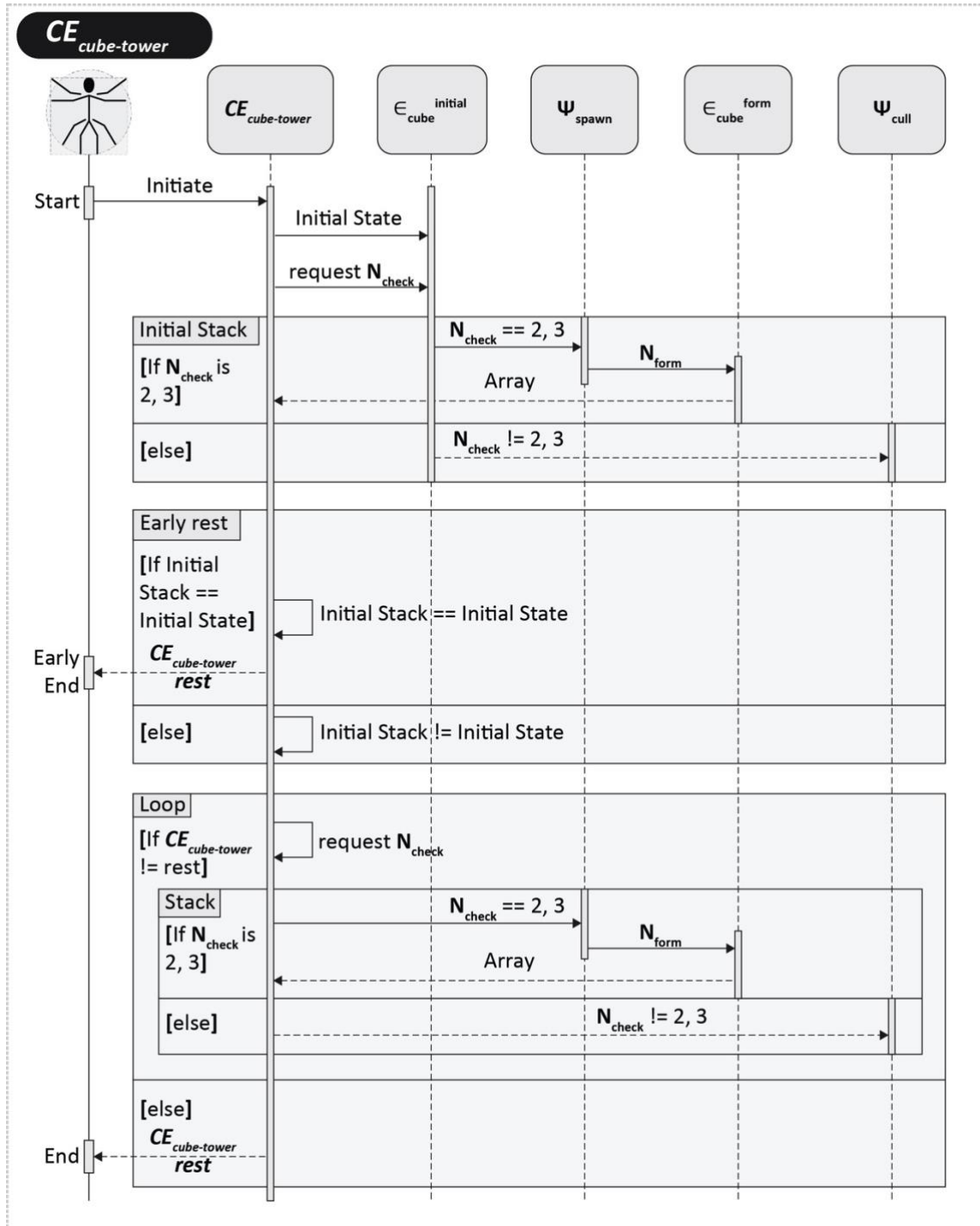


Figure 4.10 – The UML Sequence Diagram for a $CE_{cube-tower}$ with the role, interaction and runtime of the Ψ_{tower} , $\epsilon_{cube}^{initial}$, and their dependencies. Illustration and graphics by Author (Jan 2019).

Considering the UML Sequence Diagram established in fig. 4.10, an algorithm was made using Grasshopper (as Grasshopper definitions) and run inside the Rhino 7 interface (The definition labeled as $CE_{cube-tower}$ has been appended to the Annexure – Definitions). Although most of the functions mentioned in fig. 4.10 are available in the stock Grasshopper database, some operations had to be imported using third-party plugins. An overview of the various steps is mentioned below –

- **Initiate** – The Initiate function, and especially the populate operation of the $E_{cube}^{initial}$ class (as illustrated in fig. 4.3) required the algorithm to randomly generate a sequence of E_{cube} , to kickstart the definition. Although the operation of populating random points is a default component in Grasshopper, a series of operations involving conditionals was employed to ensure that the $E_{cube}^{initial}$ was focused on a certain part of the algorithm.
- **Array** – Although Grasshopper has built-in capabilities to produce cubes, a collection of points with the boundary properties of E_{cube} was implemented to maintain computational ease and manage exorbitant increase in file sizes which could occur as a result of the sudden increase in the number of E_{cube} .
- **N_{check}** – Tasked with the function of checking the cardinality of N for every E_{cube} , **N_{check}** needs to consider Ψ_{spawn} for either of two different conditional outputs that might come out of the check. Thus, the logic gate function OR was implemented to return true values for both the input conditionals 2, 3.
- **Loop** – As the computational heart of the algorithm, the loop function performs the Stack function until the $CE_{cube-tower}$ rest condition is reached. This intricate detail was achieved by employing a while loop in the algorithm. Performing loops is not a built-in tool in Grasshopper. Thus, an open-source plugin called Anemone¹³⁴ was used to perform the while loop.

¹³⁴ Anemone 0.4 (2015). Poznań, Poland: Mateusz Zwierzycki.

Employing the Grasshopper definition, several test $CE_{cube-tower}$ were prototyped in the Rhino 7 environment. As these were initial prototypes, the cardinality of $\epsilon_{cube}^{initial}$ was restricted to 2 and 3 ϵ_{cube} . Fig. 4.11 illustrates the outcomes of all possible inputs for the aforementioned $CE_{cube-tower}$ below. The figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), and the rest of the $CE_{cube-tower}$ thus denoted by ■ (R,G,B – 204,204,204) colored cube.

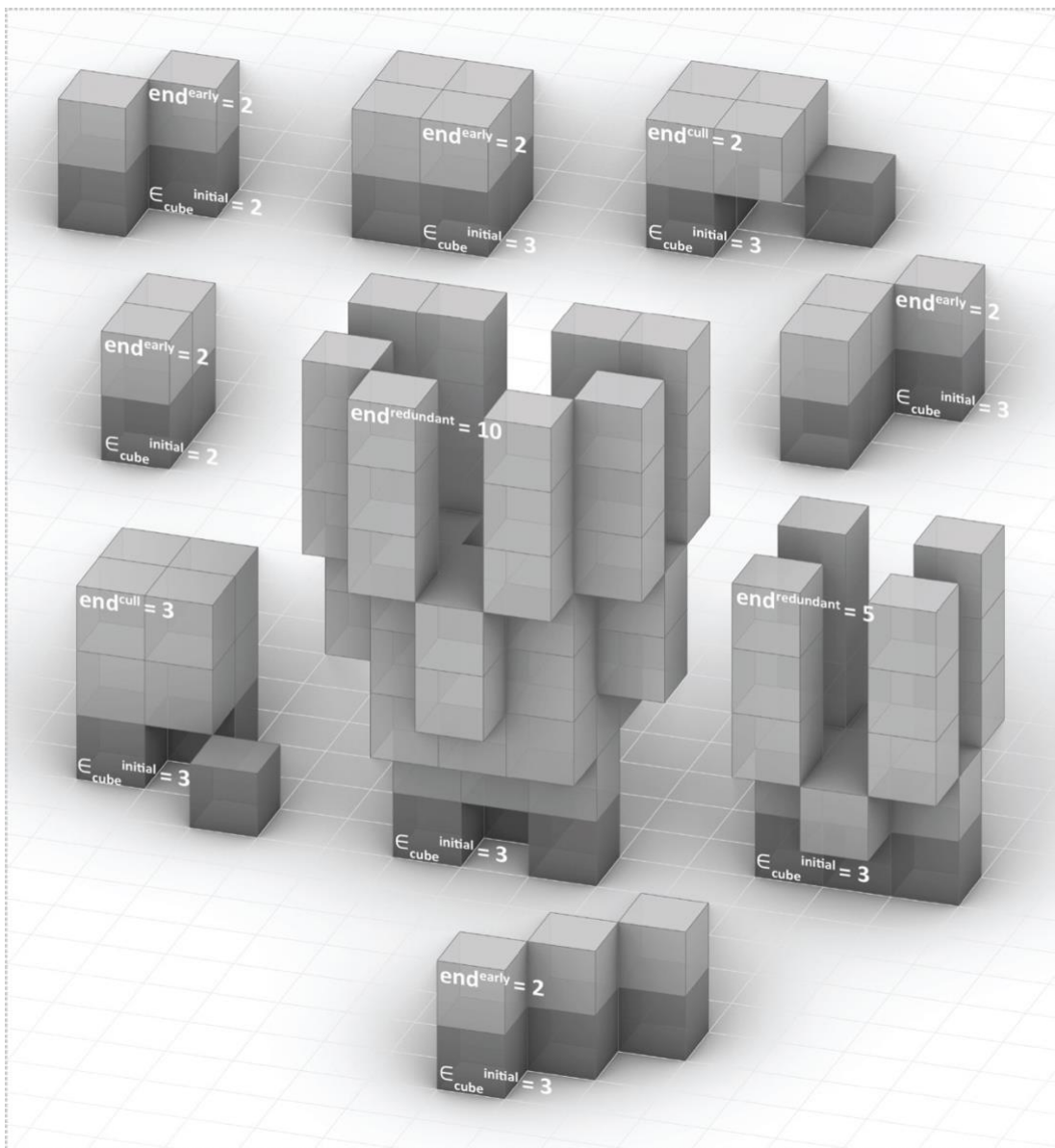


Figure 4.11 – Initial Prototypes of several $CE_{cube-tower}$ with 2 and 3 $\epsilon_{cube}^{initial}$. Model, algorithm, Illustration and graphics by Author (Jan 2019).

After the successful trials of Initial Prototypes, Fig. 4.12 illustrates the outcomes of one of the tallest $CE_{cube-tower}$ with $10 \text{ €}_{cube}^{initial}$ below. The figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), and the rest of the $CE_{cube-tower}$ thus denoted by ■ (R,G,B – 204,204,204) colored cube.

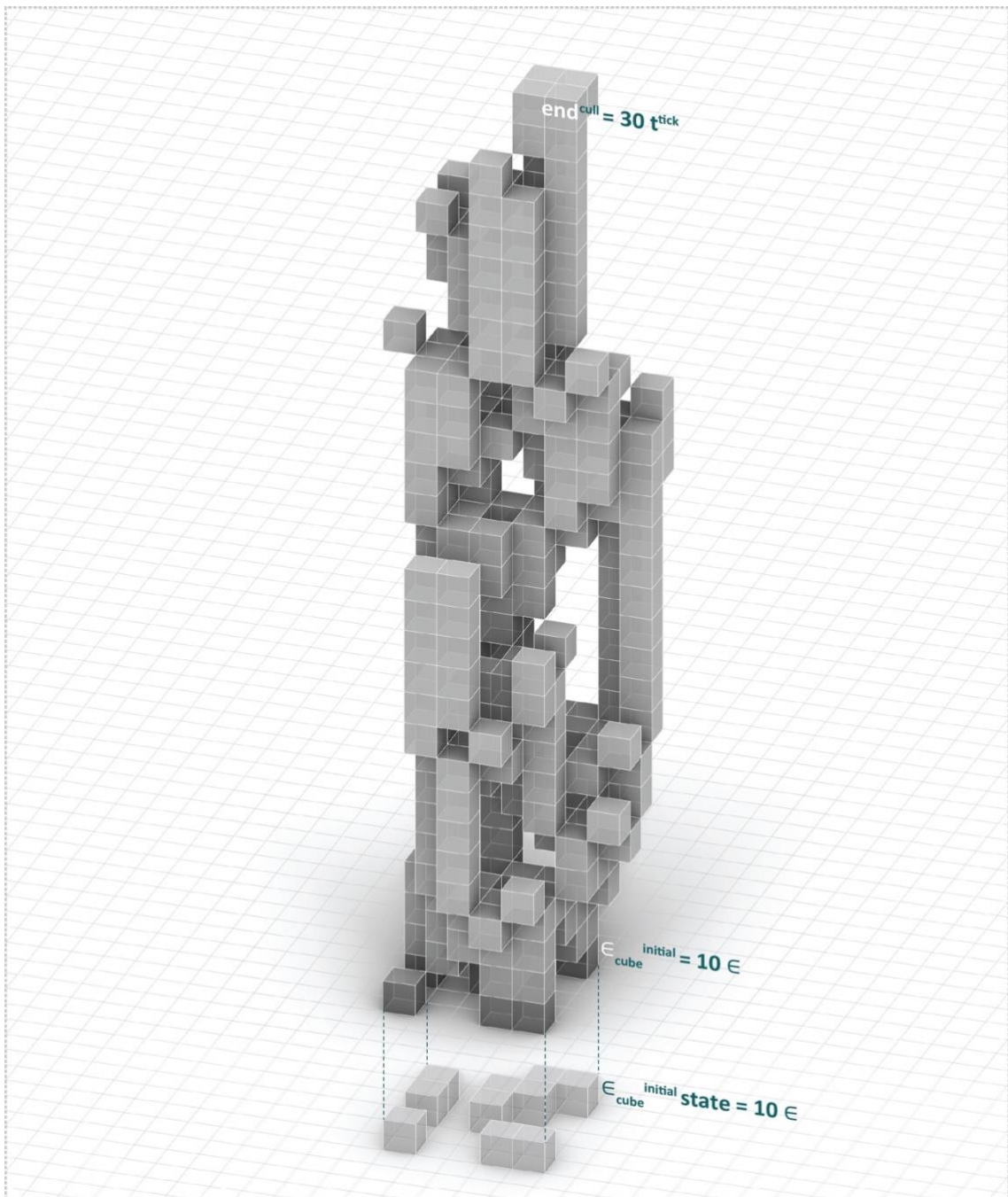


Figure 4.12 –Prototype of a selected $CE_{cube-tower}$ with $10 \text{ €}_{cube}^{initial}$ that had a runtime of 30 t^{tick} before reaching the rest state. Model, algorithm, Illustration and graphics by Author (Jan 2019).

4.2.4 $CE_{cube-tower}$

Although fig. 4.11 and fig. 4.12 show ample evidence of the successful executions of $CE_{cube-tower}$, more results focusing on the taxonomy creation have been illustrated in the next chapter (5| On the consequences of Computational Ecosystems) with a further analysis comparing all the other taxa from the procedural sequences in chapter 6 (On the investigative analysis of Computational Ecosystems).

However, before continuing with the next taxon, certain observations shall be mentioned in this chapter to avoid any redundancies that could devalue the quality of the upcoming taxa. Also, since the next taxa are far more complex as compared to the CE^{SESE} , these observations are essential before starting the Case Studies for the next taxa. The observations are as followed –

- Out of nine different iterations demonstrated in fig. 4.11 (as a combination of all of the possible iterations with 2 or 3 $\epsilon_{cube}^{initial}$) five returned to an Early End condition. Although this function is designed to filter redundancies at early stages of the algorithm, this number (which is 55.5% of the possible iterations) is quite large.
- As a solution, an optimization algorithm could be employed at this stage to minimize (and if possible, thoroughly eradicate) the possibility of equality between the **Initial Stack function** and the **Initial State function**. Several other methods could also be theorized and tested.
- The above problem is not demonstrated in fig. 4.12 (the case where one of the tallest iterations with 10 $\epsilon_{cube}^{initial}$ was selected). That is because the iteration selected in the illustration was chosen to be as different, tall, and complex as possible. All the possible Early End $CE_{cube-tower}$ were visually (and manually) searched and eradicated.

- Here, because the tower was only conformed to its structural stability there were no rule sets considering the optimization of its height. Moreover, the $CE_{cube-tower}$ rest condition practically restricted the tower from soaring with an infinitely repeating or oscillating combinatorial assemblages.
- However, a set of rules that would make sure that the tower should be taller, complex, and visually distinct while serving the structural stability rules could be considered as an interesting set of Ψ that could possibly serve as a starting point for a CE^{SEME} .
- The physical simulation with acrylic sheets and acrylic blocks was quite an imaginative approach to involve students and practitioners of the AEC industry in the evaluation and setup of a proof of concept. Thus, the workshop model can be possibly upgraded and used for the remaining taxa.
- However, the limitations of the physical simulations were also quite pronounced. Due to the cumbersome nature of handling tiny blocks, there was always a possibility of a participant making mistakes and thus making the evaluation process all the more useless. As one cannot be thoroughly sure of the errors, unless the whole process is automated, this cannot be countered. Thus, the implementation of workshops to involve participants into testing the concept physically is a bit counter-intuitive and can be avoided in the evaluation of the upcoming taxa.
- Moreover, the workshops could be directed towards developing and evaluating the algorithm to check for errors and possible improvements. Although this would involve teaching and tutoring the participants in the use of software such as Rhino 7 and Grasshopper (with the required plugins).

It would be crucial to include the above-mentioned observations in the upcoming taxa and perform the primary objectives in a more nuanced manner.

4.3 Multiple Elements Single Economy Ecosystem (CE^{MESE})

Similar to CE^{SESE} , the conceptual framework of a CE^{MESE} has already been sufficiently illustrated in the previous chapters (introduced in 1.2.2, illustrated in fig. 1.7, defined in 3.2.2). Moreover, a Structural UML Diagram establishing the attributes and operations of a canonical version the $\mathbf{\epsilon}$ and $\mathbf{\psi}$ has also been illustrated (in fig. 3.3 as per 3.2.3). However, like CE^{SESE} , to construct, taxonomize and prototype a procedural sequence, the research would have to consider specific tangible parameters for the concept of a CE^{MESE} , which could be derived from a combination of real-life examples and constraints and the case studies of the already established CE^{SESE} .

The CE^{MESE} also serves as an extension and continuation to the $CE_{cube-tower}$, primarily owing to the multiplicity of the $\mathbf{\epsilon}$ parameter. However, to make sure that the complexity is raised with relative ease and consistency, the Author has decided to continue with the conceptual example of an $MESE$ illustrated in 1.2.2. Use of the example scenario would be especially instrumental in methodically increasing the $\mathbf{\epsilon}$ parameter to 2 (instead of the ambiguous multi) and to using canonical Structural UML Diagram. Thus, before specifically defining each parameter, the overall components of the CE^{MESE} can be –

- $\mathbf{\epsilon}^1$ – As illustrated in the example scenario, the $\mathbf{\epsilon}$ would be represented by hexahedrons or cubes, with unit dimensions.
- $\mathbf{\epsilon}^2$ – As illustrated in the example scenario, the $\mathbf{\epsilon}$ would be represented by cuboctahedrons, with unit dimensions.
- $\mathbf{\psi}$ – As illustrated in the example scenario, the $\mathbf{\psi}$ would be represented by structural stability in the form of axial loads.

Regardless of the above generalization, the CE^{MESE} would require to undergo all the primary objectives while pursuing the operational objectives for this taxon.

4.3.1 Case Studies

Although serving as a conceptual successor to the CE^{SESE} , and by definition being a continuation to the CE^{SESE} , the CE^{MESE} which theoretically and semantically means that it can be treated as an ecosystem that is inhabited or cohabited by multiple predetermined species and these species (individually, categorically and collectively), are constructed, monitored, and governed by one and only one predetermined rule set, the CE^{MESE} does not serve as a direct sequel to the CE^{SESE} in terms of fulfilling the Procedural Sequences.

However, the CE^{MESE} contains and exhibits certain attributes related to the ϵ and ψ that cannot be exhibited by the CE^{SESE} due to its semantical constraints. The multiplicity of the ϵ parameter doesn't just imply that there would be two distinct ϵ entities, but the iterations of their collective assemblages would also have to be treated as a distinct ϵ parameter. Thus, the following addition must be made to the assumptions stated in the last section –

- ϵ^1 – Hexahedrons or cubes, with unit dimensions, which can be programmed as sentient elements similar to the ϵ_{cube} , that are aware of their existence, and thus can be considered as $\epsilon_{\text{biotic}}^1$.
- ϵ^2 – Cuboctahedrons, with unit dimensions, which can be programmed as sentient elements similar to the ϵ_{cube} , (however with morphological modifications) that are aware of their existence, and thus can be considered as $\epsilon_{\text{biotic}}^2$.
- ϵ^{1-2} – A collective entity of the ϵ^1 and ϵ^2 , which can be programmed as sentient elements being treated as a ϵ_{cube} array (similar to the Arrays formed by the Stack operations) that are aware of their existence, and thus can be considered as $\epsilon_{\text{biotic}}^{1-2}$.

The example of the collective assemblages of the *Eciton hamatum* forming live bridges out of their own bodies (elaborated in 4.2.1, and illustrated in fig. 4.1) can be considered again as a theoretical precedent for the CE^{MESE} . The $CE_{eciton-bridge}$ (theorized, exemplified and illustrated in 4.2.1), consequently can also be considered as semantic precedence for establishing the theory of the CE^{MESE} . However, the constituent parts only consider a single ϵ parameter, which does not conform to the requirements of a CE^{MESE} . Thus, the theoretical precedents laid down by the $CE_{eciton-bridge}$ would have to be modified to accommodate the required components.

Although in reality the ant bridges are constructed by the *Eciton hamatum*, which share the same morphologies, and thus similar physical and physiological constraints, they serve different purposes in the composition of a bridge. Owing to the different functions of structural elements of a bridge, and by considering that the $CE_{eciton-bridge}$ would be a bridge that is made out of a system of connectors, the constituent ϵ entities $CE_{eciton-bridge}$ would have to be of the following types –

- **The Anchoring elements** – These elements would serve the purpose of anchoring the bridge to the ground condition (considering that the ground condition could be the forest ground, tree barks, stone edges or a combination). These elements can thus be called as ϵ^{anchor} , and they would be the initial ϵ in the formation of the bridge.
- **The Spanning elements** – These elements would serve the purpose of spanning the bridge across the gap (irrespective of its length and height from the actual ground level). These elements can thus be called as ϵ^{span} , and they would be the ϵ that would have to be increased or decreased in case the bridge needs a change in length or composition.
- **The Connection elements** – These elements would connect the ϵ^{anchor} to the ϵ^{span} , and vice-versa. These elements can be called as $\epsilon^{connector}$, and their morphology would be as an assemblage.

These different types of elements that form the components of the ϵ parameters of the $CE_{eciton-bridge}$ have also been illustrated in the fig. 4.13 as shown below. Although the image is sourced from a result of fieldworks conducted by the research on ant bridges, illustrations have been overlaid on the image to explain the aforementioned idea of multiple ϵ entities.



Figure 4.13 – Ant bridge from below. Original Image from the fieldwork on Army ants (*Eciton hamatum*) induced to form a very large bridge over a wide gap by Chris R. Reid. Source: <https://www.princeton.edu/news/2015/11/30/ants-build-living-bridges-their-bodies-speak-volumes-about-group-intelligence>. Illustration and graphics by Author (Jan 2019).

As seen in fig. 4.13, the morphology of the as ϵ^{anchor} , ϵ^{span} , and $\epsilon^{connector}$ could be quite similar, however, their individual functionalities in maintaining the structural integrity of the $CE_{eciton-bridge}$ would be quite distinct. Moreover, their deployment stages in both the formation, modification and the eventual dismantling of the $CE_{eciton-bridge}$ would also be thoroughly defined so as to make sure that the bridge uses an optimum amount of its constituent parts.

The Ψ parameter of the $CE_{eciton-bridge}$ (as introduced in 4.2.1) would also have a few modifications to accommodate the variations done in the ϵ parameters. The updated Ψ parameter can be defined as followed –

- Ψ_{bridge} – The bridging ability. Forming collective assemblages by means of connection techniques, span-depth ratios for optimum bridge structures, and load calculations (considering the live loads and dead loads) for each of the 3 distinct ϵ entities - ϵ^{anchor} , ϵ^{span} , and $\epsilon^{connector}$.

Considering the fact that the Ψ parameter would now also have to be trifurcated into distinct Ψ entities to accommodate the ϵ parameters, the names of all the components would have to be amended. Thus, the new components of the updated $CE_{eciton-bridge}^2$ would be –

- $\epsilon_{eciton}^{anchor}$ – These would be the Army ants anchoring the bridge.
- ϵ_{eciton}^{span} – These would be the Army ants spanning the bridge.
- $\epsilon_{eciton}^{connector}$ – These would be connecting the $\epsilon_{eciton}^{anchor}$ and ϵ_{eciton}^{span} .
- Ψ_{bridge}^{anchor} – These would be the rule sets for $\epsilon_{eciton}^{anchor}$.
- Ψ_{bridge}^{span} – These would be the rule sets for ϵ_{eciton}^{span} .
- $\Psi_{bridge}^{connector}$ – These would be the rule sets for $\epsilon_{eciton}^{connector}$.

Thus, in spite of the fact that a CE^{MESE} has one and only one Ψ parameter, it could have distinct multiple Ψ entities depending on the number of ϵ parameters and their interactions. Although this further underlines the fact that the CE^{MESE} serves as a conceptual successor but to the CE^{SESE} , and neither of them can be treated as a subset of the other, it also helps in understanding the dependency of the CE^{MESE} over its procedural predecessor for its conceptual and theoretical framework.

Thus, the similarities between a CE^{MESE} (derived from the example scenario) and a $CE_{eciton-bridge}^2$ can be illustrated as shown in fig. 4.14. below.












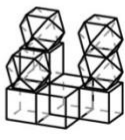
MESE						
$CE_{eciton-bridge}^2$	ϵ_{anchor}	ϵ_{span}	$\epsilon_{connector}$	ψ_{anchor}	ψ_{span}	$\psi_{connector}$
CE^{MESE}						
	ϵ_{cube}	$\epsilon_{cubocta}$	$\epsilon_{cube-cubocta}$	ψ_{cube}	$\psi_{cubocta}$	$\psi_{cube-cubocta}$

Figure 4.14 – Comparing the CE^{MESE} and a $CE_{eciton-bridge}^2$. Illustration and graphics by Author (Jan 2019).

Thus, based on the analogies illustrated in fig. 4.14 and the conceptual semantics established for a $CE_{eciton-bridge}^2$, the CE^{MESE} could be based on the already established $CE_{cube-tower}$ (as defined in 4.2.1). However, in case of the CE^{MESE} the constituent ϵ entities and ψ entities, would have to be defined in relation to the $CE_{eciton-bridge}^2$ (unlike the $CE_{cube-tower}$ which was established considering the semantics of constructing the $CE_{eciton-bridge}$). Therefore, the CE^{MESE} is defined as followed –

- ϵ entities – or ϵ^{MESE} entities will be considered as computational substitutes to all the three ϵ entities established for the $CE_{eciton-bridge}^2$. Although the functionalities and structural properties of the $\epsilon_{eciton}^{anchor}$, ϵ_{eciton}^{span} , and $\epsilon_{eciton}^{connector}$ do not directly translate to the ϵ^{MESE} entities, they contain the same semantics as established in a $CE_{eciton-bridge}^2$. Thus, continuing on the example scenario, the ϵ^{MESE} entities can be further trifurcated as ϵ_{cube} , $\epsilon_{cubocta}$, and $\epsilon_{cube-cubocta}$. The specific properties of each entity, which are relevant for simulation, will be defined in the next section (as per 4.3.2), but the general properties of all the ϵ^{MESE} entities can be as followed –

- All the ϵ^{MESE} entities should be considered as static agents that can be spawned or culled by the CE^{MESE} at the expense of the equilibrium state which will be determined by the Ψ entities.
- All the ϵ^{MESE} entities should be made to be self-aware of themselves, in the sense that they should be able to identify and quantify its position, orientation, weight, and bounds.
- All the ϵ^{MESE} entities should also be made to be self-aware of their surroundings and their physical properties, such as cardinality of neighbours, array of neighbours, and distance from the initial plane.
- Ψ entities – or Ψ^{MESE} entities will be considered as computational substitutes to the Ψ entities established for the $CE_{eciton-bridge}^2$. Although the functionalities and structural properties of the Ψ entities of the $CE_{eciton-bridge}^2$ do not directly translate to the Ψ^{MESE} entities, they have the same semantics as established in a $CE_{eciton-bridge}^2$. Thus, the Ψ^{MESE} entities can be trifurcated as Ψ_{cube} , $\Psi_{cubocta}$, and $\Psi_{cube-cubocta}$. The properties of the Ψ^{MESE} entities can be defined as –
 - The CE^{MESE} would have all the Ψ^{MESE} entities performing in unison to pursue a tower stacking logic. Thus, the Ψ^{MESE} entities can be renamed as $\Psi_{tower-cube}$, $\Psi_{tower-cubocta}$, and $\Psi_{tower-cube-cubocta}$.
 - All the Ψ^{MESE} entities should be designed with two-state rationale for all the ϵ^{MESE} entities derived from the Ψ^{tower} (defined in 4.2.1) while conforming to the understanding of vertical stacking.
 - In short, Ψ^{MESE} entities should seek an equilibrium state that spawns ϵ^{MESE} entities when the tower is under-structured, and culls ϵ^{MESE} entities when it is over-structured, thus maintaining a reciprocal coupling.

After establishing and defining the ϵ^{MESE} entities and Ψ^{MESE} entities, the CE^{MESE} could be specifically defined under the computational modelling guidelines of CA (as elaborated in 3 |) as followed –

- The CE^{MESE} which owing to its specific combination of ϵ^{MESE} entities and Ψ^{MESE} entities can be termed as $CE_{cube-cubocta-tower}$, will be defined as a functioning ecosystem constituting of sentient, context aware cubes as ϵ^{MESE} entities that interact with each other considering the rule sets assigned by Ψ^{MESE} entities to spawn or cull the ϵ^{MESE} entities until the runtime of the $CE_{cube-cubocta-tower}$.
- The $CE_{cube-cubocta-tower}$ will perform solitarily without any additional internal components or partial runtimes. Moreover, there won't be any context aware CE that would have to be added externally. Thus, also in this taxon, one and only one CE that is $CE_{cube-cubocta-tower}$ would be performed for its entire runtime.
- Although derived from the intricate bio-inspired behavior of *Eciton hamatum* creating and maintaining living bridges for foraging trails, the $CE_{cube-cubocta-tower}$ is a slightly complex vertical stacking algorithm for a combination of cubes and cuboctahedra. Thus, the rule sets for Ψ^{MESE} entities can be adapted from the Conway model of Cellular Automata (as per 2.3.2). However, these will be slightly modified to accommodate all the ϵ^{MESE} entities.
- Just like the Ψ_{tower} (established in 4.2.1), the Ψ^{MESE} entities would differ from the Conway model of Cellular Automata in the parameter of time. That is, the new ϵ^{MESE} entities will be spawned or culled on the upper levels in the 3D grid.

Considering all the above assumptions, definitions, illustrations and examples for the $CE_{cube-cubocta-tower}$, the simulations for this taxon can now be performed. However, some information still needs to be considered to illustrate the empirical derivations of all the case studies. For example, the ϵ^{MESE} and Ψ^{MESE} entities still need to be dimensionally, and geometrically defined to consider them for computational use.

4.3.2 Simulations

However, just like it was done for the CE^{SESE} , the canonical CE^{MESE} Structural UML Diagram (as per fig. 3.3) for the $CE_{cube-cubocta-tower}$ has to be reassessed and repurposed. Fig. 4.15, as shown below, illustrates the modifications to the canonical version.

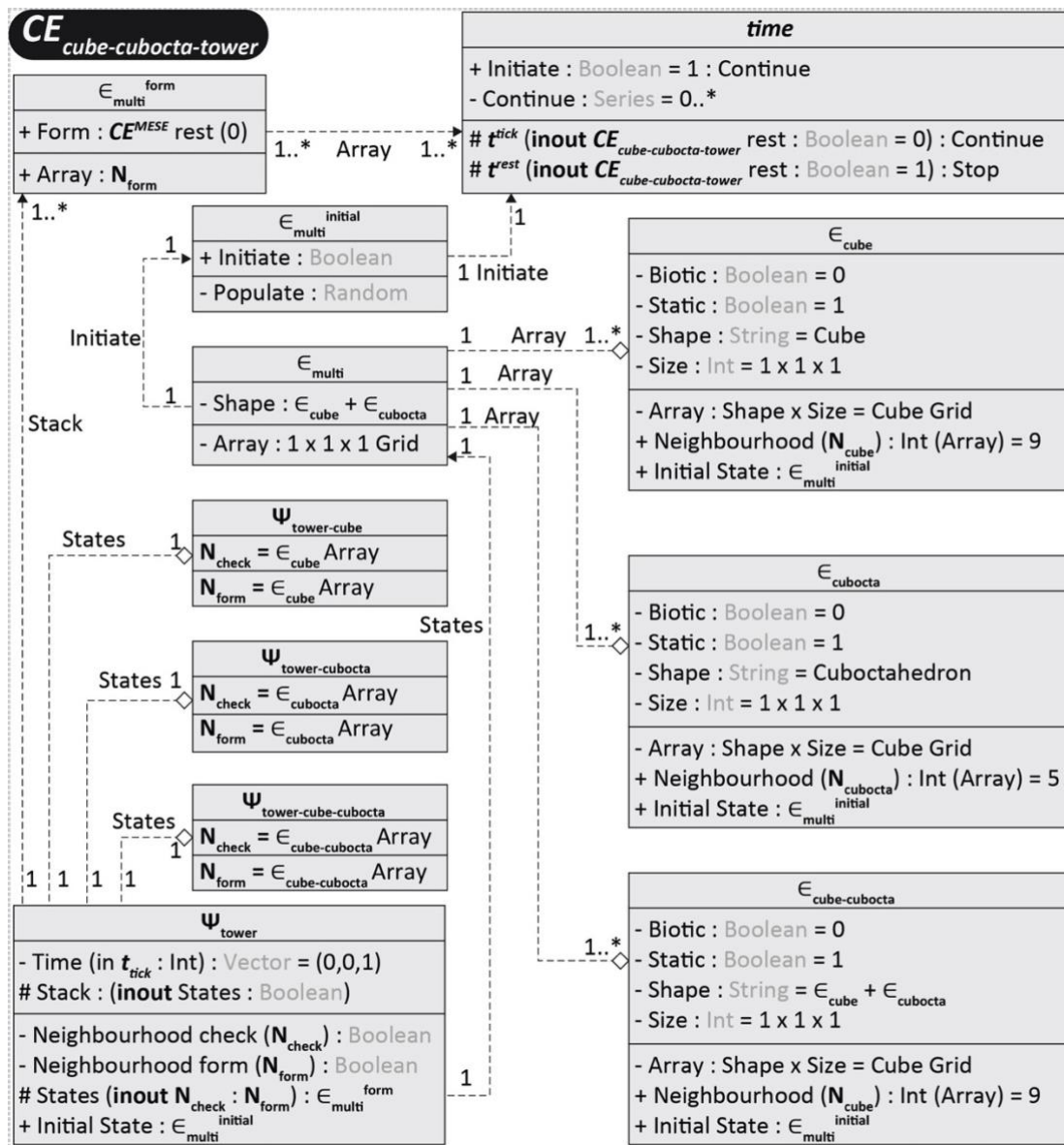


Figure 4.15 – Modifications made to the canonical CE^{MESE} UML Diagram to accommodate the parameters required for establishing the $CE_{cube-cubocta-tower}$. Illustration and graphics by Author (May 2019).

Elaborating on the terminologies introduced in fig. 4.15, the attributes and operations of the classes can be defined as followed –

- t^{tick} – The periodic increment of time, ensuring that the $CE_{cube-cubocta-tower}$ runs.
- t^{rest} – If the condition $CE_{cube-cubocta-tower}$ **rest** is fulfilled, the t^{rest} will be activated. This means that the time increment will stop and the $CE_{cube-cubocta-tower}$ will be outputted to the user.
- $CE_{cube-cubocta-tower}$ **rest** – Is the situation where an ϵ_{multi}^{form} array for the last three t^{tick} intervals is equal or repetitive.
- ϵ_{multi}^{form} array – Is the collective assemblage of ϵ_{multi} for a certain t^{tick} interval.
- ϵ_{multi}^{form} – Is the operation for the formation of a combination of ϵ_{cube} , $\epsilon_{cubocta}$, and $\epsilon_{cube-cubocta}$ depending on the N_{check} and N_{form} routines for the $\Psi_{tower-cube}$, $\Psi_{tower-cubocta}$, and $\Psi_{tower-cube-cubocta}$.
- N_{form} – Is the binary inputs for the cells to be either spawned (input = 1) or culled (input = 0) depending on the N_{check} conditional.
- N_{check} – Is the binary output from the cells of the distinct neighbourhoods (N) to check if the cells in the N of the previous t^{tick} interval are either spawned (input = 1) or culled (input = 0).
- N – The neighbourhood of the ϵ_{multi} in question (as in every ϵ_{cube} , $\epsilon_{cubocta}$, and $\epsilon_{cube-cubocta}$) depending on how many neighbouring ϵ_{multi} from the previous t^{tick} interval are to be considered to determine the N_{check} and N_{form} routines that eventually determine the **Stack** operation.

Thus, determining the N becomes the most important step before considering to explain the other concepts required to run the $CE_{cube-cubocta-tower}$.

Similar to the computational environment and the revised infinite 3D Square grid that was considered for the $CE_{cube-tower}$ (as per 4.2.2), in case of the $CE_{cube-cubocta-tower}$ as well, the modelling has been performed in Rhino 7 (as explained in 3.3), the programming has been performed in Grasshopper for Rhino 7 (as explained in 3.3) and visualizations have been performed in VRay for Rhino (as explained in 4.1).

Although most of the physical operations of the $CE_{cube-cubocta-tower}$ are similar to those of the previously established $CE_{cube-tower}$ (as explained in 4.2.2), there is a complication that arises from the multiplicity of the ϵ entities. Owing to the morphology of the 3 ϵ entities, ϵ_{cube} , $\epsilon_{cubocta}$, and $\epsilon_{cube-cubocta}$, it is not difficult to employ the infinite 3D Square grid conformed at the XY plane as the computational environment. However, to understand the distinct N for the ϵ entities, the morphology of the $\epsilon_{cubocta}$, and $\epsilon_{cube-cubocta}$ have to be thoroughly understood (as the ϵ_{cube} has been thoroughly explained in 4.2.2, it need not be explained again).

A cuboctahedron is a result of maximum truncation performed at all the vertices of a cube. The cuboctahedron can also be obtained by performing maximum truncation at all the vertices of a tetrahedron. Moreover, as a cuboctahedron can be considered as a derivative of a cube or octahedron, a cube with radius R_{cube} will always perfectly bound a cuboctahedron with radius $R_{cubocta}$. The relation between the morphologies and radii of Cubes and Cuboctahedron can be seen in fig. 4.16 as illustrated below.

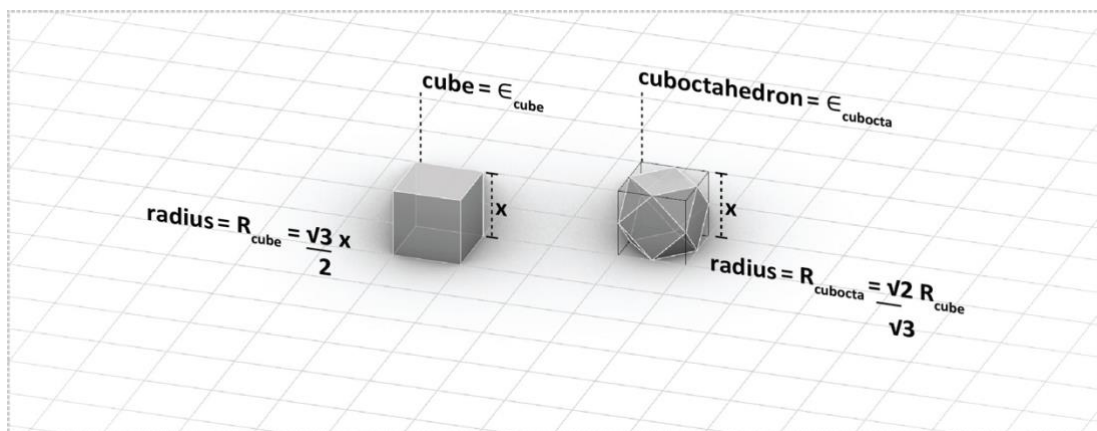


Figure 4.16 – Relation between the morphologies and radii of Cubes and Cuboctahedron. Model, Illustration and graphics by Author (May 2019).

The distinct **N** for all the **€** entities can now be established. Fig. 4.17 illustrates the **N_{cube}**, which determines the neighbourhood for every **€_{cube}**. The **N_{cube}** has been conceptually derived from a Moore's Neighbourhood (as explained in 2.3.2), however, unlike its predecessor, the **N_{cube}** is only considered for existing or spawned **€_{cube}**.

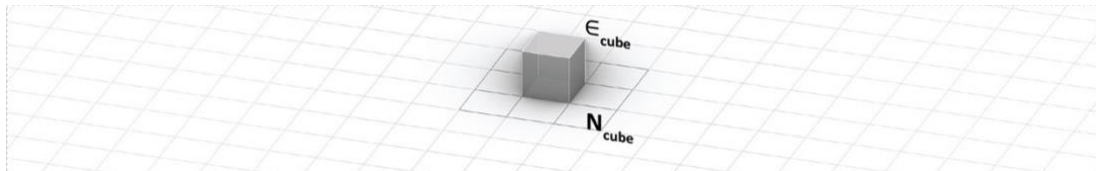


Figure 4.17 – An **€_{cube}** surrounded by the **N_{cube}**. Model and graphics by Author (May 2019).

Fig. 4.18, however illustrates the **N_{cubeocta}**, which determines the neighbourhood for every **€_{cubeocta}**. The **N_{cubeocta}** has been conceptually derived from a Von Neumann Neighbourhood (as explained in 2.3.1), however, unlike its predecessor, the **N_{cubeocta}** is only considered for existing or spawned **€_{cubeocta}**.

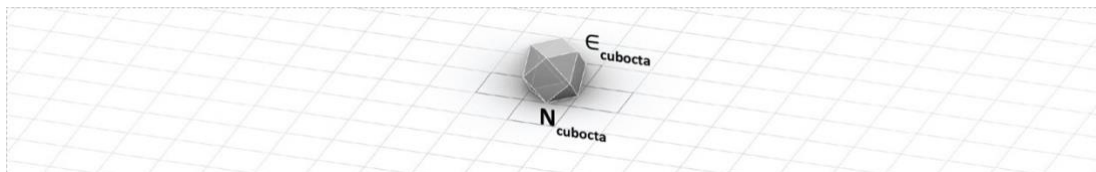


Figure 4.18 – An **€_{cubeocta}** surrounded by the **N_{cubeocta}**. Model and graphics by Author (May 2019).

Unlike the **N_{cube}** and **N_{cubeocta}** which are based on spawned **€** entities, the **N_{cube-cubeocta}** is regarded for non-existing or culled **€_{cube}** or **€_{cubeocta}**, and it considers the neighbouring existing **€_{cube}** or **€_{cubeocta}** for their respective **N_{cube}** and **N_{cubeocta}** to determine its own neighbourhood termed as **N_{cube-cubeocta}**. Fig. 4.19, illustrates a **N_{cube-cubeocta}**.

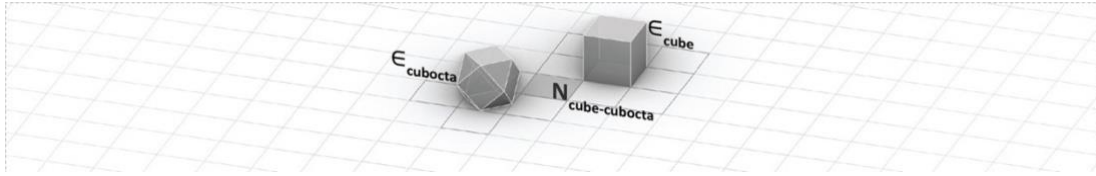


Figure 4.19 – A culled E_{cube} or $E_{cubocta}$ surrounded by the $N_{cube-cubocta}$. Model and graphics by Author (May 2019).

This specific and discrete distinction of the N , eventually helps in establishing the specific N_{check} and N_{form} routines. However, owing to the flexibility of CA and its rules, an astute modification in the state conditions of a Conway Model (as elucidated in 2.3.2), can be employed to establish bespoke, distinct N_{check} and N_{form} routines for all the 3 distinct N considerations for the $CE_{cube-cubocta-tower}$.

Thus, considering the $CE_{cube-tower}$ as precedence, and deriving from the Conway Model of CA , the state conditions for the $CE_{cube-cubocta-tower}$ can be stated as –

- Every existing $E_{multi-n+1}$ at the $t^{tick-n+1}$ interval interacts with its N consideration of the either the E_{cube-n} (in the form of N_{cube-n}) or $E_{cubocta-n}$ (in the form of $N_{cubocta-n}$) at the t^{tick-n} interval, thus performing N_{check} . For the $t^{tick-n+1}$ interval, it then performs the N_{form} routine based on the following conditions –
 - If the N is N_{cube-n} with an array of 2 or 3 E_{multi} at the t^{tick-n} interval, the N_{form} spawns a $E_{cube-n+1}$ for the $t^{tick-n+1}$ interval.
 - Also, for the $N_{cubocta-n}$ with an array of 2 or 3 E_{multi} at the t^{tick-n} interval, the N_{form} spawns a $E_{cubocta-n+1}$ for the $t^{tick-n+1}$ interval.
- Every non-existing $E_{multi-n+1}$ at the $t^{tick-n+1}$ interval interacts with its $N_{cube-cubocta}$ at the t^{tick-n} interval, thus performing N_{check} . For the $t^{tick-n+1}$ interval, it then performs the N_{form} routine based on the following conditions –
 - With an array of 4 E_{multi} in its $N_{cube-cubocta}$ at the t^{tick-n} interval, the N_{form} spawns a $E_{cubocta-n+1}$ for the $t^{tick-n+1}$ interval.

- With an array of 3 ϵ_{cube} or 4 $\epsilon_{\text{cubocta}}$ in its $N_{\text{cube-cubocta}}$ at the $t^{\text{tick-n}}$ interval, the N_{form} spawns a $\epsilon_{\text{cubocta-n+1}}$ for the $t^{\text{tick-n+1}}$ interval.
- For every other condition of N_{check} at the $t^{\text{tick-n}}$ interval, the N_{form} culls all the existing or non-existing ϵ_{multi} for the $t^{\text{tick-n+1}}$ interval.

Thus, considering the aforementioned state conditions, the condition for the existing state of the ϵ_{multi} array, which can be either continued existence (as in not culled) or newly existing (as in spawned) can be illustrated as in fig. 4.20 below.

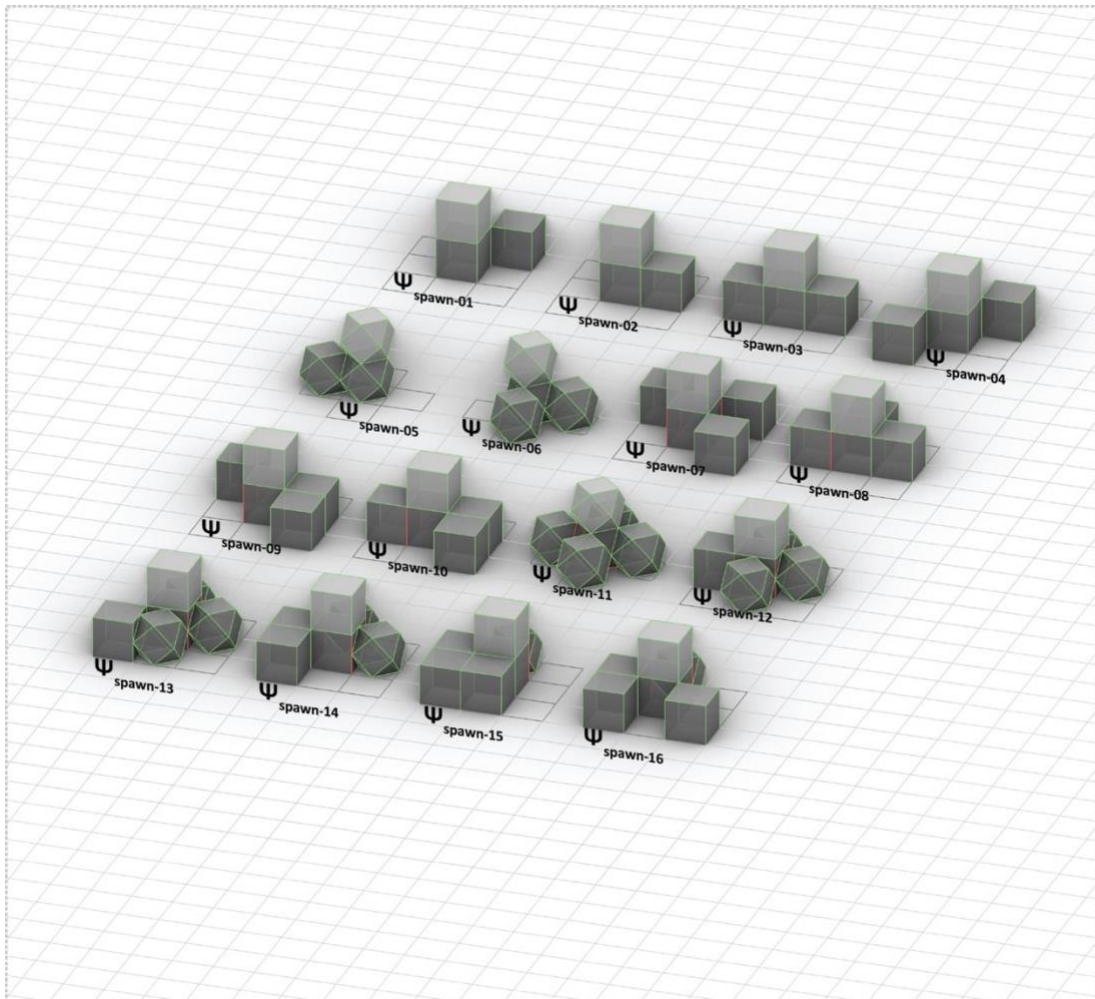


Figure 4.20 – $\epsilon_{\text{multi-n}}$ array and their corresponding $\epsilon_{\text{multi-n+1}}$ array considering the state conditions for the existing states of ϵ_{multi} array. Model and graphics by Author (May 2019).

The $\epsilon_{\text{multi-n+1}}$ at the $t^{\text{tick-n+1}}$ interval is thus denoted by \blacksquare (R,G,B – 204,204,204) colored cube, and the corresponding $\epsilon_{\text{multi-n}}$ at the $t^{\text{tick-n}}$ interval is denoted by \blacksquare colored cube (R,G,B – 129,129,129). Also, the existing cubes are denoted by \blacksquare (R,G,B – 57,188,40) colored cube, and the non-existing cubes are denoted by \blacksquare (R,G,B – 255,21,21) colored cubes.

And, considering the aforementioned state conditions, the condition for the non-existing state of the ϵ_{multi} array, which can be terminated existence (as in culled) or not newly existing (as in not spawned) can be illustrated as in fig. 4.21 below.

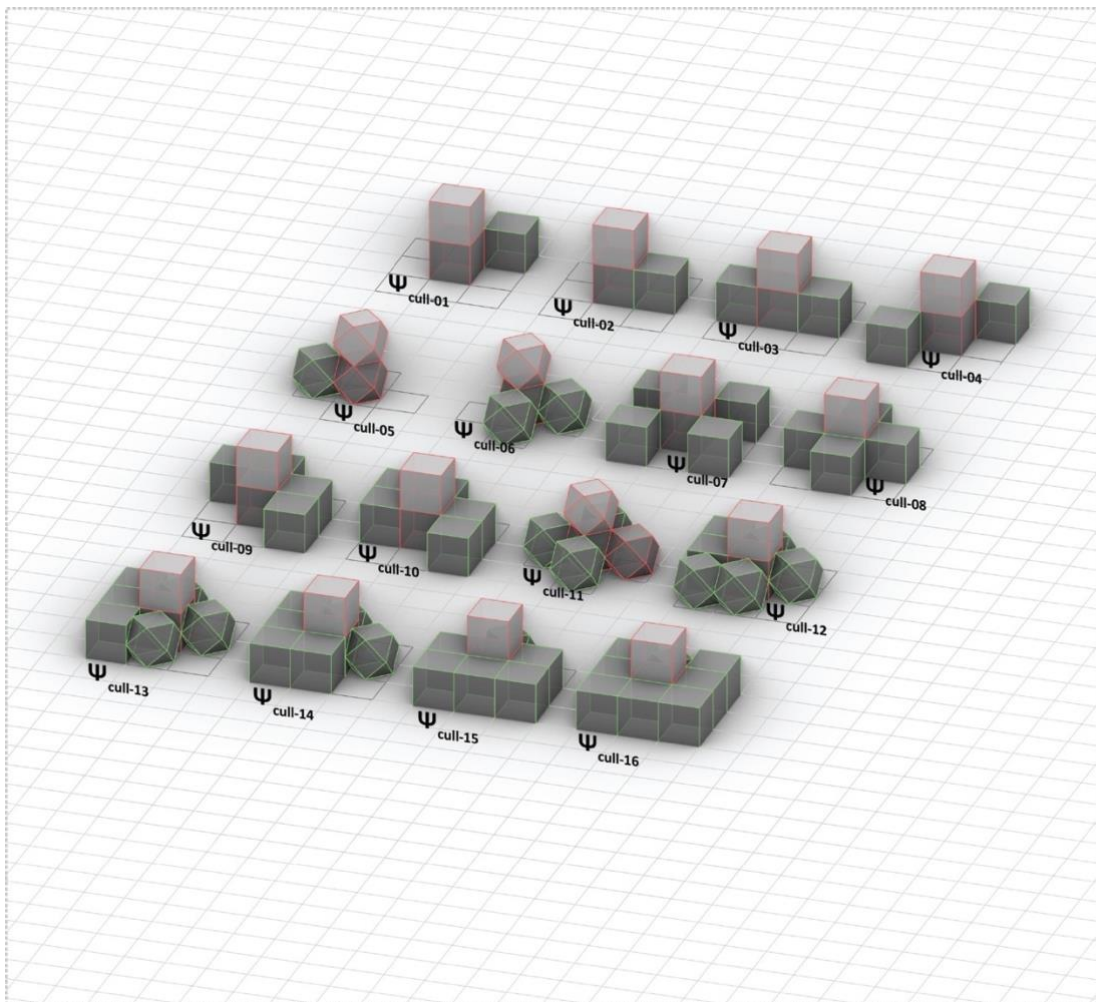


Figure 4.21 – $\epsilon_{\text{multi-n}}$ array and their corresponding $\epsilon_{\text{multi-n+1}}$ array considering the state conditions for the non-existing states of ϵ_{multi} array. Model and graphics by Author (May 2019).

The $\mathbf{E}_{\text{multi-}n+1}$ at the $t^{\text{tick-}n+1}$ interval is thus denoted by \blacksquare (R,G,B – 204,204,204) colored cube, and the corresponding $\mathbf{E}_{\text{multi-}n}$ at the $t^{\text{tick-}n}$ interval is denoted by \blacksquare colored cube (R,G,B – 129,129,129). Also, the existing cubes are denoted by \blacksquare (R,G,B – 57,188,40) colored cube, and the non-existing cubes are denoted by \blacksquare (R,G,B – 255,21,21) colored cubes.

4.3.3 Prototyping

The state conditions in terms of the $\mathbf{N}^{\text{check}}$ and the \mathbf{N}^{form} routines of all the Ψ_{tower} entities (namely, $\Psi_{\text{tower-cube}}$, $\Psi_{\text{tower-cubocta}}$, and $\Psi_{\text{tower-cube-cubocta}}$) that determine all the distinct **Stack** operations for all the $\mathbf{E}_{\text{multi}}$ **array** (and the individual \mathbf{E}_{cube} **array**, $\mathbf{E}_{\text{cubocta}}$ **array**, and $\mathbf{E}_{\text{cube-cubocta}}$ **array**) have been sufficiently demonstrated and illustrated in terms of the Ψ_{spawn} and Ψ_{cull} rules (as per fig. 4.18 and fig. 4.19). The illustrations show all possible conditions for the Ψ_{spawn} and Ψ_{cull} rules. However, similar to the computational stacking logic for the $\mathbf{CE}_{\text{cube-tower}}$, the computational stacking logic of the $\mathbf{CE}_{\text{cube-cubocta-tower}}$ is quite simple, and can be generalized as summarized below –

- If the cardinality of the \mathbf{N} consideration for an existing $\mathbf{E}_{\text{multi}}$ is equal to 2 or 3, the $\mathbf{E}_{\text{multi}}$ survives (i.e. not culled). If the cardinality is otherwise, the $\mathbf{E}_{\text{multi}}$ does not survive (i.e. culled).
- If the cardinality of the \mathbf{N} consideration for a non-existing cell is equal to 4, the $\mathbf{E}_{\text{multi}}$ is created (i.e. spawned). If the cardinality is otherwise, the $\mathbf{E}_{\text{multi}}$ is not created (i.e. not spawned).

As the $\mathbf{CE}_{\text{cube-cubocta-tower}}$ serves as a conceptual continuation of the $\mathbf{CE}_{\text{cube-tower}}$, the above rules which have a considerable precedence on the state conditions of the $\mathbf{CE}_{\text{cube-tower}}$, are supported by the entire series of procedural sequences (with case studies, simulation and prototyping) performed on the taxon, and can thus be thoroughly relied upon. However, owing to some significant modifications which

were made to the state conditions, the research necessitates rigorous evaluation, testing, prototyping and versioning if required. Also, as the **CE_{cube-cubocta-tower}** involves the introduction of a different morphology (as compared to the solitary assemblages of **E_{cube}**) in the form of **E_{cubocta}**, the prototyping becomes exceedingly indispensable for the robustness of this taxon. Thus, the following section details the functioning of the various means of prototyping undertaken to verify the **CE_{cube-cubocta-tower}**.

The initial prototyping of the **CE_{cube-tower}** was performed by means of a physical simulations workshop (conducted in collaboration with students and practitioners of the AEC industry) that generated several iterations by means of physical prototyping that involved participants constructing the **CE_{cube-tower}** manually while consulting an instruction manual that directed on the state conditions. However, as outlined in 4.2.4, the implementation of the physical simulations was quite counter-intuitive owing to the fact that the process of prototyping was quite cumbersome and highly susceptible to human error.

Thus, this method of prototyping (by means of physical prototyping that is built on a non-computational simulation methodology) was discontinued. However, the concept of involving collaboration with students and practitioners of the AEC industry had no major disadvantage (in fact it had a hidden advantage of testing the concept of **CE** in pedagogy and industry at the same instance) and was thus continued in a modified manner.

The workshop to test the **CE_{cube-cubocta-tower}** would have to follow a slightly different approach as compared to the one undertaken to test, evaluate, and taxonomize the **CE_{cube-tower}**. Since the **CE_{cube-cubocta-tower}** would have to be computationally simulated, the research would have to configure the computational semantics and processes before performing any tests (unlike as for the **CE_{cube-tower}** the algorithm was first tested and then configured as a computational process in the form of a UML Sequence Diagram). Thus, a UML Sequence Diagram was the natural progression in the prototyping stage for this taxon (as it was accommodating the changes

recommended by the previous one). As the $CE_{cube-cubocta-tower}$ serves as a conceptual progression to the $CE_{cube-tower}$, it would be quite intuitive to establish the UML Sequence Diagram based on the one for $CE_{cube-tower}$ (as per fig. 4.10 in 4.2.3). However, fig. 4.22 and fig. 4.23 differ from their predecessor by determining the role, interaction and runtime of all the ϵ entities and ψ entities as identified in the UML Class Diagram (illustrated in fig. 4.14). They jointly illustrate a UML Sequence diagram that determines the outcome of several different $CE_{cube-cubocta-tower}$.

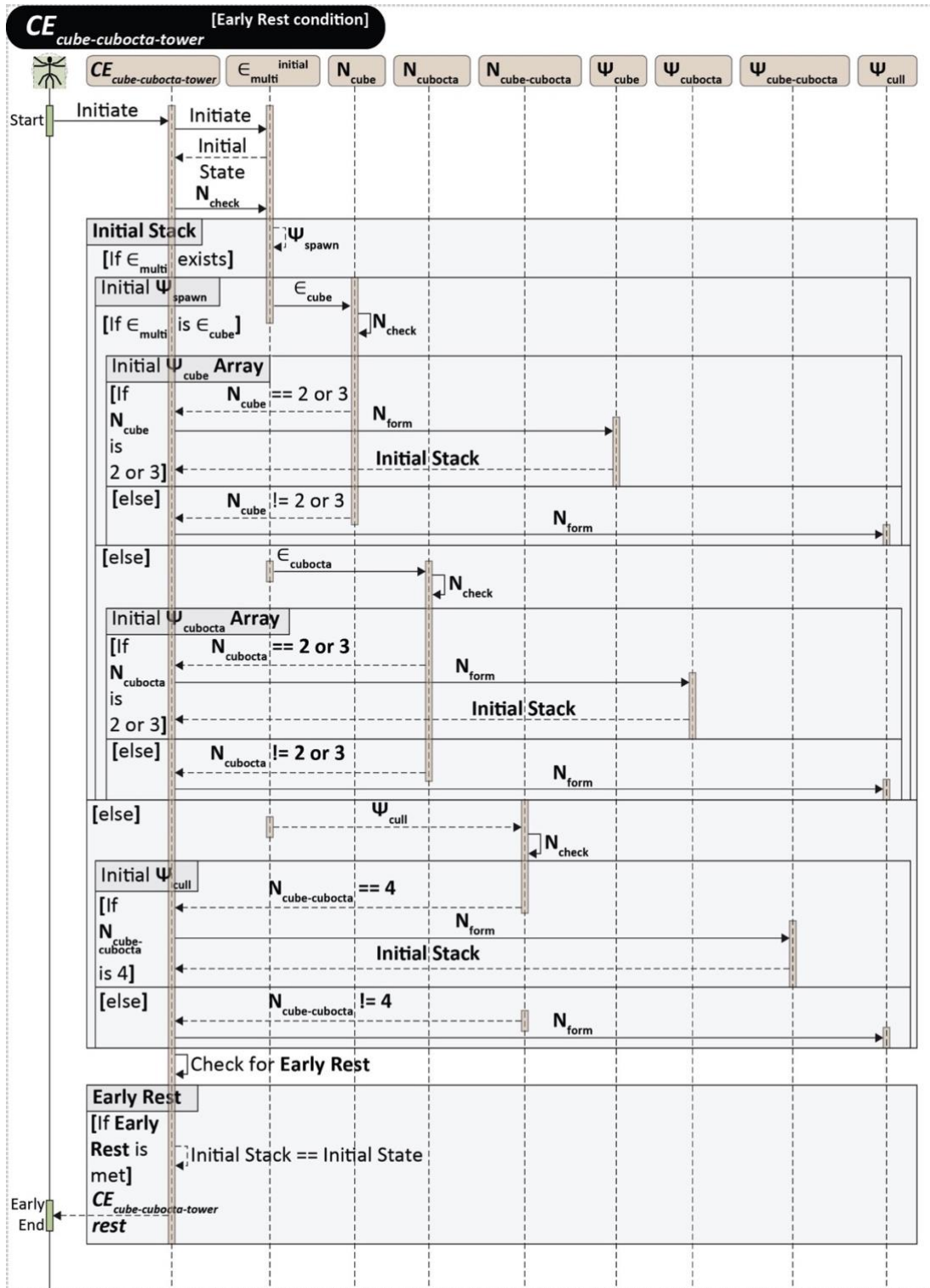


Figure 4.22 – The UML Sequence Diagram for a *CEcube-cubocta-tower* with the role, interaction and runtime of all the ϵ entities and ψ entities, and their dependencies for the Early Rest condition. Illustration and graphics by Author (May 2019).

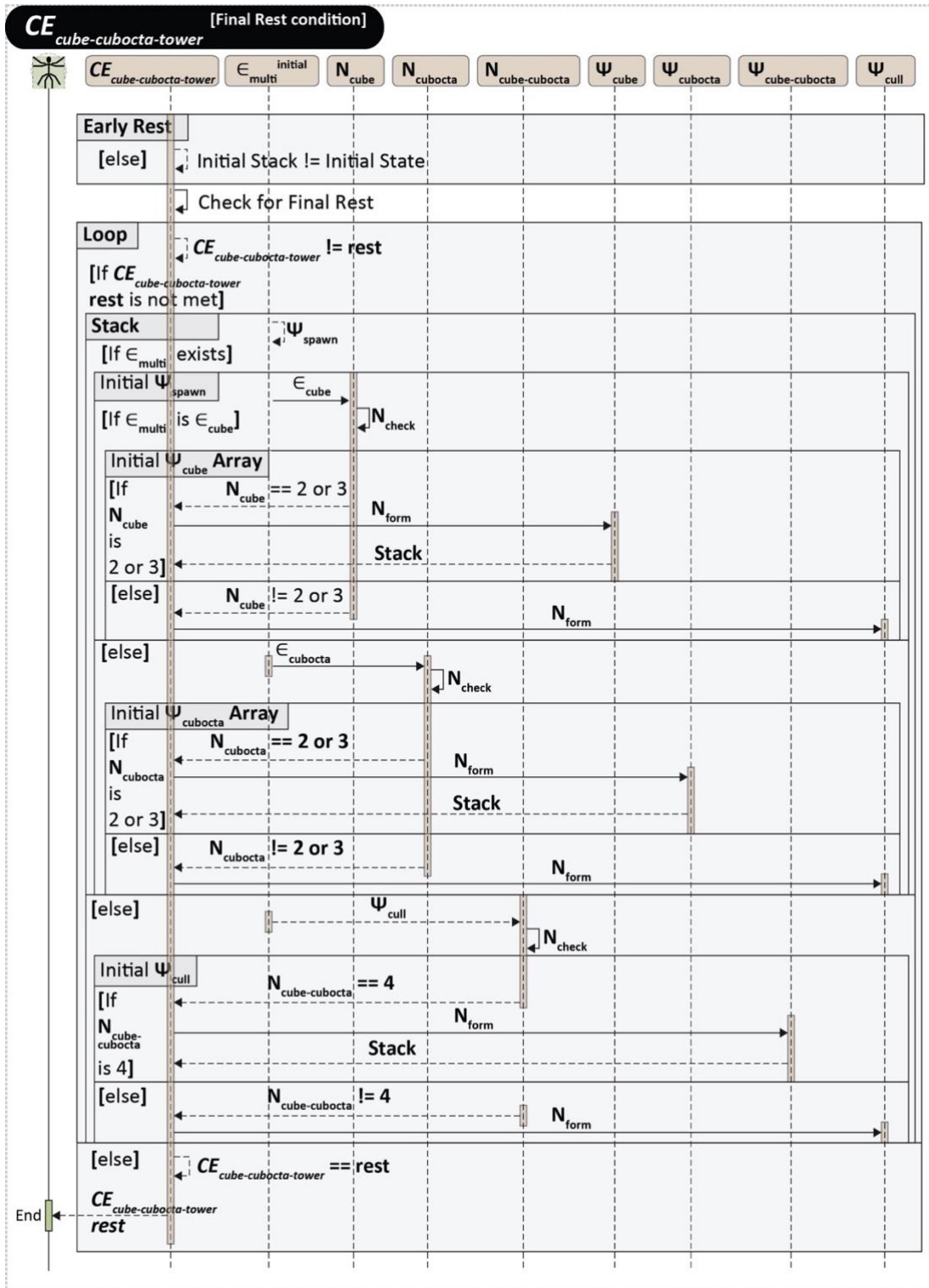


Figure 4.23 – The UML Sequence Diagram for a $CE_{\text{cube-cubocta-tower}}$ with the role, interaction and runtime of all the ϵ entities and Ψ entities, and their dependencies for the Final Rest condition. Illustration and graphics by Author (May 2019).

Similar to the UML Sequence Diagram for the $CE_{cube-tower}$, the one jointly illustrated for the $CE_{cube-cubocta-tower}$ in fig. 4.22 and fig. 4.23 has been generated with the following assumptions –

- The Actor would be the user who initiates the $CE_{cube-cubocta-tower}$.
- $CE_{cube-cubocta-tower}$ would be considered as one of the objects, as it has its own lifeline that represents the runtime of the entire algorithm, while the other objects in the diagram would be $\epsilon_{multi}^{initial}$, N_{cube} , $N_{cubocta}$, $N_{cube-cubocta}$, Ψ_{cube} , $\Psi_{cubocta}$, $\Psi_{cube-cubocta}$, and Ψ_{cull} in the chronological order of their use and application in the $CE_{cube-cubocta-tower}$.
- The Diagram also returns the $CE_{cube-cubocta-tower}$ as an **early rest** condition, if the **Initial State Array** and the **Initial Stack Array** are the same. If not, the remaining algorithm continues on a while loop until the $CE_{cube-cubocta-tower}$ **rest** condition (mentioned in 4.3.2) is met to end the algorithm.

Considering the UML Sequence Diagram, a Grasshopper definition was run inside the Rhino 7 interface (The definition labelled as $CE_{cube-cubocta-tower}$ has been appended to the Annexure – Definitions). Similar to the $CE_{cube-tower}$, the third-party components used to perform the algorithm functions are as followed –

- **Initiate** – To populate the random Initial array of $\epsilon_{multi}^{initial}$.
- **Array** – To generate ϵ_{cube} and $\epsilon_{cubocta}$ using the Lunchbox¹³⁵ plugin.
- **N_{check}** – To check the cardinality of Neighbouring cells using the **OR** logic gate.
- **Loop** – To generate a while loop using the Anemone¹³⁶ plugin.

¹³⁵ Lunchbox for Grasshopper (2012). Omaha, USA: Proving Ground Apps.

¹³⁶ Anemone 0.4 (2015). Poznań, Poland: Mateusz Zwierzycki.

Employing the Grasshopper definition, several test $CE_{cube-cubocta-tower}$ were prototyped in Rhino 7. As these were initial prototypes, the cardinality of $\epsilon_{multi}^{initial}$ was restricted to 3 and 4 ϵ_{multi} . Fig. 4.24 illustrates the outcomes of a few inputs for the aforementioned $CE_{cube-cubocta-tower}$ below. The figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), and the rest of the $CE_{cube-tower}$ thus denoted by ■ (R,G,B – 204,204,204) colored cube.

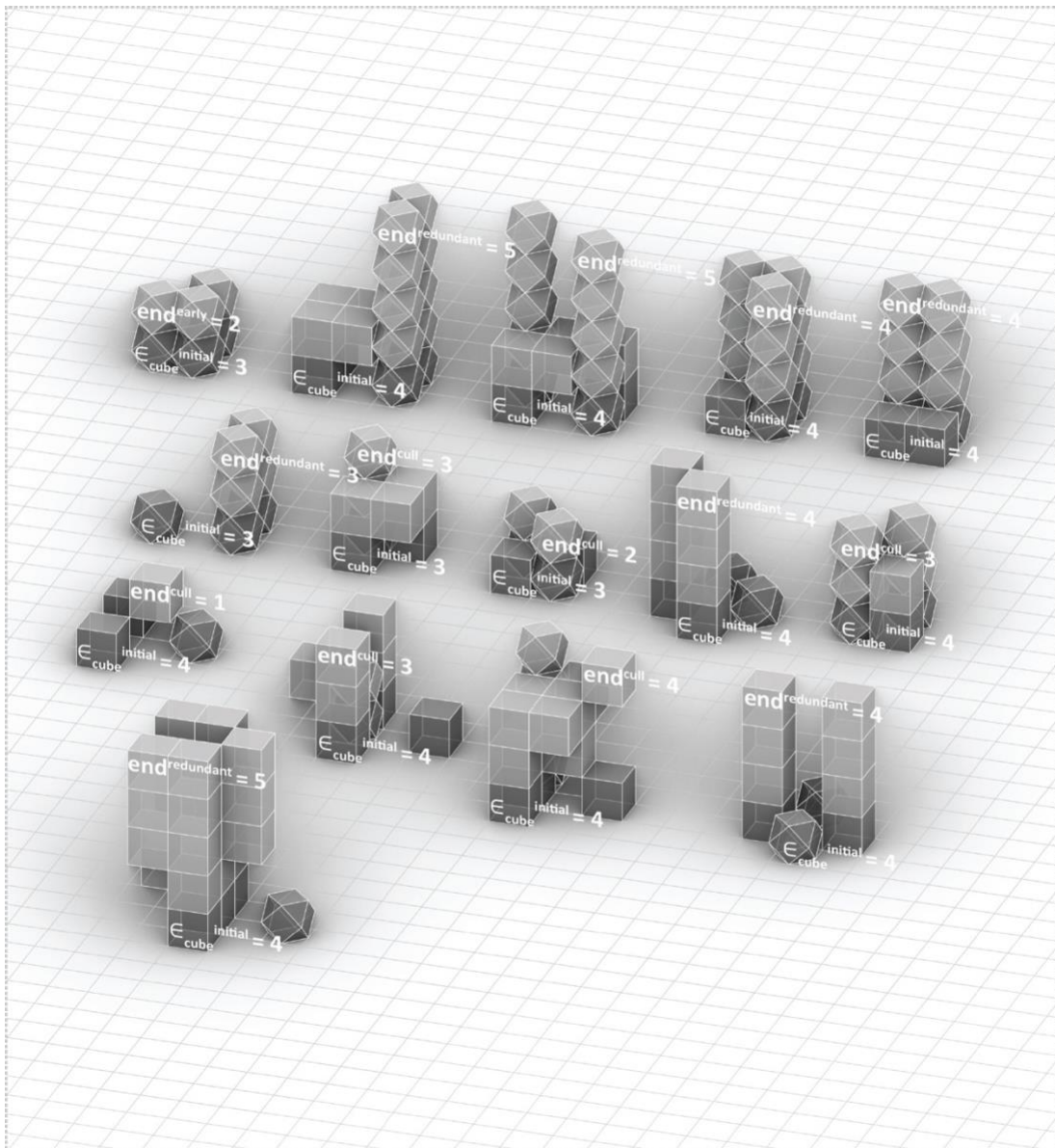


Figure 4.24 – Initial Prototypes of several $CE_{cube-cubocta-tower}$ with 3 and 4 $\epsilon_{cube}^{initial}$. Model, algorithm, Illustration and graphics by Author (Jan 2020).

After conducting the initial computational prototypes for the $CE_{cube-cubocta-tower}$, the procedural sequence was ready to be evaluated by a test group so as to conclude all the preliminary objectives (as in, case studies, simulation and prototyping) for the CE^{MESE} . Moreover, having already performed a considerable amount of basic prototyping, the research was well aware of the results and key $\epsilon_{multi}^{initial}$ **Array** that could be capitalized to generate interesting results.

Thus, the author conducted a student workshop which aimed at prototyping the CE^{MESE} (in the form of the $CE_{cube-cubocta-tower}$) by performing computational simulations to test, evaluate, taxonomize and if required, amend and update the $CE_{cube-cubocta-tower}$. The workshop, titled as '**Computation as a Design tool**' was conducted in July 2019 at RIT (Rajiv Gandhi Institute of Technology) in Kottayam, India. It was attended by 20 candidates, who were all students pursuing a BArch degree at RIT, Kottayam (20 students of the 7th and 9th semesters of the BArch course were selected on the basis of their personal interests in learning computational design).

For the testing, participants were first introduced to the concept of Computational Ecosystems and its research, and were provided with a lecture on Cellular Automata and its implementation in the research. Further, the participants were demonstrated several results that were obtained in the previous iteration of a similar workshop (i.e. the Designing ways of designing workshop conducted at IES, Mumbai – as elaborated in 4.2.3). As the participants were not well versed with using any computational design software, they were tutored on using different tools and functionalities with Rhino 7 and Grasshopper 3D that were relevant to the research on Computational Ecosystems. The participants were also encouraged to test the already established $CE_{cube-tower}$ before explicitly evaluating the $CE_{cube-cubocta-tower}$.

After establishing and upgrading their computational skills to a considerable level, the participants were individually tasked with testing the $CE_{cube-cubocta-tower}$. Although they were asked to test the algorithm with $\epsilon_{multi}^{initial}$ **Array** of 10 to 12, the participants were encouraged to experiment with higher number of $\epsilon_{multi}^{initial}$ **Array**.

After the successful trials of the $CE_{cube-cubocta-tower}$, Fig. 4.25 illustrates the outcomes of one of the tallest $CE_{cube-cubocta-tower}$ with $12 \epsilon_{cube}^{initial}$ that was generated during the workshop – ‘*Computation as a Design tool*’. The figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), and the rest of the $CE_{cube-tower}$ thus denoted by ■ (R,G,B – 204,204,204) colored cube.

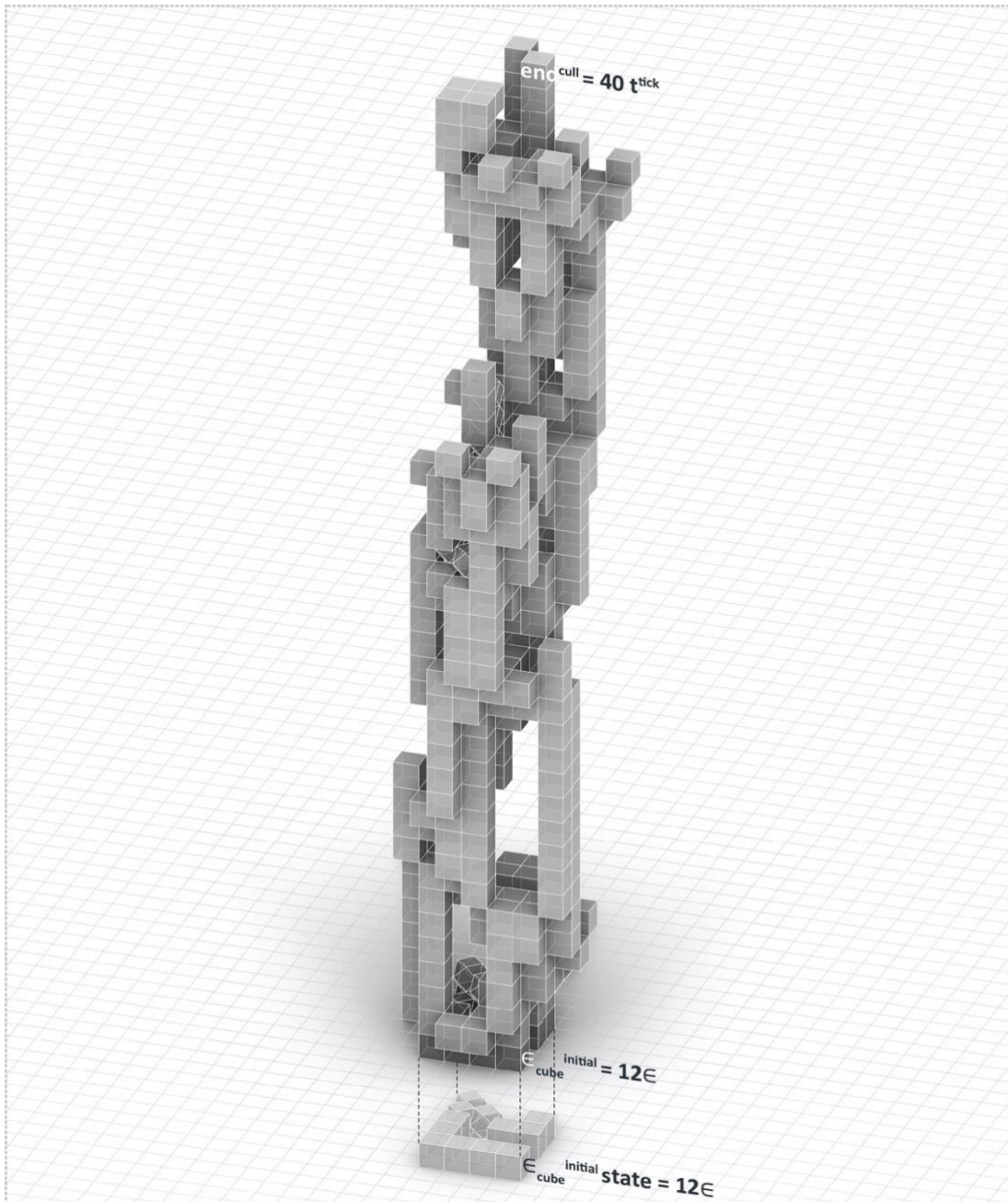


Figure 4.25 – Prototype of a selected $CE_{cube-cubocta-tower}$ with $12 \epsilon_{cube}^{initial}$ with a runtime of $40 t^{tick}$ before reaching the rest state. Model, algorithm, Illustration and graphics by Author (July 2019).

4.3.4 CE^{MESE}

Fig. 4.24 and fig. 4.25 sufficiently demonstrate the executions and prototyping of the $CE_{cube-cubocta-tower}$, however, (similar to the $CE_{cube-tower}$) more outcomes focusing on the taxonomy creation have been illustrated in the next chapter (5 | On the consequences of Computational Ecosystems) with further analysis comparing all the other taxa from the procedural sequences in chapter 6.

However, before continuing with the next taxa, certain observations need to be mentioned in this section to underline the incorporation of observations made in the previous section (as in 4.2.4). Also, a reflection on the performance of this procedural sequences and the primary objectives employed to execute it need to be mentioned to avoid any redundancies that could devalue the quality of the upcoming taxa. Also, owing to the fact that the next taxon (CE^{SEME}) serves as a conceptual equivalent, these observations are essential before starting the Case Studies for the CE^{SEME} .

Following are the changes made after considering the observations for the CE^{SESE} –

- The redundancies generated by performing physical prototyping workshops (that is through workshops where participants manually prototype computational ecosystems while following instruction manuals) were counteracted by conducting computational prototyping workshops (that is through workshops where participants computationally prototype computational ecosystems through computational design).
- This amendment had to be incorporated into the methodological workflow which had to be considerably modified to accommodate the unavailability of resources to conduct the workshop. For example, in the scenario of the CE^{SESE} the algorithm was determined by the results generated during the workshop – *Designing ways of designing* (as outlined in 4.2.3).

- However, in the scenario of the CE^{MESE} , the algorithm had to be synthesized, and tested on a preliminary basis before being considered for the computational prototyping with the participants of the workshop - **Computation as a Design tool** (as outlined in 4.3.3).

Following are the observations made for the procedural sequence CE^{MESE} , which should be considered in the next procedural sequences CE^{SEME} and CE^{MEME} –

- Out of the 15 iterations demonstrated in fig. 4.24 (as a combination of some of the possible iterations with 3 or 4 $\epsilon_{\text{multi}}^{\text{initial}}$) only 1 returned with an Early End condition. Although this observation has been made in the synthesis of the CE^{SESE} (as outlined in 4.2.4), the methodology was not changed.
- Although this number (which is 6.67% of the possible iterations) has reduced considerably as compared to its predecessor, as there was no deliberate modification made to the algorithm, this can be considered as a virtue of the combination of morphologies used.
- More strategies could be employed for the Ψ entities to make sure that the rule sets do not end up replicating the **Initial State** as the **Initial Stack**. This, could also be achieved by making sure that some context sensitive rule sets could be added to ensure a modified ground condition.
- The ϵ entities, which resemble Platonic and Archimedean shapes at the moment give the appearance of the direct continuation of a 3D Cellular Automata (as in a **CA** of the Conway Model expressed in 3D). Although, there are no issues with such a comparison at this stage, the results should more resemble a built-form that can be imagined in the structure of the world.

It would be crucial to include the above-mentioned observations in the upcoming taxa and perform the primary objectives in a more refined way.

4.4 Single Element Multiple Economies Ecosystem (CE^{SEME})

Similar to CE^{SESE} and CE^{MESE} , the conceptual framework of a CE^{SEME} has already been sufficiently illustrated in the previous chapters (introduced in 1.2.2, illustrated in fig. 1.8, defined in 3.2.2). Moreover, a Structural UML Diagram establishing the attributes and operations of a canonical version the $\mathbf{\epsilon}$ and $\mathbf{\Psi}$ has also been illustrated (in fig. 3.4 as part of sub section 3.2.4). However, like CE^{SESE} and CE^{MESE} , to actually construct, taxonomize and prototype this procedural sequence, the research would have to consider specific tangible parameters for the concept of a CE^{SEME} , which could be derived from a combination of real-life examples and constraints and the conceptual framework for the case studies of the already established CE^{SESE} and CE^{MESE} .

The CE^{SEME} also serves as an extension and continuation to the $CE_{cube-tower}$, primarily due to the multiplicity of the $\mathbf{\Psi}$ parameter. Moreover, the CE^{SEME} performs as a slight modification on the $CE_{cube-cubocta-tower}$ owing to the way the multiplicity was handled theoretically, semantically and computationally. Thus, to maintain the complexity in of the CE^{SESE} and CE^{MESE} , the Author has decided to continue with the conceptual example of an $SEME$ illustrated in 1.2.2. Use of the example scenario would be especially instrumental in methodically increasing the $\mathbf{\Psi}$ parameter to 2 (instead of the ambiguous multi) and using the canonical Structural UML Diagram. Thus, before specifically defining each parameter, the overall components of the CE^{SEME} can be –

- $\mathbf{\epsilon}$ – As illustrated in the example scenario, the $\mathbf{\epsilon}$ would be represented by hexahedrons or cubes, with unit dimensions.
- $\mathbf{\Psi}^1$ – As illustrated in the example scenario, the $\mathbf{\Psi}^1$ would be represented by structural stability in the form of axial loads.
- $\mathbf{\Psi}^2$ – As illustrated in the example scenario, the $\mathbf{\Psi}^2$ would be represented by buoyancy (as per the Archimedes principle).

4.4.1 Case Studies

Although serving as a conceptual successor to the CE^{SESE} , and by definition being able to serve as a continuation to the CE^{SESE} , the CE^{SEME} like CE^{MESE} which theoretically and semantically means that it can be treated as an ecosystem that is inhabited by one and only one predetermined species and this species is constructed, monitored, and governed (individually, categorically and collectively) by a combination of multiple rule sets, the CE^{SEME} does not serve as a direct sequel to the CE^{SESE} or as an alternative to the CE^{MESE} in terms of fulfilling the Procedural Sequences.

However, the CE^{SEME} has certain attributes related to the ϵ and Ψ that cannot be exhibited by either the CE^{SESE} or the CE^{MESE} due to their semantical constraints. The multiplicity of the Ψ parameter doesn't just imply that there would be two individual Ψ entities, but the order or preference within these two predetermined Ψ entities and their deployment in terms of the ϵ would unmistakably require a third Ψ entity that governs the sequence of the two. Thus, as per the requirements of the CE^{SEME} and the modifications already made to the assumptions of the example scenarios of CE^{SESE} and the CE^{MESE} , the following additions must be made to the example scenario and the assumptions stated in the last section –

- ϵ – Cubes, with unit dimensions, which can be programmed as sentient elements similar to the ϵ_{cube} , that are aware of their existence.
- Ψ^1 – A rule set that determines the stacking of ϵ_{cube} in the form of state conditions considering an overall structural stability in the form of axial loads.
- Ψ^2 – A rule set that maintains the buoyancy of ϵ_{cube} in the form of state conditions (as per the Archimedes principle).
- Ψ^{1-2} – A rule set that determines the order and preference of the Ψ^1 and Ψ^2 for every ϵ_{cube} in the form of state conditions.

Following the case study of the collective assemblages of the *Eciton hamatum* forming live bridges out of their own bodies (elaborated in 4.2.1, and illustrated in fig. 4.1) as a theoretical precedent for the CE^{SESE} and CE^{MESE} , the CE^{SEME} seeks a slightly different collective assemblage of fire ants.

It has already been established that this taxon relies on the existence of 3 Ψ parameters for the formation of a CE (namely, Ψ^1 , Ψ^2 , and Ψ^{1-2}) while exploring and imbibing the concept of buoyancy into the previously setup $CE_{eciton-bridge}$ (which establishes how a computational model could be made to perform simulations of the live bridges created by the fire ants). Thus, a study conducted on the collective assemblages of *Solenopsis Invicta* (a species of fire ants found extensively in Brazil) in the form of live rafts that float on water was examined and inferences were extrapolated as theoretical precedents for the CE^{SEME} .

Fig. 4.26 shown below demonstrates an Ant Raft formed by the collective assemblages of the aforementioned *Solenopsis Invicta* created by 500 ants.



Figure 4.26 – A raft of 500 fire ants, composed of a partially wetted layer of ants on the bottom and dry ants on top. Original Image from the research conducted by schools of Mechanical Engineering, Industrial and Systems Engineering and Biology of the Georgia Institute of Technology, GA, USA. Source: <https://www.pnas.org/content/pnas/108/19/7669.full.pdf>

The researchers simulated the creation of the ant rafts by dropping a hive of 3000 ants on different surfaces. *“When dropped on solid surfaces, the hive quickly disintegrated and the ants fled in all directions. However, when placed on the surface of water (water that is free of surfactants), the ants redistribute and reconnect themselves into raft”* (Mlot et al, 2011)¹³⁷. The study also found that, the raft reaches a stable equilibrium within several minutes, and *“at equilibrium, the rafts are pancake shaped, whereby a dry portion of the colony stands atop a monolayer of stationary ants”* (as shown in fig. 4.25). The spreading of the raft also resembles that of a drop of fluid, which directed the researchers to simulate the ant raft construction by considering the ants as a *“liquid continuum, and running the models of fluid dispersion and diffusion.”* The simulations of the ant raft construction result in the following understanding about the collective assemblages –

- As the same ants perform all roles required in the construction of the ant raft (even the role of the passengers), these collective assemblages are reversible.
- Considering that the ants maintain the integrity of the raft by communicating how many ants are aboard the ant raft (just like in the form of the ant bridges, the ants are aware of how many ants are crossing the bridge), the collective assemblages are self-aware, and always interpolating their physical properties such as shape, size and weight.

Thus, the study on the ant rafts and the observations on the simulations can be considered as a theoretical precedent for the CE^{SEME} . However, the ant raft caters to only one of the Ψ parameters, (i.e., Ψ^2 determining rule sets for Buoyancy of the ϵ_{cube}). Thus, the assumptions for the $CE_{eciton-bridge}$ could be appended with the above observation.

¹³⁷ Mlot, N. J., Tovey, C. A., Hu, D. L. (2011). Fire ants self-assemble into waterproof rafts to survive floods. In: *Proceedings of the National Academy of Sciences of the United States of America*. [online] Washington DC: PNAS, p 6. . Available at: <https://www.pnas.org/content/108/19/7669.full>

Hence a modified $CE_{ant-floating-bridge}$ could have the following modified components –

- ϵ – The ants. Acting as biotic, ambulatory agents that are self-aware of their physical properties such as their weight, weight-carrying capacity, movement speed, and gripping abilities. Thus, ϵ_{ants} .
- Ψ^1 – The bridging ability. Forming collective assemblages by means of connection techniques, span-depth ratios for optimum bridge structures, and load calculations (considering the live loads and dead loads). Thus, Ψ_{bridge} .
- Ψ^2 – The rafting ability. Forming collective assemblages by means of connection techniques, span-depth ratios for optimum floating structures, and load calculations (considering the live loads and dead loads). Thus, Ψ_{raft} .

However, as floating ant bridges don't really exist in nature, in the same way that the above assumption suggests, the 3rd Ψ entity which determines the priority rule sets for the ϵ_{ants} over forming a bridge or raft first, requires some more understanding of the simulation model. Moreover, as the CE^{SEME} serves as conceptual kin to the CE^{MESE} (owing to the fact that both the taxa involve multiplicity in either of the ϵ or Ψ parameters), the theoretical assumptions made for the CE^{MESE} can be further implemented in exemplifying the CE^{SEME} .

However, the CE^{MESE} treats the multiplicity of the ϵ parameters slightly differently. It does so by assuming that the same species of ants take up different roles to form the bridge, thus behaving as different types of elements serving different purposes in the construction of the $CE_{eciton-bridge}$ (as elaborated upon in 4.3.1). Thus, although the *Eciton hamatum* and the *Solenopsis invicta* are different ant species, for the sake of the simulation they can be treated as ambulatory agents which are capable of being governed by two different Ψ parameters – the Ψ_{bridge} and the Ψ_{raft} . While being simultaneously governed by the rule on which Ψ parameter to perform first.

Thus, 3rd Ψ entity, formerly identified as Ψ^{1-2} , can be further defined as followed –

- Ψ^{1-2} – The prioritization ability. Assessing the neighbourhood condition and the physical and structural properties of the Ψ_{bridge} and the Ψ_{raft} by means of density calculation of context and load calculations. Thus, $\Psi_{\text{bridge-raft}}$.

Consequently, the ϵ_{ants} parameter of the $CE_{\text{ant-floating-bridge}}$ would also have a few modifications to accommodate the variations done in the Ψ parameters. The updated ϵ parameter can be defined as followed –

- ϵ_{ants} – The ants. Acting as biotic, ambulatory agents that are self-aware of their physical properties such as their weight, weight-carrying capacity, movement speed, and gripping abilities for each of the two distinct Ψ entities - ϵ^{bridge} , and ϵ^{raft} .

Considering the fact that the ϵ parameter would now also have to be bifurcated into distinct ϵ entities to accommodate the Ψ parameters, the names of all the components would have to be amended. Thus, the new components of the updated $CE_{\text{ant-floating-bridge}}$ would be –

- $\epsilon_{\text{ants}}^{\text{bridge}}$ – These would be the Ants performing the Ψ_{bridge} .
- $\epsilon_{\text{ants}}^{\text{raft}}$ – These would be the Ants performing the Ψ_{raft} .
- Ψ_{bridge} – These would be the rule sets for $\epsilon_{\text{ants}}^{\text{bridge}}$.
- Ψ_{raft} – These would be the rule sets for $\epsilon_{\text{ants}}^{\text{raft}}$.
- $\Psi_{\text{bridge-raft}}$ – These would be the rulesets prioritizing either of the ϵ entities over the other.

Thus, like the CE^{MESE} , a CE^{SEME} would have multiple ϵ entities despite the singularity.

Thus, the similarities between a CE^{SEME} (derived from the example scenario and the CE^{SESE} and CE^{MESE}) and a $CE_{ant-floating-bridge}$ can be illustrated as shown in fig. 4.27 below.





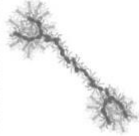


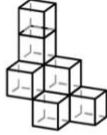
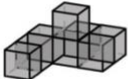
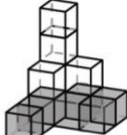
SEME					
$CE_{ant-floating-bridge}$					
	ϵ_{ants}^{bridge}	ϵ_{ants}^{raft}	ψ_{bridge}	ψ_{raft}	$\psi_{bridge-raft}$
CE^{SEME}					
	ϵ_{cube}^{tower}	ϵ_{cube}^{raft}	ψ_{tower}	$\psi_{floating}$	$\psi_{floating-tower}$

Figure 4.27 – Comparing the CE^{SEME} and a $CE_{ant-floating-bridge}$. Illustration and graphics by Author (May 2019).

Thus, based on the analogies illustrated in fig. 4.27 and the conceptual semantics established for a $CE_{ant-floating-bridge}$, The CE^{SEME} could be based on the already established $CE_{cube-tower}$ (as defined in 4.2.1) and $CE_{cube-tower}^2$ (as defined in 4.3.1). However, in case of the CE^{SEME} the constituent ϵ entities and ψ entities, would have to be defined in relation to the updated $CE_{ant-floating-bridge}$. Therefore, the CE^{SEME} is defined as followed –

- ϵ entities – or ϵ^{SEME} entities will be considered as computational substitutes to the two ϵ entities established for the $CE_{ant-floating-bridge}$. Although the functionalities and structural properties of the ϵ_{ants}^{bridge} , and $\epsilon_{ants}^{floating}$ do not directly translate to the ϵ^{SEME} entities, they contain the same semantics as established to be the components of the $CE_{ant-floating-bridge}$. Thus, continuing on the example scenario, the ϵ^{SEME} entities can be further bifurcated as ϵ_{cube}^{tower} , and $\epsilon_{cube}^{floating}$. The specific properties of each entity, which are relevant for simulation, will be defined in the next section (4.3.2), but the general properties of all the ϵ^{SEME} entities can be as followed –

- All the ϵ^{SEME} entities should be considered as static agents that can be spawned or culled by the CE^{SEME} at the expense of the Ψ entities.
- All the ϵ^{SEME} entities should be made to be self-aware of themselves, as in identifying and quantifying their positions, weights, and bounds.
- All the ϵ^{SEME} entities should also be made to be self-aware of their surroundings and their physical properties, like cardinality of neighbours, array of neighbours, and distance from the initial plane.
- Ψ entities – or Ψ^{SEME} entities will be considered as computational substitutes to the Ψ entities established for the $CE_{ant-floating-bridge}$. Although the functionalities and structural properties of the Ψ_{bridge} , Ψ_{raft} , and $\Psi_{bridge-raft}$ do not directly translate to the Ψ^{SEME} entities, they have the same semantics. Thus, the Ψ^{SEME} entities can be further trifurcated as Ψ_{tower} , $\Psi_{floating}$, and $\Psi_{floating-tower}$. The properties of all the Ψ^{SEME} entities can be defined as –
 - Contrary to the $CE_{cube-tower}$, and the $CE_{cube-cubocta-tower}$ the CE^{SEME} would have the distinct Ψ^{SEME} entities performing a tower stacking logic and a floating raft logic in unison.
 - They should be however, designed as a two-state rationale for all the ϵ^{SEME} entities. While, Ψ_{tower}^{SEME} could be derived from Ψ_{tower}^{SESE} (4.2.1), the other two Ψ^{SEME} entities – $\Psi_{floating}$, and $\Psi_{floating-tower}$ would require the understanding of different structural logics derived from the Archimedes principle of buoyancy.
 - However, on the whole, Ψ^{SEME} entities should seek an equilibrium state that spawns and culls ϵ^{SEME} entities considering if the tower is under-structured, overstructured, sinking or over-buoyant, thus maintaining a reciprocal coupling.

After establishing and defining the ϵ^{SEME} entities and Ψ^{SEME} entities, the CE^{SEME} could be defined under the computational guidelines of **CA** (as elaborated in 3|) as –

- The CE^{SEME} due to its specific combination of ϵ^{SEME} entities and Ψ^{SEME} entities can be termed as $CE_{cube-floating-tower}$, and will be defined as a functioning ecosystem constituting of sentient, context aware cubes as ϵ^{SEME} entities that interact with each other considering the rule sets assigned by Ψ^{SEME} entities to spawn or cull the ϵ^{SEME} entities until the runtime of the $CE_{cube-floating-tower}$.
- As the $CE_{cube-floating-tower}$ will perform while considering two distinct Ψ^{SEME} entities, the 3rd Ψ^{SEME} entity will operate as a determiner of partial runtimes for the distinct equilibria of the $CE_{cube-floating-tower}$. Moreover, the $\Psi_{floating}$ would perform while considering the ϵ^{SEME} entities in a specific context. However, one and only one CE that is $CE_{cube-floating-tower}$ would be performed for its entire runtime.
- Although derived from the hypothetical amalgamation of intricate bio-inspired behavior of army ants constructing living bridges for foraging trails, and fire ants constructing living rafts to avoid drowning, the $CE_{cube-cubocta-tower}$ is a quite complex combination of vertical stacking algorithm and a floating raft algorithm for cubes. Therefore, the rule sets for the Ψ^{SEME} entities can be adapted from the Conway model of **CA** (as per 2.3.2). However, these will be considerably modified.
- The Ψ^{SEME} entities would differ considerably from the Conway model of **CA** in the parameter of time. That is, while the new ϵ^{SEME} entities will be spawned or culled on the upper levels in the 3D grid for the Ψ_{tower} entity, the $\Psi_{floating}$ entity would direct spawning and culling ϵ^{SEME} entities on the XY plane.

Considering all the above assumptions, definitions, illustrations and examples for the $CE_{cube-floating-tower}$, the simulations for this taxon can now be performed.

4.4.2 Simulations

However, just like the CE^{SESE} , and the CE^{MESE} , the canonical CE^{SEME} UML Class Diagram (as per fig. 3.4) for the $CE_{cube-floating-tower}$ has to be reassessed and repurposed. Fig. 4.28, as shown below, illustrates the modifications to the canonical version.

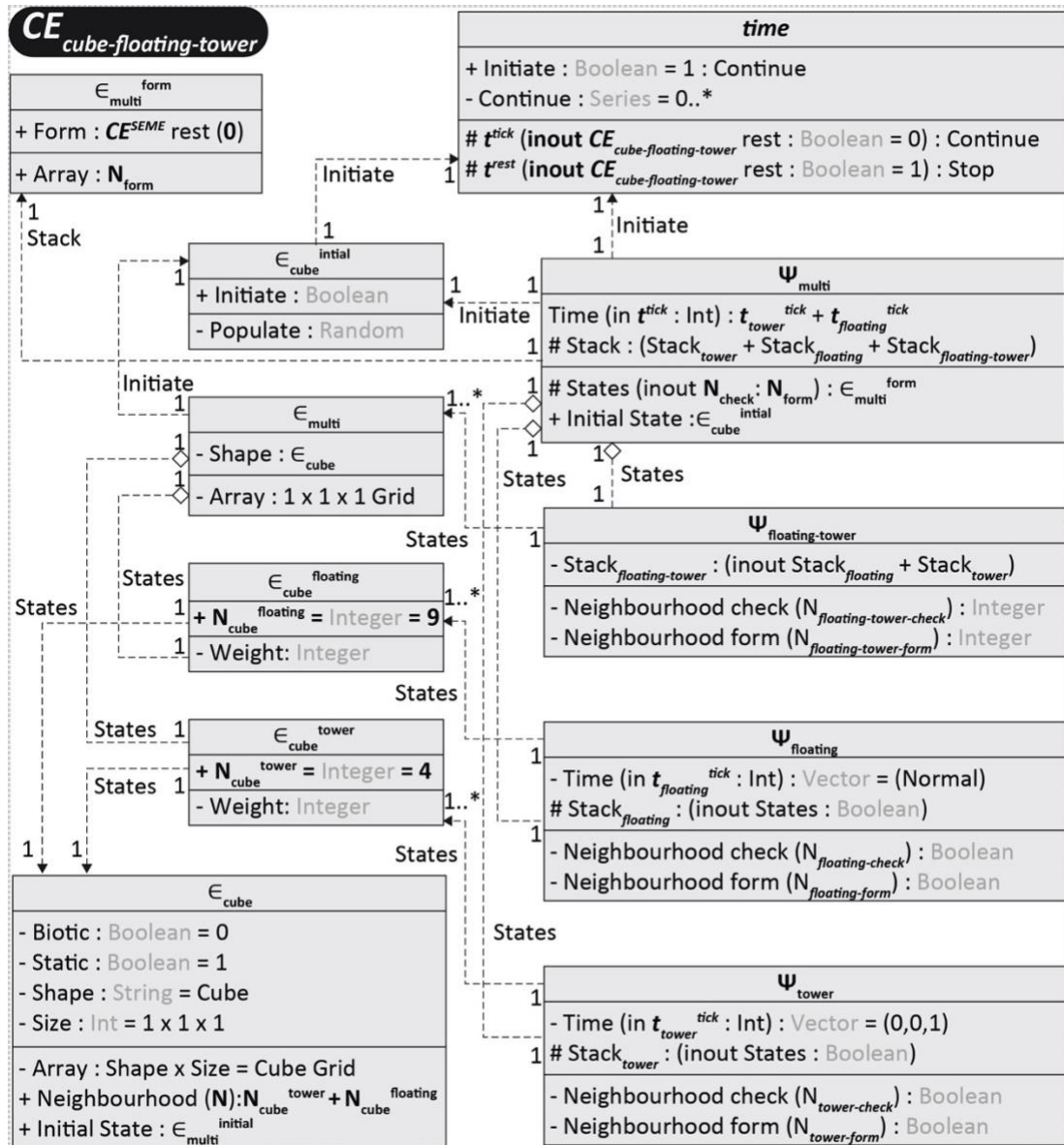


Figure 4.28 – Modifications made to the canonical CE^{SEME} UML Diagram to accommodate the parameters required for establishing the $CE_{cube-floating-tower}$. Illustration and graphics by Author (May 2019).

Elaborating on the terminologies introduced in fig. 4.28, the attributes and operations of the classes can be defined as followed –

- t^{tick} – The periodic increment of time, ensuring that the $CE_{cube-floating-tower}$ runs.
- t^{rest} – If the condition $CE_{cube-floating-tower}$ **rest** is fulfilled, the t^{rest} will be activated. This means that the time increment will stop and the $CE_{cube-floating-tower}$ will be outputted to the user.
- $CE_{cube-floating-tower}$ **rest** – Is the situation where an ϵ_{multi}^{form} **array** for the last 3 t^{tick} intervals is equal or repetitive.
- ϵ_{multi}^{form} **array** – Is the collective assemblage of ϵ_{multi} for a certain t^{tick} interval.
- ϵ_{multi}^{form} – Is the operation for the formation of a combination of ϵ_{cube}^{tower} and $\epsilon_{cube}^{floating}$ depending on the N_{check} and N_{form} routines for the Ψ_{tower} , $\Psi_{floating}$, and $\Psi_{tower-floating}$.
- N_{form} – Is the binary inputs for the cells to be either spawned (input = 1) or culled (input = 0) depending on the N_{check} conditional.
- N_{check} – Is the binary output from the cells of the distinct neighbourhoods (N) to check if the cells in the N of the previous t^{tick} interval are either spawned (input = 1) or culled (input = 0).
- N – The neighbourhood of the ϵ_{multi} in question (as in every ϵ_{cube}^{tower} and $\epsilon_{cube}^{floating}$) depending on how many neighbouring ϵ_{multi} from the previous t^{tick} interval are to be considered to determine the N_{check} and N_{form} routines that eventually determine the **Stack** operation.

Thus, determining the N becomes the most important step before considering to explain the other concepts required to run the $CE_{cube-floating-tower}$.

Similar to the computational environment and the revised infinite 3D Square grid that was considered for the $CE_{cube-tower}$ (refer 4.2.2) and the $CE_{cube-cubocta-tower}$ (refer 4.3.2), in case of the $CE_{cube-floating-tower}$ as well, the modelling has been performed in Rhino 7 (as per 3.3), the programming has been performed in Grasshopper for Rhino 7 (as per 3.3) and visualizations have been performed in V-Ray for Rhino (as per 4.1).

Although most of the physical properties required for the $CE_{cube-floating-tower}$ are similar to those of the previously established, there is a complication that arises from the multiplicity of the Ψ entities. The Ψ^2 and Ψ^{1-2} entities for the $CE_{cube-floating-tower}$ are rule sets to consider the state of equilibrium for the E_{multi} determined by their buoyancy. This directs the formerly considered computational environment - ***hypothetical revised infinite 3D Square grid*** to be comprised of a certain material. Thus, to evaluate and govern the buoyancy parameters the computational environment shall be considered as made of two different materials. However, these two materials shall have to expand the ***revised infinite 3D Square grid*** beyond the bounds of the ***+Z Axis***. Thus, the computational environment for the $CE_{cube-floating-tower}$ shall be the entire ***3D Cube Grid*** (that is, extending on all 3 axes without any bounds). However, the domain of the materials shall be precisely defined, where one does not interact with the other. Their distinctions are as followed:

- ***Air*** – This shall be the atmosphere populating the ***3D Cube Grid*** on the ***+Z Axis***. It shall not experience any lateral forces caused by wind or any atmospheric friction. It will be considered as a computational environment for the Ψ_{tower} and thus will affect the $N_{tower-check}$ and $N_{tower-form}$ routines.
- ***Water*** - This shall be a hypothetical infinite water body populating the ***3D Cube Grid*** on the ***-Z Axis***. It shall not contain the presence of any surfactants or other solvents. It shall also not contain any lateral forces caused by a water current or any kind of turbulence. It will be considered as a computational environment for the $\Psi_{floating}$ and thus will considerably affect the $N_{floating-check}$ and $N_{floating-form}$ routines.

Thus, considering the revised computational environment and its constituent materials, fig. 4.29 illustrates the bifurcation of the environment and the population of the two ϵ entities.

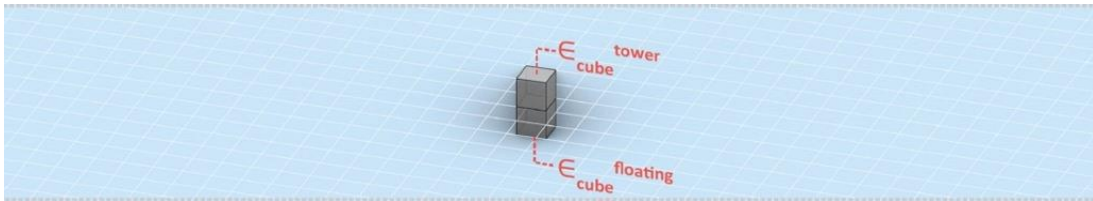


Figure 4.29 – Bifurcation of the environment and the population of the two ϵ entities. Model, Illustration and graphics by Author (May 2019).

The N_{tower} considered for the spawning or culling of the ϵ_{cube}^{tower} remains unchanged from the assumptions made in the case of the $CE_{cube-tower}$ and the $CE_{cube-cubocta-tower}$ previously (in 4.2.2 and 4.3.2 respectively). Thus, fig. 4.30 illustrates the N_{tower} for a ϵ_{cube}^{tower} .

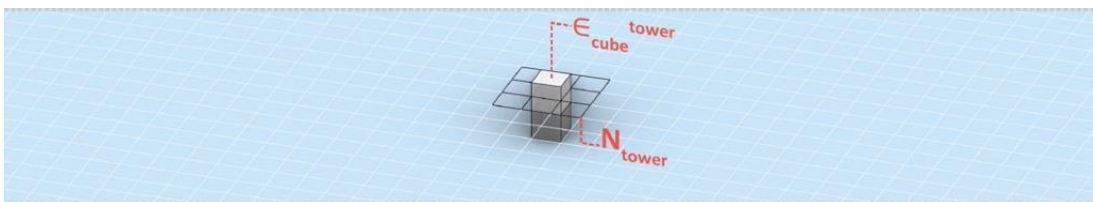


Figure 4.30 – An ϵ_{cube}^{tower} surrounded by the 8 possible N_{tower} . Model, Illustration and graphics by Author (May 2019).

However, the $N_{floating}$ considered for the spawning or culling of the $\epsilon_{cube}^{floating}$ operates in water, and thus requires to be illustrated as shown in fig. 4.31.

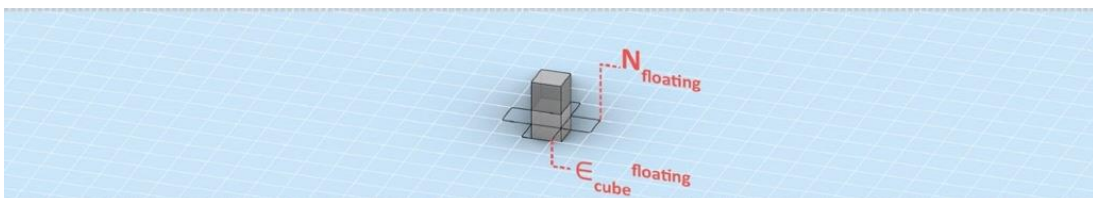


Figure 4.31 – An $\epsilon_{cube}^{floating}$ surrounded by the 4 possible $N_{floating}$. Model, Illustration and graphics by Author (May 2019).

The concept of structural stability considered and assumed in the construction of the rule sets of Ψ_{tower} has been already sufficiently explained and illustrated (in 4.2.1). Moreover, the concept has also been evaluated and validated in the '*Designing ways of Designing*' workshop (as explained and illustrated in 4.2.3). However, the Ψ_{tower} and the consequent rules assumed for structural stability have been derived for a condition without any atmosphere, and here, in the case of $CE_{\text{cube-floating-tower}}$ a medium of atmosphere has already been established as the partial computational environment. Despite these considerations, however, the same previous assumptions of Ψ_{tower} (setup in 4.2.1 and 4.2.3) shall be followed in the case of the $CE_{\text{cube-floating-tower}}$. The assumptions for the Ψ_{floating} on the other hand, have to be made in relation to the Ψ_{tower} so that the CE^{SEME} can perform all the Ψ entities symbiotically. As already established in the example scenario and the $CE_{\text{cube-floating-tower}}$, the Ψ_{floating} shall consist of rule sets that eventually control the spawning or culling of the $E_{\text{cube}}^{\text{floating}}$ to maintain an equilibrium state for the entire $CE_{\text{cube-floating-tower}}$. In other words, the Ψ_{floating} shall add or remove $E_{\text{cube}}^{\text{floating}}$ to ensure that the $CE_{\text{cube-floating-tower}}$ does not sink. To do this, the following assumptions can be made –

- All the operations related to the spawning or culling of the $E_{\text{cube}}^{\text{floating}}$ in the context of the $CE_{\text{cube-floating-tower}}$ shall be made in the **Water**. That means, only the N_{floating} can be considered for the N_{form} routine.
- The $E_{\text{cube}}^{\text{floating}}$ and $E_{\text{cube}}^{\text{tower}}$ shall be considered as hollow cubes made of unit mass and unit volume, such that, if one $E_{\text{cube}}^{\text{floating}}$ is solitarily dropped in water, it shall float while having exactly 10% of its volume immersed.
- However, if more $E_{\text{cube}}^{\text{tower}}$ are added (by the virtue of the Ψ_{tower}), the resultant array of E_{multi} shall never have more than 20% of the collective volume of the $E_{\text{cube}}^{\text{floating}}$ immersed in the water.
- In case the volume of $E_{\text{cube}}^{\text{floating}}$ immersed in the water is more than 20%, more $E_{\text{cube}}^{\text{floating}}$ shall be added to the N_{floating} by means of the N_{form} routine.

Thus, considering the aforementioned assumptions, an array of $n\epsilon_{\text{cube}}^{\text{tower}}$ ($n\epsilon_{\text{cube}}^{\text{tower}}$ elements) will always require $n\epsilon_{\text{cube}}^{\text{floating}}$ ($n\epsilon_{\text{cube}}^{\text{floating}}$ elements) to make sure that the $CE_{\text{cube-floating-tower}}$ floats in such a way that no more than 20% of the collective volume of the $\epsilon_{\text{cube}}^{\text{floating}}$ is immersed in the water. Moreover, since the $\epsilon_{\text{cube}}^{\text{tower}}$ and the $\epsilon_{\text{cube}}^{\text{floating}}$ are morphologically same, the $CE_{\text{cube-floating-tower}}$ will basically be populating hollow cubes both vertically (to make a tower by following Ψ_{tower}) and horizontally (to ensure that the tower floats by following Ψ_{floating}). To demonstrate this, fig. 4.32 shown below takes the example of a specific $\epsilon_{\text{multi}}^{\text{initial}}$ array, and documents implementation of the Ψ_{floating} assumptions made above.

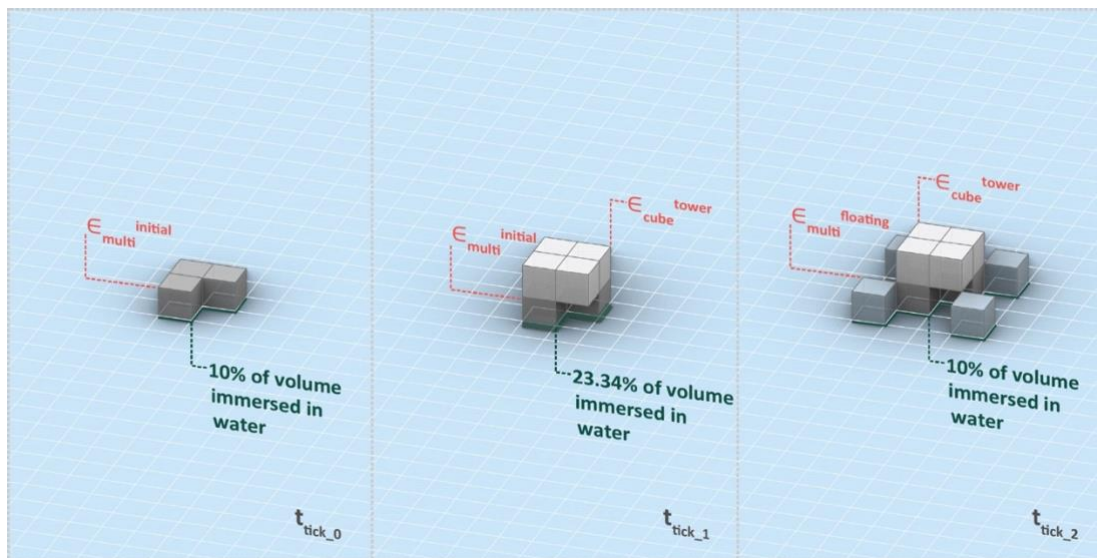


Figure 4.32 – Example implementation of $\Psi_{\text{floating-tower}}$ for the runtime of a $CE_{\text{cube-floating-tower}}$ with 3 $\epsilon_{\text{multi}}^{\text{initial}}$ elements. Model, Illustration and graphics by Author (May 2019).

As illustrated in fig. 4.32 above, at t^{tick_0} interval the array of three $\epsilon_{\text{multi}}^{\text{initial}}$ elements (with 10% of their volume immersed in water) employs a typical Ψ_{tower} rule set and spawns four more $\epsilon_{\text{cube}}^{\text{tower}}$ at the t^{tick_1} interval thus making a total of seven $\epsilon_{\text{multi}}^{\text{initial}}$ elements in the $CE_{\text{cube-floating-tower}}$. Without the implementation of the Ψ_{tower} rule set at the t^{tick_1} interval, the $CE_{\text{cube-floating-tower}}$ would still float, albeit with 23.34% of its $\epsilon_{\text{multi}}^{\text{initial}}$ elements immersed in water. This is not allowed as per the assumptions, and thus, at t^{tick_2} interval, four new $\epsilon_{\text{cube}}^{\text{floating}}$ are spawned to ensure that only a maximum of 20% of the $\epsilon_{\text{multi}}^{\text{initial}}$ and the $\epsilon_{\text{cube}}^{\text{floating}}$ elements is immersed in water.

Although the assumptions are to ensure that the $CE_{cube-floating-tower}$ floats, they can't be considered as the $\Psi_{floating}$ rulesets. Because, to ensure the spawning and culling of $E_{cube}^{floating}$ specific state conditions would be required, but the assumptions merely provide the number of elements to be spawned (to ensure that the $CE_{cube-floating-tower}$ floats at the predetermined level). The above assumptions can however be considered as the $\Psi_{floating-tower}$ rule sets, because the assumptions examine the results of the Ψ_{tower} rule and then direct the $\Psi_{floating}$, thus serving as a prioritizing junction between the two. The $\Psi_{floating-tower}$ rule sets also serve as a time buffer, as it halts to compute how many $E_{cube}^{floating}$ are required to be spawned. The $\Psi_{floating-tower}$ serving as a junction and a buffer between the Ψ_{tower} and the $\Psi_{floating}$ is quite crucial as the neighbourhood considerations of the Ψ_{tower} and the $\Psi_{floating}$ are quite distinct.

Similar to its previous applications for the $CE_{cube-tower}$ and the $CE_{cube-cubocta-tower}$, the state conditions of the $CE_{cube-floating-tower}$ will also be determined by employing the Conway Model of CA (as explained in 2.3.2). Thus, considering the $CE_{cube-tower}$ and the $CE_{cube-cubocta-tower}$ as precedence, and deriving from the Conway Model of CA , the state conditions for the $CE_{cube-floating-tower}$ can be established as followed –

- Ψ_{tower} - Every $E_{cube-n+1}^{tower}$ at the $t^{tick-n+1}$ interval interacts with the N_{cube}^{tower} of its corresponding (or in this case preceding) E_{cube-n}^{tower} at the t^{tick-n} interval, thus performing $N^{tower-check}$. For the $t^{tick-n+1}$ interval, it then performs the N^{form} routine based on the following conditions –
 - Any existing (spawned) E_{cube-n}^{tower} with 2 or 3 existing E_{cube-n}^{tower} in its N_{cube}^{tower} continues to exist and is not culled at the $t^{tick-n+1}$ interval, as if it was being perfectly supported by its counterparts on the floor below.
 - Any non-existing (culled) E_{cube-n}^{tower} with 3 existing E_{cube-n}^{tower} in its N_{cube}^{tower} is spawned for the $t^{tick-n+1}$ interval, as if it was being perfectly supported by its counterparts on the floor below.

- All other existing $\mathbf{E}_{\text{cube-}n}^{\text{tower}}$ at the $t^{\text{tick-}n}$ interval (those with other than two or three neighbours in their $\mathbf{N}_{\text{cube}^{\text{tower}}}$) stop existing at their corresponding $t^{\text{tick-}n+1}$ interval. Similarly, all other non-existing $\mathbf{E}_{\text{cube-}n}$ at the $t^{\text{tick-}n}$ interval (those with other than three neighbours in their $\mathbf{N}_{\text{cube}^{\text{tower}}}$) remain nonexistent at the $t^{\text{tick-}n+1}$ interval.
- $\Psi_{\text{floating-tower}}$ - At every $t^{\text{tick-}n+1}$ interval, the cardinality of $\mathbf{E}_{\text{cube-}n}^{\text{tower}}$ elements that are newly spawned and those that are not culled is measured, and the total number of new elements $\mathbf{E}_{\text{cube-}n+1}^{\text{floating}}$ elements is determined by –
 - Reducing the cardinality of the $\mathbf{E}_{\text{cube-}n}^{\text{tower}}$ elements in the $t^{\text{tick-}n}$ interval from the cardinality of the $\mathbf{E}_{\text{cube-}n}^{\text{tower}}$ elements in the $t^{\text{tick-}n+1}$ interval.
- Ψ_{floating} - Every $\mathbf{E}_{\text{cube-}n+1}^{\text{floating}}$ at the $t^{\text{tick-}n+1}$ interval interacts with the $\mathbf{N}_{\text{cube}^{\text{floating}}}$ of its corresponding (or in this case preceding) $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ at the $t^{\text{tick-}n}$ interval, thus performing $\mathbf{N}^{\text{floating-check}}$. For the $t^{\text{tick-}n+1}$ interval, it then performs the \mathbf{N}^{form} routine based on the following conditions –
 - Any non-existing (culled) $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ with no existing $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ in its $\mathbf{N}_{\text{cube}^{\text{floating}}}$ is spawned for the $t^{\text{tick-}n+1}$ interval, as if it was being perfectly supported by its counterparts on the edge of the cube, thus leaving the faces open for access.
 - Any non-existing (culled) or existing (spawned) $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ with any number of existing $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ in its $\mathbf{N}_{\text{cube}^{\text{floating}}}$ remains culled or is culled for the $t^{\text{tick-}n+1}$ interval, as if it was blocking access, and thus would not sufficiently provide buoyancy.

The $\mathbf{CE}_{\text{cube-floating-tower}}$ thus continues until the $\mathbf{CE}_{\text{cube-floating-tower}}$ **rest** condition is met, after which the \mathbf{CE} is returned to the user as an iteration.

The Spawn and Cull conditions for the Ψ_{tower} of the $CE_{cube-floating-tower}$ shall be considered to be similar to those in the $CE_{cube-tower}$ (as shown in fig. 4.7 and fig. 4.8) and thus need not be illustrated again. Moreover, the $\Psi_{floating-tower}$ does not have any Spawn and Cull conditions, as it merely states an integer value that determines the total number of $\epsilon_{cube}^{floating}$ elements to be spawned to ensure that the $CE_{cube-floating-tower}$ actually floats. Before following the next primary objective, and heading over to the prototyping stage for this procedural sequence, however, the $\Psi_{floating}^{spawn}$ and $\Psi_{floating}^{cull}$ need to be illustrated and highlighted for the $CE_{cube-floating-tower}$.

Fig. 4.33 thus demonstrates all the $\Psi_{floating}^{spawn}$ and $\Psi_{floating}^{cull}$ conditions for every possible state of array in the $N_{cube}^{floating}$ of the $\epsilon_{cube}^{floating}$ in question. Moreover, since the $\Psi_{floating}$ operates after the Ψ_{tower} and the $\Psi_{floating-tower}$ have concluded, the examples shown below already have the ϵ_{cube}^{tower} in place.

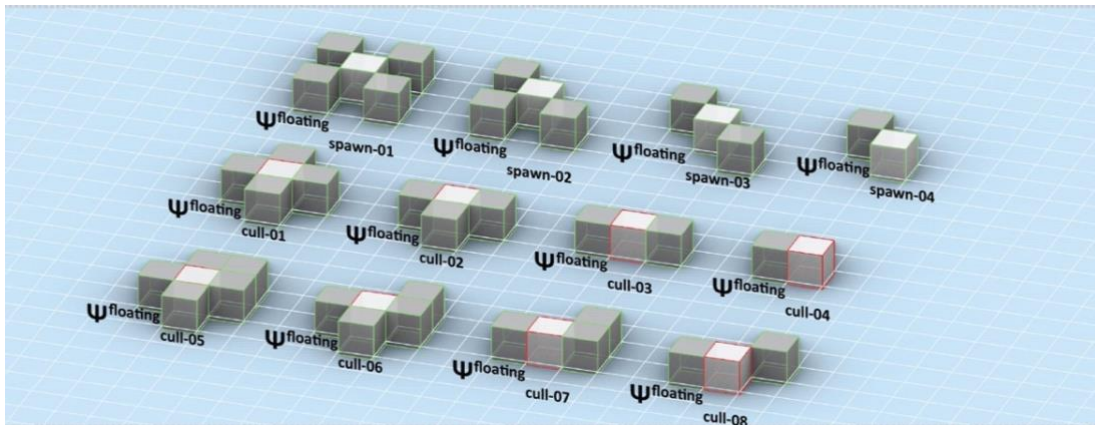


Figure 4.33 – $\epsilon_{cube-n}^{floating}$ array and their corresponding $\epsilon_{cube-n+1}^{floating}$ array considering the state conditions for the existing and the non-existing states of $\epsilon_{cube}^{floating}$ array. Model and graphics by Author (May 2019).

The $\epsilon_{cube-n+1}^{floating}$ at the $t^{tick-n+1}$ interval is thus denoted by \blacksquare (R,G,B – 204,204,204) colored cube, and the corresponding $\epsilon_{cube-n}^{floating}$ at the t^{tick-n} interval is denoted by \blacksquare colored cube (R,G,B – 129,129,129). Also, the existing cubes are denoted by \blacksquare (R,G,B – 57,188,40) colored cube, and the non-existing cubes are denoted by \blacksquare (R,G,B – 255,21,21) colored cubes.

4.4.3 Prototyping

The state conditions in terms of the $\mathbf{N}^{\text{check}}$ and the \mathbf{N}^{form} routines of all the Ψ_{tower} , Ψ_{floating} , and $\Psi_{\text{floating-tower}}$ that determine all the distinct **Stack** operations for all the $\mathbf{E}_{\text{multi}}$ array have been sufficiently demonstrated and illustrated in terms of the Ψ_{spawn} and Ψ_{cull} rules (as shown in fig. 4.7, fig. 4.8 and 4.33). The illustrations show all possible conditions for the Ψ_{spawn} and Ψ_{cull} rules. However, the computational stacking logic for the $\mathbf{CE}_{\text{cube-floating-tower}}$ is different from the previously established $\mathbf{CE}_{\text{cube-tower}}$, and $\mathbf{CE}_{\text{cube-cubocta-tower}}$. However, it can be generalized as below –

- As part of the Ψ_{tower} –
 - If the cardinality of the $\mathbf{N}_{\text{cube}}^{\text{tower}}$ consideration for an existing $\mathbf{E}_{\text{cube}}^{\text{tower}}$ is equal to 2 or 3, the $\mathbf{E}_{\text{cube}}^{\text{tower}}$ survives (i.e. not culled). If the cardinality is otherwise, the $\mathbf{E}_{\text{cube}}^{\text{tower}}$ does not survive (i.e. culled).
 - If the cardinality of the $\mathbf{N}_{\text{cube}}^{\text{tower}}$ consideration for a non-existing $\mathbf{E}_{\text{cube}}^{\text{tower}}$ is equal to 3, the $\mathbf{E}_{\text{cube}}^{\text{tower}}$ is created (i.e. spawned). If the cardinality is otherwise, the $\mathbf{E}_{\text{cube}}^{\text{tower}}$ is not created (i.e. not spawned).
- As part of the $\Psi_{\text{floating-tower}}$ –
 - Thereafter, at every tick the cardinality of all the surviving $\mathbf{E}_{\text{cube}}^{\text{tower}}$ is checked, and based on the assumptions made for the $\Psi_{\text{floating-tower}}$ it is determined and directed to be used for the Ψ_{floating} .
- As part of the Ψ_{floating} –
 - If the cardinality of the $\mathbf{N}_{\text{cube}}^{\text{floating}}$ consideration for an existing or non-existing $\mathbf{E}_{\text{cube}}^{\text{tower}}$ is equal to 0, the $\mathbf{E}_{\text{multi}}$ is created. If the cardinality is otherwise, the $\mathbf{E}_{\text{cube}}^{\text{floating}}$ is not created.

As the $CE_{cube-floating-tower}$ serves as a conceptual continuation of the $CE_{cube-tower}$, and the $CE_{cube-cubocta-tower}$, the aforementioned state conditions which have a considerable precedence on the state conditions of the $CE_{cube-tower}$, are supported by the entire series of procedural sequences (with case studies, simulation and prototyping) performed on the taxon, and thus need not be tested again. However, regarding the new state conditions pertaining to the $\Psi_{floating}$, and $\Psi_{floating-tower}$, the research necessitates rigorous evaluation, testing, prototyping and versioning if required.

The prototyping methodology considered for the $CE_{cube-tower}$ was absolutely revised for the $CE_{cube-cubocta-tower}$, which produced relatively favorable and precise results for the same. Following the outcomes of the $CE_{cube-cubocta-tower}$, the prototyping for the construction of the $CE_{cube-floating-tower}$ would also be performed by means of a workshop that conducts computational simulation of the CE to ensure all the 3 Ψ entities - Ψ_{tower} , $\Psi_{floating}$, and $\Psi_{floating-tower}$ are performing accurately, consistently and yet symbiotically without any bugs or redundancies.

Thus, a similar approach considered for the prototyping of the $CE_{cube-cubocta-tower}$ (as explained and illustrated in the 4.3.3) was implemented. However, as established in the $CE_{cube-cubocta-tower}$ where, the entire computational strategy had to be laid down before initiating a workshop, the $CE_{cube-floating-tower}$ would also require to be strategized by drawing and illustrating a UML Sequence Diagram that demonstrates a user or designer (or actor) going through the runtimes of all the components of the $CE_{cube-floating-tower}$ to eventually produce an outcome driven by the sequential operations of the Ψ_{tower} , $\Psi_{floating}$, and $\Psi_{floating-tower}$ in spawning or culling ϵ_{cube}^{tower} and $\epsilon_{cube}^{floating}$ while maintaining the equilibrium state of constructing a floating tower made of cubes. Following up on the UML Sequence Diagrams established for the $CE_{cube-tower}$ (as illustrated in fig. 4.10), fig. 4.34 illustrates the role, interaction and runtime of all the ϵ entities and Ψ entities as identified in the UML Class Diagram (illustrated in fig. 4.28). It illustrates a UML Sequence diagram that determines the outcome of all possible different $CE_{cube-floating-tower}$.

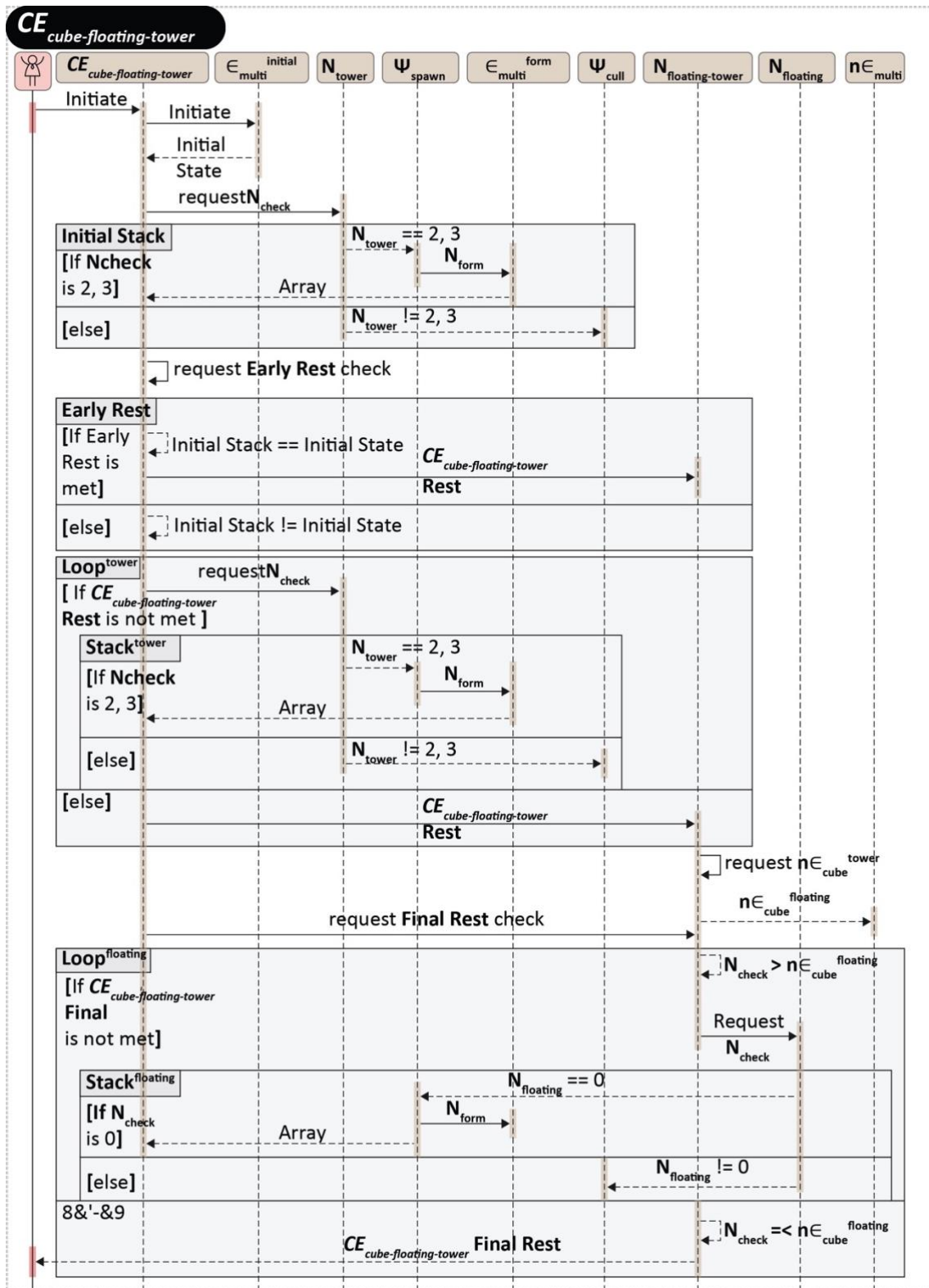


Figure 4.34 – The UML Sequence Diagram for a $CE_{cube-floating-tower}$ with the role, interaction and runtime of all the ϵ entities and ψ entities, and their dependencies for the Early Rest condition.

Illustration and graphics by Author (May 2019).

Similar to the previously established UML Sequence Diagram for the $CE_{cube-tower}$, and the $CE_{cube-cubocta-tower}$, the one jointly illustrated for the $CE_{cube-floating-tower}$ in fig. 4.34 has been generated with the following assumptions –

- The Actor would be the user who initiates the $CE_{cube-floating-tower}$.
- $CE_{cube-floating-tower}$ would be considered as one of the objects, as it has its own lifeline that represents the runtime of the entire algorithm, while the other objects in the diagram would be considered as $\epsilon_{multi}^{initial}$, N_{tower} , Ψ_{spawn} , ϵ_{multi}^{form} , Ψ_{cull} , $N_{floating-tower}$, and ϵ_{multi} in the chronological order of their use and application in the $CE_{cube-cubocta-tower}$.
- The Diagram also involves returning the $CE_{cube-floating-tower}$ as an **early rest** condition, if the **Initial State Array** and the **Initial Stack Array** are the same. If not, the remaining algorithm continues on a while loop until the $CE_{cube-floating-tower}$ **rest** condition (mentioned in 4.3.2) is met to end the algorithm.

Considering the UML Sequence Diagram, a Grasshopper definition was run inside the Rhino 7 interface (The definition labelled as $CE_{cube-floating-tower}$ has been appended to the Annexure – Definitions). Similar to the $CE_{cube-cubocta-tower}$, third-party components used to perform the specific algorithm functions of the $CE_{cube-floating-tower}$ are –

- **Initiate** – To populate the random Initial array of $\epsilon_{multi}^{initial}$.
- **Array** – To generate ϵ_{cube}^{tower} and $\epsilon_{cube}^{floating}$ using the Lunchbox¹³⁸ plugin.
- **N_{check}** – To check the cardinality of Neighbouring cells using the **OR** logic gate.
- **Loop** – To generate a while loop using the Anemone¹³⁹ plugin.

¹³⁸ Lunchbox for Grasshopper (2012). Omaha, USA: Proving Ground Apps.

¹³⁹ Anemone 0.4 (2015). Poznań, Poland: Mateusz Zwierzycki.

Employing the Grasshopper definition, several test $CE_{cube-floating-tower}$ were prototyped in the Rhino 7 environment. As these were initial prototypes, the cardinality of $\epsilon_{multi}^{initial}$ was restricted to 2 and 3. Fig. 4.35 illustrates the outcomes of a few inputs for the $CE_{cube-floating-tower}$ below. The figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), the rest of the ϵ_{cube}^{tower} denoted by ■ (R,G,B – 204,204,204) colored cube, and the $\epsilon_{cube}^{floating}$ denoted by ■ (R,G,B – 207,235,255).

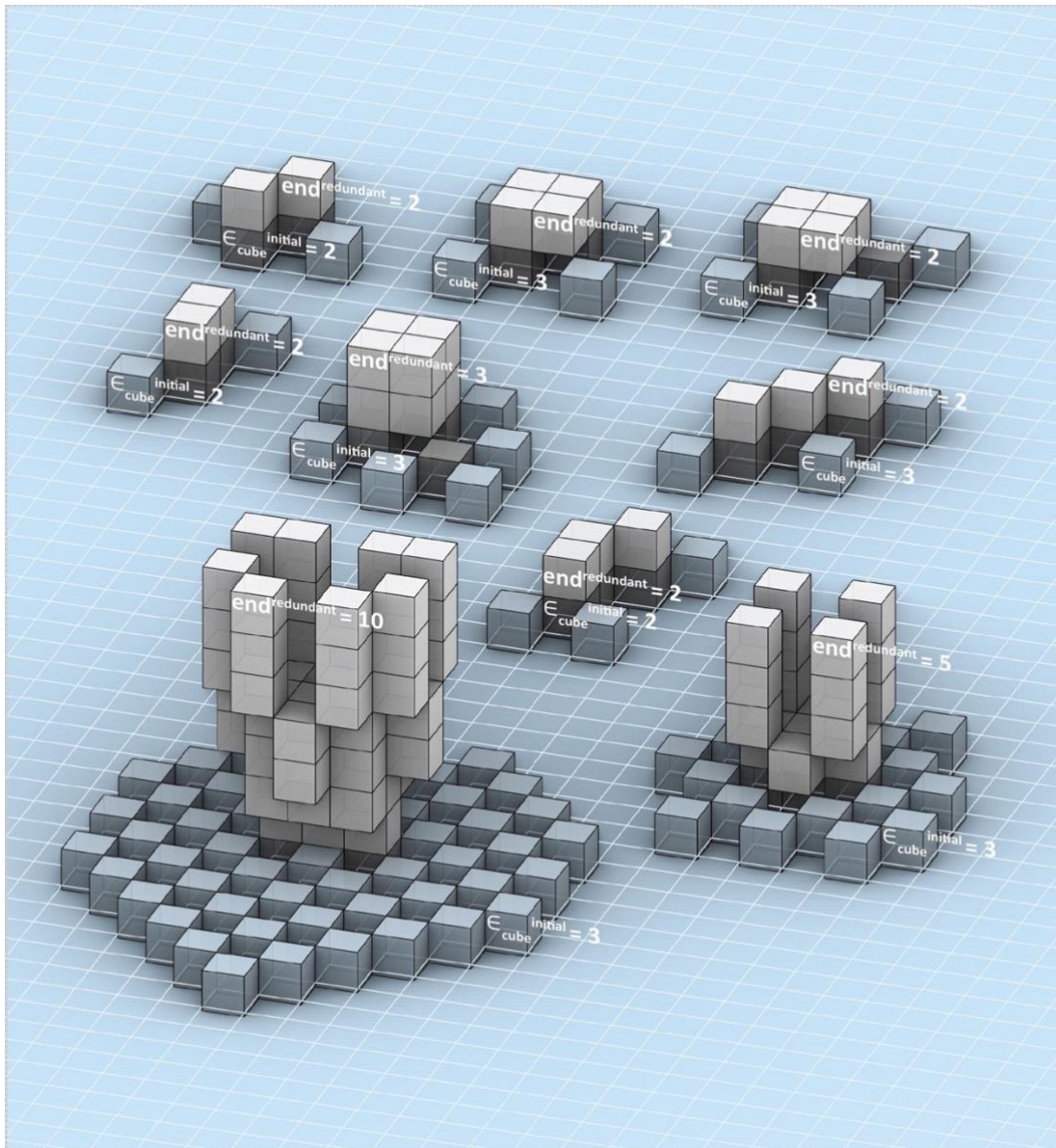


Figure 4.35 – Initial Prototypes of several $CE_{cube-floating-tower}$ with 2 and 3 $\epsilon_{cube}^{initial}$. Model, algorithm, illustration and graphics by Author (May 2019).

Fig. 4.36 illustrates the outcomes of one of the tallest $CE_{cube-floating-tower}$ with 10 $\epsilon_{multi}^{initial}$ that was generated during the 'Computation as a Design tool'. The initial state is denoted by ■ (R,G,B – 129,129,129), the rest of the ϵ_{cube}^{tower} denoted is by ■ (R,G,B – 204,204,204), and the $\epsilon_{cube}^{floating}$ is denoted by ■ (R,G,B – 207,235,255).

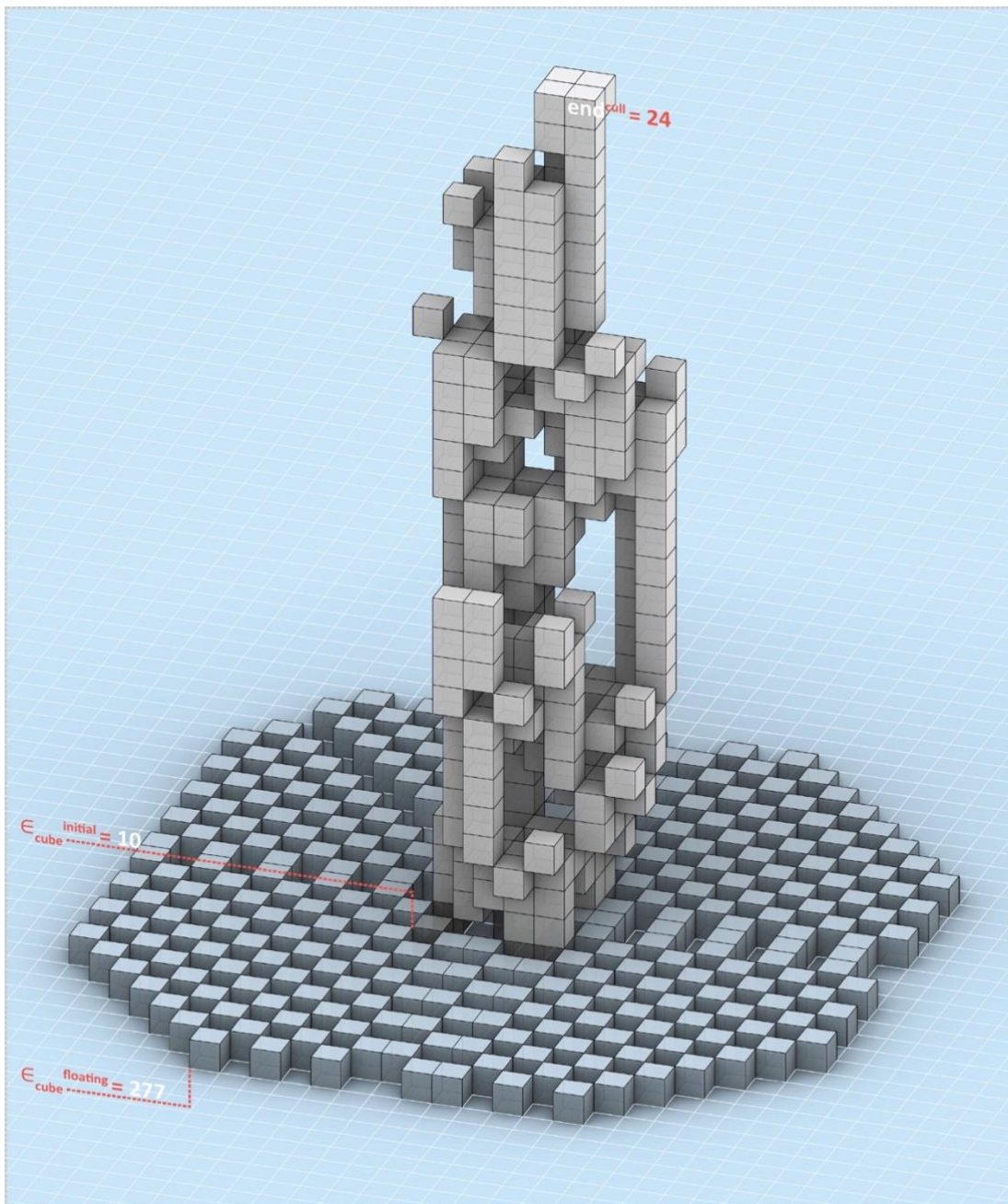


Figure 4.36 – Prototype of a selected $CE_{cube-floating-tower}$ with 10 $\epsilon_{cube}^{initial}$ that had a runtime of 24 t^{tick} before reaching the rest state. Model, algorithm, Illustration and graphics by Author (May 2019).

4.4.4 CE^{SEME}

Similar to the previous taxa (as explained in 4.2.4 and 4.3.4), fig. 4.35 and fig. 4.36 demonstrate the executions and prototyping of the $CE_{cube-floating-tower}$, however, more outcomes focusing on the taxonomy creation have been illustrated in the next chapter (5 | On the consequences of Computational Ecosystems) with further analysis comparing all the other taxa from the procedural sequences in chapter 6 (On the investigative analysis of Computational Ecosystems). However, (similar to 4.2.4 and 4.3.4) the methodology adopted for the CE needs to be analyzed before continuing. Also, as the CE^{SEME} serves as an extension and continuation to the CE^{SESE} and a conceptual equivalent to the CE^{MESE} , the amendments made in the methodology of CE^{SEME} should be addressed as the following –

- To ensure that the CE do not just serve as a continuation of the 3D Cellular Automata (as in a CA of the Conway Model expressed in 3D) efforts have been made to ensure resemblance to actual built-form that can be imagined in the structure of the world.
- The inclusion of 2 different material environments has provided enough context for the $CE_{cube-floating-tower}$ demonstrating how the CE^{SEME} (or any CE) could eventually generate built form within real-life context and parameters.

The CE^{SEME} handles the complexity of the multiplicity of the distinct Ψ entities. But, the CE^{MEME} provides the ultimate complexity in the multiplicity of both the ϵ and Ψ entities. Thus, the following observations made for the CE^{SEME} could be crucial.

- The prototyping stage needs to move on from merely the computational simulation to also provide physical, empirical prototyping.
- A workshop with an amalgam of Computational Design and Digital Fabrication serving as a link between the *algorithm* and *built form* could be conducted.

4.5 Multi Elements Multi Economies Ecosystem (CE^{MEME})

Similar to the CE^{SESE} , CE^{MESE} , and CE^{SEME} , the conceptual framework of a CE^{MEME} has already been sufficiently illustrated in the previous chapters (introduced in 1.2.2, illustrated in fig. 1.9, defined in 3.2.5). Moreover, a Structural UML Diagram establishing the attributes and operations of a canonical version of the ϵ and ψ has also been illustrated (in fig. 3.5 as part of 3.2.5). Like CE^{SESE} , CE^{MESE} , and CE^{SEME} , to construct, taxonomize and prototype this procedural sequence, the research would have to consider tangible parameters in the context of a CE^{MEME} , which could be derived from a combination of real-life examples and constraints, and the conceptual framework for the case studies already established in CE^{SESE} , CE^{MESE} , and CE^{SEME} .

However, unlike already established taxa, the CE^{MEME} would serve as the amalgamation and continuation of CE^{MESE} , and CE^{SEME} , due to the multiplicity of both the ϵ and ψ parameters. Thus, following the previous procedural sequences, the Author has decided to continue with the conceptual example of **MEME** illustrated in 1.2.2. Using the example scenario would methodically increase the ϵ and ψ entities to 2 (instead of the ambiguous multi) and use the canonical UML Class Diagram. Thus, before defining each parameter, the components of the CE^{MEME} can be –

- ϵ^1 – As illustrated in the example scenario, the ϵ would be represented by hexahedrons or cubes, with unit dimensions.
- ϵ^2 – As illustrated in the example scenario, the ϵ would be represented by cuboctahedrons, with unit dimensions.
- ψ^1 – As illustrated in the example scenario, the ψ^1 would be represented by structural stability in the form of axial loads.
- ψ^2 – As illustrated in the example scenario, the ψ^2 would be represented by buoyancy (as per the Archimedes principle).

4.5.1 Case Studies

Although serving as an amalgamation of CE^{MESE} , the CE^{SEME} which theoretically and semantically means that it can be treated as an ecosystem that is inhabited or cohabited by multiple predetermined species and these species (individually, categorically and collectively), are constructed, monitored, and governed by multiple predetermined rule sets, the CE^{MEME} does not serve as a direct sequel to the either the CE^{MESE} , or the CE^{SEME} in terms of fulfilling the Procedural Sequences.

Moreover, the multiplicity of the ϵ and ψ parameters doesn't just imply that there would be two distinct ϵ and ψ entities, but the iterations of their collective assemblages would also have to be treated as a distinct ϵ and ψ parameters. Thus, the following addition must be made to the assumptions stated in the last section (similar to those done in 4.3.1 and 4.4.1) –

- For the ϵ parameters –
 - ϵ^1 – Hexahedrons or cubes, with unit dimensions, which can be programmed as sentient elements similar to the ϵ_{cube} , that are aware of their existence, and thus can be considered as $\epsilon_{\text{biotic}}^1$.
 - ϵ^2 – Cuboctahedrons, with unit dimensions, which can be programmed as sentient elements similar to the ϵ_{cube} , (however with morphological modifications) that are aware of their existence, and thus can be considered as $\epsilon_{\text{biotic}}^2$.
 - ϵ^{1-2} – A collective entity of the ϵ^1 and ϵ^2 , which can be programmed as sentient elements being treated as a ϵ_{cube} **array** (similar to the Arrays formed by the Stack operations) that are aware of their existence, and thus can be considered as $\epsilon_{\text{biotic}}^{1-2}$.

- For the Ψ parameters –
 - Ψ^1 – A rule set that determines the stacking of $\mathbf{\epsilon}_{\text{cube}}$ in the form of state conditions considering an overall structural stability (as per axial loads, and thus can be considered as Ψ_{stack} .
 - Ψ^2 – A rule set that maintains the buoyancy of $\mathbf{\epsilon}_{\text{cube}}$ in the form of state conditions (as per the Archimedes principle), and thus can be considered as Ψ_{buoyancy} .
 - Ψ^{1-2} – A rule set that determines the order and preference of the Ψ^1 and Ψ^2 for every $\mathbf{\epsilon}_{\text{cube}}$ in the form of state conditions), and thus can be considered as $\Psi_{\text{buoyant-stack}}$.

Following the case study of the collective assemblages of the *Eciton hamatum* forming live bridges out of their own bodies (elaborated in 4.2.1, and illustrated in fig. 4.1) as a theoretical precedent for the CE^{SESE} and CE^{MESE} , and the case study of the collective assemblages of the *Solenopsis Invicta* forming live rafts out of their own bodies (elaborated in 4.4.1, and illustrated in fig. 4.26) as a theoretical precedent for the CE^{SEME} , the CE^{MEME} seeks an amalgamated collective assemblage of the *Eciton hamatum* and *Solenopsis Invicta* in a considerably complex manner as compared to that sought by the CE^{SEME} . (Although they don't exist in nature in the way that the case study suggests), the floating ant bridges elucidated in 4.4.1 would be considered as theoretical precedent for establishing the simulations and prototyping of the CE^{MEME} . However, the CE^{SEME} which has multiplicity in only the Ψ parameters establishes the procedural sequence with relatively less complexity. The CE^{MEME} on the other hand, is the most complex taxon that has been established thus far. Hence, *Eciton hamatum*, *Solenopsis Invicta* and their inter-related assemblages would be considered as precedents for the three aforementioned $\mathbf{\epsilon}$ entities. And, the live bridges, live rafts and the rules pertaining to connecting the bridges to the rafts would be considered as precedents for the three aforementioned Ψ entities.

If a rudimentary computational model was made for the simulation of the bridging and rafting behavior of the ants it would translate to a **CE** quite literally (as a **CE** has been already defined to be a Hybrid Bio Plausible Bio-inspired Stochastic Optimization Algorithm). Say, this proposed **CE**, following a basic nomenclature system, could be termed as **CE_{eciton-solenopsis-bridge-raft}** (as the **CE** reflects the bridging properties of the eciton hamatum interacting with the rafting properties of the solenopsis invictus), and would then consist of the following components:

- ϵ^1 – The Eciton Hamatum as biotic, ambulatory agents that are self-aware of their physical properties such as their weight, weight-carrying capacity, movement speed, and gripping abilities. Thus, ϵ_{eciton} .
- ϵ^2 – The Solenopsis Invictus as biotic, ambulatory agents that are self-aware of their physical properties such as their weight, weight-carrying capacity, movement speed, and gripping abilities. Thus, $\epsilon_{solenopsis}$.
- ψ^1 – The bridging ability. Forming collective assemblages by means of connection techniques, span-depth ratios for optimum bridge structures, and load calculations (considering the live and dead loads). Thus, ψ_{bridge} .
- ψ^2 – The rafting ability. Forming collective assemblages by means of connection techniques, span-depth ratios for optimum floating structures, and load calculations (considering the live and dead loads). Thus, ψ_{raft} .
- ϵ^{1-2} – The Eciton Hamatum and Solenopsis Invictus. Acting as biotic, ambulatory agents that serve as an amalgam of ϵ_{eciton} and, $\epsilon_{solenopsis}$. Thus, $\epsilon_{eciton-solenopsis}$.
- ψ^{1-2} – The prioritization ability. Assessing the neighbourhood condition and the physical and structural properties of the ψ_{bridge} and the ψ_{raft} by means of density calculation of context and load calculations. Thus, $\psi_{bridg-raft}$.

However, owing to the multiplicity in both the ϵ and ψ parameters, all the ϵ entities would have to be considered for all the ψ entities, and all the ψ entities would have to be considered for all the ϵ entities. Thus, establishing all the possible combinations for all possible ϵ and ψ parameters. These new components of the already established $CE_{eciton-solenopsis-bridge-raft}$ would be –

- $\epsilon_{eciton}^{bridge}$ – The Eciton Hamatum performing the ψ_{bridge} .
- ϵ_{eciton}^{raft} – The Eciton Hamatum performing the ψ_{raft} .
- $\epsilon_{eciton}^{bridge-raft}$ – The Eciton Hamatum performing the $\psi_{bridge-raft}$.
- $\epsilon_{solenopsis}^{bridge}$ – The Solenopsis Invictus performing the ψ_{bridge} .
- $\epsilon_{solenopsis}^{raft}$ – The Solenopsis Invictus performing the ψ_{raft} .
- $\epsilon_{solenopsis}^{bridge-raft}$ – And the Solenopsis Invictus performing the $\psi_{bridge-raft}$.
- ψ_{bridge}^{eciton} – These would be the bridging rule sets for ϵ_{eciton} .
- ψ_{raft}^{eciton} – These would be the rafting rule sets for ϵ_{eciton} .
- $\psi_{bridge-raft}^{eciton}$ – These would be the prioritizing rule sets for the ϵ_{eciton} .
- $\psi_{bridge}^{solenopsis}$ – These would be the bridging rule sets for $\epsilon_{solenopsis}$.
- $\psi_{raft}^{solenopsis}$ – These would be the rafting rule sets for $\epsilon_{solenopsis}$.
- $\psi_{bridge-raft}^{solenopsis}$ – These would be the prioritizing rule sets for the $\epsilon_{solenopsis}$.

Thus, like the CE^{MESE} and CE^{SEME} (as per 4.3.1 and 4.4.1), the CE^{MEME} would have more ϵ and ψ entities than the predetermined number of ϵ and ψ parameters.

Thus, the similarities between a CE^{MEME} (derived from the example scenario and the CE^{MESE} and CE^{SEME}) and a $CE_{eciton-solenopsis-bridge-raft}$ can be illustrated in fig. 4.37 below.

























MEME							
$CE_{eciton-solenopsis-bridge-raft}$							
	€ bridge eciton	€ raft eciton	€ bridge eciton-solenopsis	€ bridge solenopsis	€ raft solenopsis	€ bridge-raft solenopsis	
							
	Ψ eciton bridge	Ψ solenopsis bridge	Ψ eciton-solenopsis bridge	Ψ eciton raft	Ψ solenopsis raft	Ψ eciton-solenopsis raft	
	$CE_{cube-cubocta-floating-tower}$						
		€ tower cube	€ floating cube	€ tower cube-cubocta	€ tower cubocta	€ floating cubocta	€ floating cube-cubocta
							
Ψ cube tower		Ψ cubocta tower	Ψ cube-cubocta tower	Ψ cube floating	Ψ cubocta floating	Ψ cube-cubocta floating	

Figure 4.37 – Comparing the CE^{MEME} and a $CE_{eciton-solenopsis-bridge-raft}$. Illustration and graphics by Author (October 2019).

Thus, based on the analogies illustrated in fig. 4.37 and the conceptual semantics established for a $CE_{eciton-solenopsis-bridge-raft}$, The CE^{MEME} could be based on the already established $CE_{cube-tower}$ ² (refer 4.3.1) and the $CE_{cube-floating-tower}$ (refer 4.4.1). However, in case of the CE^{MEME} the constituent € and Ψ entities, would have to be defined in relation to the updated $CE_{eciton-solenopsis-bridge-raft}$. Therefore, the components of the CE^{MEME} are defined as followed –

- € entities – or €^{MEME} entities will be considered as computational substitutes to the 6 € entities established for the $CE_{eciton-solenopsis-bridge-raft}$. Although the functionalities and structural properties of the aforementioned six € entities do not directly translate to the €^{MEME} entities, they contain the same semantics as established to be the components of the $CE_{eciton-solenopsis-bridge-raft}$. Thus, continuing on the example scenario, the €^{MEME} entities can be further hexfurcated as illustrated in fig. 4.37. The properties of each entity, relevant for simulation, will be defined further (in 4.5.2), but the general properties can be as followed –

- All the ϵ^{MEME} entities should be considered as static agents that can be spawned or culled by the CE^{MEME} at the expense of the Ψ^{MEME} entities.
- All the ϵ^{MEME} entities should be made to be self-aware of themselves, as in identifying and quantifying their positions, weights, and bounds.
- All the ϵ^{MEME} entities should also be made to be self-aware of their surroundings and their physical properties, like cardinality of neighbours, array of neighbours, and distance from the initial plane.
- Ψ entities – or Ψ^{MEME} entities will be considered as computational substitutes to the 6 Ψ entities established for the $CE_{eciton-solenopsis-bridge-raft}$. Although the functionalities and structural properties of the six Ψ entities do not directly translate to the Ψ^{MEME} entities, they contain the same semantics. Thus, the Ψ^{MEME} entities and can be further hexfurcated as illustrated in fig. 4.37. The general properties of all the Ψ^{MEME} entities can be defined as followed –
 - To a certain extent, combining operation of the $CE_{cube-cubocta-tower}$, and the $CE_{cube-floating-tower}$ the CE^{MEME} would have the distinct Ψ^{SEME} entities performing a tower stacking logic and a floating raft logic in unison.
 - They should be however, designed as a two-state rationale for all the ϵ^{MEME} entities. While all the Ψ_{tower}^{MEME} could be derived from Ψ_{tower}^{MESE} (defined in 4.3.1), all the Ψ_{tower}^{MEME} could be derived from Ψ_{tower}^{SEME} (defined in 4.4.1)
 - However, on the whole, Ψ^{MEME} entities should seek an equilibrium state that spawns and culls ϵ^{MEME} entities considering if the tower is under-structured, overstructured, sinking or over-buoyant, thus maintaining a reciprocal coupling.

After establishing and defining the ϵ^{SEME} entities and Ψ^{SEME} entities, the CE^{SEME} could be defined under the computational guidelines of **CA** (as elaborated in 3|) as –

- The CE^{MEME} which owing to its specific combination of ϵ^{MEME} entities and Ψ^{MEME} entities can be termed as $CE_{cube-cubocta-floating-tower}$, and will be defined as a functioning ecosystem constituting of sentient, context aware cubes and cuboctahedra as ϵ^{SEME} entities that interact with each other considering the rule sets assigned by Ψ^{MEME} entities to spawn or cull the ϵ^{MEME} entities until the runtime of the $CE_{cube-cubocta-floating-tower}$.
- As the $CE_{cube-cubocta-floating-tower}$ will perform while considering 6 distinct Ψ^{SEME} entities, the $\Psi_{floating-tower}^{cube}$ and $\Psi_{floating-tower}^{cubocta}$ entities will operate as determiners of partial runtimes for the distinct equilibria for the construction of $CE_{cube-cubocta-floating-tower}$. However, one and only one CE that is the aforementioned $CE_{cube-cubocta-floating-tower}$ would be performed for the entire runtime.
- Although derived from the hypothetical amalgamation of intricate bio-inspired behavior of army ants constructing living bridges for foraging trails, and fire ants constructing living rafts to avoid drowning, the $CE_{cube-cubocta-floating-tower}$ is a very complex combination of vertical stacking algorithm and a floating raft algorithm for a combination of cubes and cuboctahedra. Thus, the rule sets for the Ψ^{SEME} entities can be adapted from the Conway model of **CA** (as per 2.3.2). Although these will be considerably modified.
- The Ψ^{MEME} entities would differ considerably from the Conway model of **CA** in the parameter of time. The operation of the Ψ^{MEME} entities will however remain similar to those of the Ψ^{MESE} entities (defined in 4.4.1).

Considering all the above assumptions, definitions, illustrations and examples for the $CE_{cube-cubocta-floating-tower}$, the simulations for this taxon can now be performed.

Elaborating on the terminologies introduced in fig. 4.38, the attributes and operations of the classes can be defined as followed –

- t^{tick} – The periodic increment of time, ensuring the $CE_{cube-cubocta-floating-tower}$ runs.
- t^{rest} – If the condition $CE_{cube-cubocta-floating-tower}$ **rest** is fulfilled, the t^{rest} will be activated. This means that the time increment will stop and the final outcome of the $CE_{cube-cubocta-floating-tower}$ will be outputted to the user.
- $CE_{cube-cubocta-floating-tower}$ **rest** – Is the situation where an ϵ_{multi}^{form} **array** for the last 3 t^{tick} intervals is equal or repetitive.
- ϵ_{multi}^{form} **array** – Is the collective assemblage of ϵ_{multi} for a certain t^{tick} interval.
- ϵ_{multi}^{form} – Is the operation for the formation of a combination of all the six ϵ entities depending on the N_{check} and N_{form} routines for all the 6 Ψ entities.
- N_{form} – Are the binary inputs for the cells to be either spawned (input = 1) or culled (input = 0) depending on the N_{check} conditional.
- N_{check} – Is the binary output from the cells of the distinct neighbourhoods (N) to check if the cells in the N of the ϵ_{multi}^{form} **array** for the previous t^{tick} interval are either spawned (input = 1) or culled (input = 0).
- N – The neighbourhood of the ϵ_{multi} in question (as in all the 6 ϵ entities) depending on how many neighbouring ϵ_{multi} from the previous t^{tick} interval are to be considered to determine the N_{check} and N_{form} routines that eventually determine the **Stack** operation.

Thus, like in the case of the previous taxa, determining the N becomes the most important step before considering to explain the other concepts required to run the $CE_{cube-cubocta-floating-tower}$.

Similar to the computational environment and the *infinite 3D Square grid* that was considered for the $CE_{cube-floating-tower}$ (refer 4.4.2), in case of the construction of the $CE_{cube-cubocta-floating-tower}$ as well, the modelling has been performed in Rhino 7 (refer 3.3), the programming has been performed in Grasshopper for Rhino 7 (refer 3.3) and visualizations have been performed in VRay for Rhino (refer 4.1). Moreover, as the $CE_{cube-cubocta-floating-tower}$ relies on a CE that floats, its computational environment shall be constituted of similar elements of *Air* and *Water* as theorized in 4.4.2.

Although most of the physical properties required for setting up the computational environment by the $CE_{cube-cubocta-floating-tower}$ are similar to those of the previously established $CE_{cube-floating-tower}$ there is a renewed complication that arises from the multiplicity of the ϵ entities. Amongst the six ϵ entities that are already theorized for the $CE_{cube-cubocta-floating-tower}$, only the ϵ_{cube}^{tower} , $\epsilon_{cube}^{floating}$, and $\epsilon_{cube}^{floating-tower}$ have been established in the $CE_{cube-floating-tower}$. Thus, the $\epsilon_{cubocta}^{tower}$, the $\epsilon_{cubocta}^{floating}$, and the $\epsilon_{cubocta}^{floating-tower}$ have to be instilled in the computational environment that was already established for the $CE_{cube-floating-tower}$. Thus, considering computational environment and its constituent materials, fig. 4.39 illustrates the N of all the 6 ϵ entities.



Figure 4.39 – Neighbourhood (N) of all 6 ϵ entities required for the $CE_{cube-cubocta-floating-tower}$ Model, Illustration and graphics by Author (October 2019).

As illustrated in fig. 4.39, the following assumptions can be made for the **N** of all the six **€** entities –

- $N_{\text{cube}^{\text{tower}}}$ – This is the **N** for the $\text{€}_{\text{cube}^{\text{tower}}}$, and it operates in the **Air** environment with eight possible €_{multi} in the neighbourhood.
- $N_{\text{cube}^{\text{floating}}}$ – This is the **N** for the $\text{€}_{\text{cube}^{\text{floating}}}$, and it operates in the **Water** environment with eight possible €_{multi} in the neighbourhood.
- $N_{\text{cube-cubocta}^{\text{tower}}}$ – This is the **N** for the $\text{€}_{\text{cube-cubocta}^{\text{tower}}}$, and it operates in the **Air** environment, where it prioritizes eight €_{multi} in the neighbourhood.
- $N_{\text{cubocta}^{\text{tower}}}$ – This is the **N** for the $\text{€}_{\text{cubocta}^{\text{tower}}}$, and it operates in the **Air** environment with four possible €_{multi} in the neighbourhood.
- $N_{\text{cubocta}^{\text{floating}}}$ – This is the **N** for the $\text{€}_{\text{cubocta}^{\text{floating}}}$, and it operates in the **Water** environment with four possible €_{multi} in the neighbourhood.
- $N_{\text{cube-cubocta}^{\text{floating}}}$ – The **N** for the $\text{€}_{\text{cube-cubocta}^{\text{floating}}}$, and it operates in the **Water** environment, where it prioritizes eight €_{multi} in the neighbourhood.

The above definitions of all distinct **N** give a clear understanding of the physical constraints that each of the six distinct **€** entities will have while having their respective N_{check} and N_{form} routines driven by their respective Stack conditions. The manner in which $N_{\text{cube}^{\text{tower}}}$, $N_{\text{cubocta}^{\text{tower}}}$, and the $N_{\text{cube-cubocta}^{\text{tower}}}$ direct the €_{multi} to construct an optimally structurally supported tower has been already established in the CE^{MESE} in the form of $\text{CE}_{\text{cube-cubocta-tower}}$ (as established in 4.3.2 and as illustrated in fig. 4.17, fig. 4.18, and fig. 4.19), and thus, it shall be implemented for the $\text{CE}_{\text{cube-cubocta-floating-tower}}$. Also, the manner in which the $N_{\text{cube}^{\text{floating}}}$ direct the €_{multi} to construct an optimally structurally supported tower that floats with a maximum of 20% of its volume immersed in water has been already established in the CE^{SEME} in the form of the $\text{CE}_{\text{cube-floating-tower}}$ (as established in 4.4.2 and as illustrated in fig. 4.32).

However, the $N_{\text{cubocta}}^{\text{floating}}$, and the $N_{\text{cube-cubocta}}^{\text{floating}}$, in relation with the other four N considerations have not been introduced yet, and thus need to be established before setting up the **State Conditions** for the $CE_{\text{cube-cubocta-floating-tower}}$. Similar to the assumptions made for the $\epsilon_{\text{cube}}^{\text{floating}}$ in context of the Ψ_{floating} as established for the $CE_{\text{cube-floating-tower}}$ (4.4.2), following are the assumptions made for the $\epsilon_{\text{cubocta}}^{\text{floating}}$, and $\epsilon_{\text{cube-cubocta}}^{\text{floating}}$ in context of the $\Psi_{\text{floating}}^{\text{cubocta}}$ and the $\Psi_{\text{floating}}^{\text{cube-cubocta}}$ –

- All the operations related to the spawning or culling of the $\epsilon_{\text{cubocta}}^{\text{floating}}$, and $\epsilon_{\text{cube-cubocta}}^{\text{floating}}$ in the context of the $CE_{\text{cube-cubocta-floating-tower}}$ shall be made in the **Water**. That means, only the $N_{\text{cubocta}}^{\text{floating}}$ and the $N_{\text{cube-cubocta}}^{\text{floating}}$ can be considered for the respective N_{form} routines.
- The $\epsilon_{\text{cubocta}}^{\text{floating}}$ (and all other cuboctahedra) shall be hollow, made of unit mass and unit volume, such that, if one $\epsilon_{\text{cubocta}}^{\text{floating}}$ is solitarily dropped in water, it shall float while having exactly 10% of its volume immersed.
- However, if more $\epsilon_{\text{cubocta}}^{\text{floating}}$ or $\epsilon_{\text{cube-cubocta}}^{\text{floating}}$ are added (by the virtue of any of the 6 Ψ entities) the resultant array of ϵ_{multi} shall never have more than 20% of the collective volume of the $\epsilon_{\text{cube-cubocta}}^{\text{floating}}$ immersed in the water.
- In case the volume of $\epsilon_{\text{cubocta}}^{\text{floating}}$ or $\epsilon_{\text{cube-cubocta}}^{\text{floating}}$ immersed in the water is more than 20%, more or $\epsilon_{\text{cube-cubocta}}^{\text{floating}}$ shall be added to the $N_{\text{cube-cubocta}}^{\text{floating}}$ by means of the N_{form} routine.

Albeit the above assumptions, the hypothesis for the morphology of a cuboctahedron established for the $CE_{\text{cube-cubocta-tower}}$ (in 4.3.2), shall also be assumed in case of the $CE_{\text{cube-cubocta-floating-tower}}$. Thus, for a cuboctahedron with radius R_{cubocta} and perfectly bound inside a cube with radius R_{cube} , the following are true –

$$R_{\text{cubocta}} = \frac{\sqrt{2} R_{\text{cube}}}{\sqrt{3}} \quad ; \quad V_{\text{cubocta}} = \frac{5V_{\text{cube}}}{6} \quad ; \quad \text{where, } V_{\text{cubocta}} \text{ is volume of 1 cuboctahedron bound inside a cube with volume } V_{\text{cube}}.$$

Thus, considering the aforementioned assumptions, an array of $n\epsilon_{\text{cubocta}}^{\text{tower}}$ (n number of $\epsilon_{\text{cubocta}}^{\text{tower}}$ elements) will always require $5n/6 \epsilon_{\text{cube}}^{\text{floating}}$ to make sure that the $CE_{\text{cube-cubocta-floating-tower}}$ floats in such a way that no more than 20% of the collective volume of the $\epsilon_{\text{cubocta}}^{\text{floating}}$ is immersed in the water. Moreover, since the $\epsilon_{\text{cube}}^{\text{tower}}$ and the $\epsilon_{\text{cubocta}}^{\text{tower}}$ are morphologically same, the $CE_{\text{cube-cubocta-floating-tower}}$ will basically be populating hollow cuboctahedra both vertically (to make a tower while following the $\Psi_{\text{tower}}^{\text{cubocta}}$) and horizontally (to ensure the tower floats while following the $\Psi_{\text{floating}}^{\text{cubocta}}$). Thus fig. 4.40 below exemplifies a $\epsilon_{\text{multi}}^{\text{initial}}$ array, and documents its implementation of the assumptions made for the $\Psi_{\text{floating}}^{\text{cubocta}}$ above.

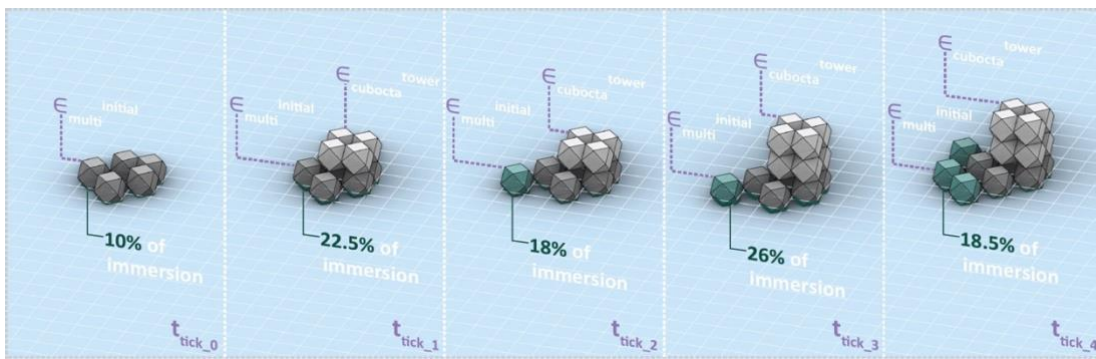


Figure 4.40 – Example implementation of $\Psi_{\text{tower}}^{\text{cubocta}}$ followed by $\Psi_{\text{floating}}^{\text{cubocta}}$ for the runtime of a $CE_{\text{cube-cubocta-floating-tower}}$ with 4 $\epsilon_{\text{multi}}^{\text{initial}}$. Model and Illustration by Author (October 2019).

As illustrated above, at t^{tick_0} interval array of five $\epsilon_{\text{multi}}^{\text{initial}}$ elements employs a Ψ_{tower} and spawns four more $\epsilon_{\text{cubocta}}^{\text{tower}}$ at the t^{tick_1} interval thus making a total of nine $\epsilon_{\text{multi}}^{\text{initial}}$ elements. Without the implementation of the Ψ_{floating} at the t^{tick_1} interval, the CE would still float, albeit with 22.5% immersion of its $\epsilon_{\text{multi}}^{\text{initial}}$ elements. This is barred as per the assumptions, and thus, at t^{tick_2} interval, one new $\epsilon_{\text{cubocta}}^{\text{floating}}$ is spawned which ensures 18% immersion of the $\epsilon_{\text{multi}}^{\text{initial}}$ and the $\epsilon_{\text{cubocta}}^{\text{floating}}$ elements. At t^{tick_3} interval, four more $\epsilon_{\text{cubocta}}^{\text{tower}}$ are spawned, thus making a total of fourteen $\epsilon_{\text{multi}}^{\text{initial}}$ elements. With no new $\epsilon_{\text{cubocta}}^{\text{floating}}$ elements, the CE would still float, but now with 26% immersion of its $\epsilon_{\text{multi}}^{\text{initial}}$ elements. This is still barred, and thus, at t^{tick_4} interval, two new $\epsilon_{\text{cubocta}}^{\text{floating}}$ are spawned which ensure 18.5% immersion. As the CE reaches its **end**^{redundant} condition, the $CE_{\text{cube-cubocta-floating-tower}}$ is returned.

Similar to the consideration of the $\Psi_{\text{floating-tower}}$ rulesets for the $CE_{\text{cube-floating-tower}}$, the above assumptions will be considered as the $\Psi_{\text{floating-tower}^{\text{cubocta}}}$ rule sets, as the assumptions examine the results of the $\Psi_{\text{tower}^{\text{cubocta}}}$ rule and then direct the $\Psi_{\text{floating}^{\text{cubocta}}}$, thus serving as a prioritizing junction. $\Psi_{\text{floating-tower}^{\text{cubocta}}}$ rule set also serves as a time buffer, as it halts to compute how many $E_{\text{cubocta}^{\text{floating}}}$ are required to be spawned. Similar to its previous applications for the $CE_{\text{cube-cubocta-tower}}$ and the $CE_{\text{cube-floating-tower}}$, the state conditions of the amalgam - $CE_{\text{cube-cubocta-floating-tower}}$ will also be determined by employing the Conway Model of **CA** (refer 2.3.2). Thus, considering the CE^{MESE} and the CE^{SEME} as precedence, and deriving from the Conway Model, the state conditions for the $CE_{\text{cube-cubocta-floating-tower}}$ can be established as –

- Every existing $E_{\text{multi-n+1}}$ at the $t^{\text{tick-n+1}}$ interval interacts with its **N** consideration of either the $E_{\text{cube-n}}$ (as $N_{\text{cube-n}}$) or $E_{\text{cubocta-n}}$ (as $N_{\text{cubocta-n}}$) at the $t^{\text{tick-n}}$ interval, thus performing **N**_{check}. For the $t^{\text{tick-n+1}}$ interval, it then performs the **N**_{form} routine as part of the $\Psi_{\text{tower}^{\text{cube}}}$, $\Psi_{\text{tower}^{\text{cubocta}}}$, and $\Psi_{\text{tower}^{\text{cube-cubocta}}}$ based on the following conditions –
 - If the **N** is $N_{\text{cube-n}}$ with an array of two or three E_{multi} at the $t^{\text{tick-n}}$ interval, the **N**_{form} spawns a $E_{\text{cube-n+1}}$ for the $t^{\text{tick-n+1}}$ interval.
 - Also, for the $N_{\text{cubocta-n}}$ with an array of two or three E_{multi} at the $t^{\text{tick-n}}$ interval, the **N**_{form} spawns a $E_{\text{cubocta-n+1}}$ for the $t^{\text{tick-n+1}}$ interval.
- Every non-existing $E_{\text{multi-n+1}}$ at the $t^{\text{tick-n+1}}$ interval interacts with its $N_{\text{cube-cubocta}}$ at the $t^{\text{tick-n}}$ interval, thus performing **N**_{check}. For the $t^{\text{tick-n+1}}$ interval, it then performs the **N**_{form} as part of the $\Psi_{\text{tower}^{\text{cube}}}$, $\Psi_{\text{tower}^{\text{cubocta}}}$, and $\Psi_{\text{tower}^{\text{cube-cubocta}}}$ based on the following conditions –
 - With an array of four E_{multi} in its $N_{\text{cube-cubocta}}$ at the $t^{\text{tick-n}}$ interval, the **N**_{form} spawns a $E_{\text{cubocta-n+1}}$ for the $t^{\text{tick-n+1}}$ interval.

- With an array of three \mathbf{E}_{cube} or four $\mathbf{E}_{\text{cubocta}}$ in its $\mathbf{N}_{\text{cube-cubocta}}$ at the $t^{\text{tick-n}}$ interval, the \mathbf{N}_{form} spawns a $\mathbf{E}_{\text{cubocta-n+1}}$ for the $t^{\text{tick-n+1}}$ interval.
- For every other condition of $\mathbf{N}_{\text{check}}$ at the $t^{\text{tick-n}}$ interval, as part of the $\Psi_{\text{tower}^{\text{cube}}}$, $\Psi_{\text{tower}^{\text{cubocta}}}$, and $\Psi_{\text{tower}^{\text{cube-cubocta}}}$, the \mathbf{N}_{form} culls all the existing or non-existing $\mathbf{E}_{\text{multi}}$ for the $t^{\text{tick-n+1}}$ interval.
- At every $t^{\text{tick-n+1}}$ interval, as part of the $\Psi_{\text{floating-tower}^{\text{cube}}}$, $\Psi_{\text{floating-tower}^{\text{cubocta}}}$, and $\Psi_{\text{floating-tower}^{\text{cube-cubocta}}}$, the cardinality of $\mathbf{E}_{\text{cube-n}^{\text{tower}}}$ elements that are newly spawned and those that are not culled is measured, and the total number of new elements $n\mathbf{E}_{\text{multi-n+1}^{\text{floating}}}$ elements is determined by –
 - Reducing the cardinality of the $\mathbf{E}_{\text{cube-n}^{\text{tower}}}$ elements in the $t^{\text{tick-n}}$ interval from the cardinality of the $\mathbf{E}_{\text{cube-n}^{\text{tower}}}$ elements in the $t^{\text{tick-n+1}}$ interval. This determines the $n\mathbf{E}_{\text{cube-n+1}^{\text{floating}}}$ elements.
 - Reducing the cardinality of the $\mathbf{E}_{\text{cubocta-n}^{\text{tower}}}$ elements in the $t^{\text{tick-n}}$ interval from that of the $\mathbf{E}_{\text{cubocta-n}^{\text{tower}}}$ elements in the $t^{\text{tick-n+1}}$ interval and determining two-third of the result thus obtained. This determines the $n\mathbf{E}_{\text{cubocta-n+1}^{\text{floating}}}$ elements.
- Every $\mathbf{E}_{\text{cube-n+1}^{\text{floating}}}$ at the $t^{\text{tick-n+1}}$ interval interacts with the $\mathbf{N}_{\text{cube}^{\text{floating}}}$ of its preceding $\mathbf{E}_{\text{cube-n}^{\text{floating}}}$ at the $t^{\text{tick-n}}$ interval, thus performing $\mathbf{N}_{\text{cube}^{\text{floating-check}}}$. For the $t^{\text{tick-n+1}}$ interval, it then performs the \mathbf{N}_{form} routine based on the following conditions as part of the $\Psi_{\text{floating}^{\text{cube}}}$, and $\Psi_{\text{floating}^{\text{cube-cubocta}}}$, until the required $n\mathbf{E}_{\text{cube-n+1}^{\text{floating}}}$ elements is met –
 - Any non-existing (culled) $\mathbf{E}_{\text{cube-n}^{\text{floating}}}$ with no existing $\mathbf{E}_{\text{cube-n}^{\text{floating}}}$ in its $\mathbf{N}_{\text{cube}^{\text{floating}}}$ is spawned for the $t^{\text{tick-n+1}}$ interval, as if it was being perfectly supported by its counterparts on the edge of the cube, thus leaving the faces open for access.

- Any non-existing (culled) or existing (spawned) $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ with any number of existing $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ in its $\mathbf{N}_{\text{cube}}^{\text{floating}}$ remains culled or is culled for the $t^{\text{tick-}n+1}$ interval, as if it was blocking access, and thus would not sufficiently provide buoyancy.
- Every $\mathbf{E}_{\text{cubocta-}n+1}^{\text{floating}}$ at the $t^{\text{tick-}n+1}$ interval interacts with the $\mathbf{N}_{\text{cubocta}}^{\text{floating}}$ of its preceding $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ at the $t^{\text{tick-}n}$ interval, thus performing distinct $\mathbf{N}_{\text{cubocta}}^{\text{floating-check}}$. For the $t^{\text{tick-}n+1}$ interval, it then performs the \mathbf{N}^{form} routine based on the following conditions as the $\Psi_{\text{floating}}^{\text{cubocta}}$, and $\Psi_{\text{floating}}^{\text{cube-cubocta}}$, until the required $n\mathbf{E}_{\text{cubocta-}n+1}^{\text{floating}}$ elements is met –
 - Any non-existing (culled) $\mathbf{E}_{\text{cubocta-}n}^{\text{floating}}$ with one existing $\mathbf{E}_{\text{cubocta-}n}^{\text{floating}}$ in its $\mathbf{N}_{\text{cubocta}}^{\text{floating}}$ or 1 existing $\mathbf{E}_{\text{cube-cubocta-}n}^{\text{floating}}$ in its $\mathbf{N}_{\text{cube-cubocta}}^{\text{floating}}$ but not both, is spawned for the $t^{\text{tick-}n+1}$ interval, as if it was being perfectly supported by its counterparts on the edge of the cube, thus leaving the faces open for access.
 - Any non-existing (culled) or existing (spawned) $\mathbf{E}_{\text{cube-}n}^{\text{floating}}$ with any other number of existing $\mathbf{E}_{\text{cubocta-}n}^{\text{floating}}$ or $\mathbf{E}_{\text{cube-cubocta-}n}^{\text{floating}}$ in its $\mathbf{N}_{\text{cubocta}}^{\text{floating}}$ or $\mathbf{N}_{\text{cube-cubocta}}^{\text{floating}}$ remains culled or is culled for the next $t^{\text{tick-}n+1}$ interval, as if it was blocking access, and thus would not sufficiently provide buoyancy.

At $t^{\text{tick-}n}$ interval the \mathbf{CE} performs the $\Psi_{\text{tower}}^{\text{cube}}$, $\Psi_{\text{tower}}^{\text{cubocta}}$, and $\Psi_{\text{tower}}^{\text{cube-cubocta}}$ state conditions in unison, and at the $t^{\text{tick-}n+1}$ interval, it performs the $\Psi_{\text{floating-tower}}^{\text{cube}}$, $\Psi_{\text{floating-tower}}^{\text{cubocta}}$, and $\Psi_{\text{floating-tower}}^{\text{cube-cubocta}}$ to determine the $n\mathbf{E}_{\text{cube-}n+1}^{\text{floating}}$ and $n\mathbf{E}_{\text{cubocta-}n+1}^{\text{floating}}$ elements, after which it activates the $\Psi_{\text{floating}}^{\text{cube}}$, $\Psi_{\text{floating}}^{\text{cubocta}}$, and $\Psi_{\text{floating}}^{\text{cube-cubocta}}$, until the required $n\mathbf{E}_{\text{multi-}n+1}^{\text{floating}}$ elements is met, after which the \mathbf{CE} is returned to the user as an iteration. Contrary to the previous taxa, the construction of the $\mathbf{CE}_{\text{cube-cubocta-floating-tower}}$ is much more complex, and has a wide range of Ψ_{spawn} and Ψ_{spawn} conditions, and thus can't be illustrated categorically.

4.5.3 Prototyping

The state conditions in terms of the $\mathbf{N}^{\text{check}}$ and the \mathbf{N}^{form} routines of all the Ψ entities that determine all the distinct **Stack** operations for all the $\mathbf{\epsilon}_{\text{multi}}$ array (and the individual $\mathbf{\epsilon}_{\text{cube}}$ array, $\mathbf{\epsilon}_{\text{cubocta}}$ array, and $\mathbf{\epsilon}_{\text{cube-cubocta}}$ array) have been sufficiently documented. Although, owing to the multiplicity of the $\mathbf{\epsilon}$ and the Ψ parameters, the illustrations for all the possible Ψ_{spawn} and Ψ_{cull} situations have not been demonstrated. However, similar to the computational logic of the $\mathbf{CE}_{\text{cube-cubocta-tower}}$ and the $\mathbf{CE}_{\text{cube-floating-tower}}$, the $\mathbf{CE}_{\text{cube-cubocta-floating-tower}}$ serving as an amalgam of the two, can have generalized **stack conditions** as summarized below –

- As part of the distinct Ψ_{tower} – If the cardinality of the $\mathbf{N}_{\text{cube}}^{\text{tower}}$ consideration for an existing $\mathbf{\epsilon}_{\text{multi}}^{\text{tower}}$ is equal to two or three, the $\mathbf{\epsilon}_{\text{cube}}^{\text{tower}}$ survives (i.e. not culled). If the cardinality is otherwise, the $\mathbf{\epsilon}_{\text{cube}}^{\text{tower}}$ does not survive (i.e. culled). And if the cardinality of the $\mathbf{N}_{\text{cube}}^{\text{tower}}$ consideration for a non-existing $\mathbf{\epsilon}_{\text{multi}}^{\text{tower}}$ is equal to four, the $\mathbf{\epsilon}_{\text{cubocta}}^{\text{tower}}$ is created (i.e. spawned). If the cardinality is otherwise, the $\mathbf{\epsilon}_{\text{multi}}^{\text{tower}}$ is not created (i.e. not spawned).
- As part of the distinct $\Psi_{\text{floating-tower}}$ – Thereafter, at every tick the cardinality of the $\mathbf{\epsilon}_{\text{multi}}^{\text{tower}}$ is checked, and based on the assumptions for the $\Psi_{\text{floating-tower}}$ the $n\mathbf{\epsilon}_{\text{multi}}^{\text{floating}}$ is distinctly determined and used for the Ψ_{floating} .
- As part of the Ψ_{floating} – If the cardinality of the $\mathbf{N}_{\text{multi}}^{\text{floating}}$ consideration for a non-existing $\mathbf{\epsilon}_{\text{multi}}^{\text{floating}}$ is equal to 0, the $\mathbf{\epsilon}_{\text{cube}}^{\text{floating}}$ is created (i.e. spawned). If the cardinality of the $\mathbf{N}_{\text{multi}}^{\text{floating}}$ consideration for a non-existing $\mathbf{\epsilon}_{\text{multi}}^{\text{floating}}$ is equal to 1, the $\mathbf{\epsilon}_{\text{cubocta}}^{\text{floating}}$ is created (i.e. spawned). if the cardinality is otherwise, $\mathbf{\epsilon}_{\text{multi}}^{\text{floating}}$ is not created (i.e. not spawned).

As a natural progression, UML Sequence Diagram needs to be established to prototype the $\mathbf{CE}^{\text{MEME}}$. Fig. 4.41 and fig. 4.42, thus jointly illustrate UML Sequence diagram that determines the outcome of several different $\mathbf{CE}_{\text{cube-cubocta-floating-tower}}$.

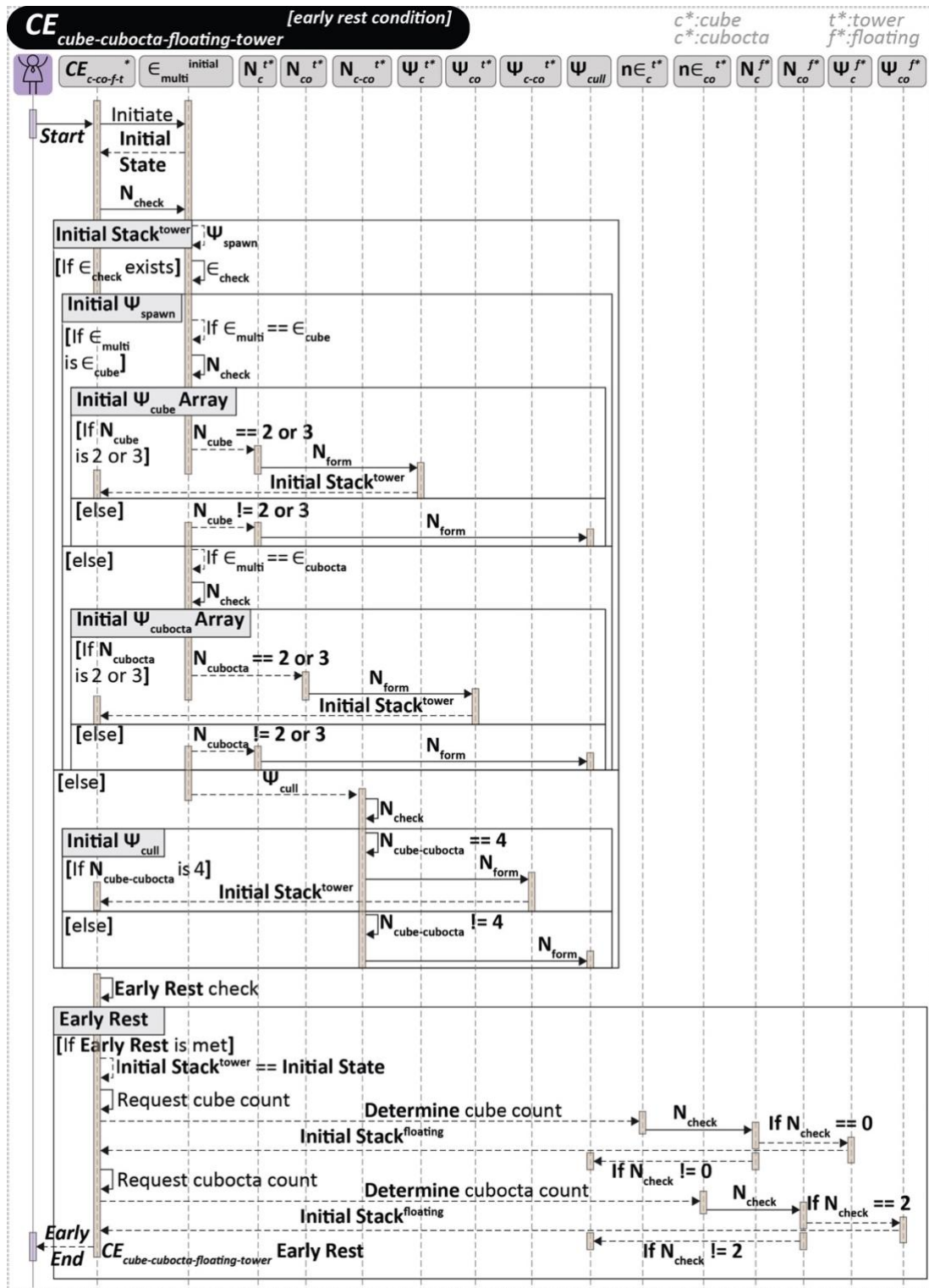


Figure 4.41 – The UML Sequence Diagram for a **CE***cube-cubocta-floating-tower* with the role, interaction and runtime of all the \in entities and Ψ entities, and their dependencies for the Early Rest condition. Illustration and graphics by Author (November 2019).

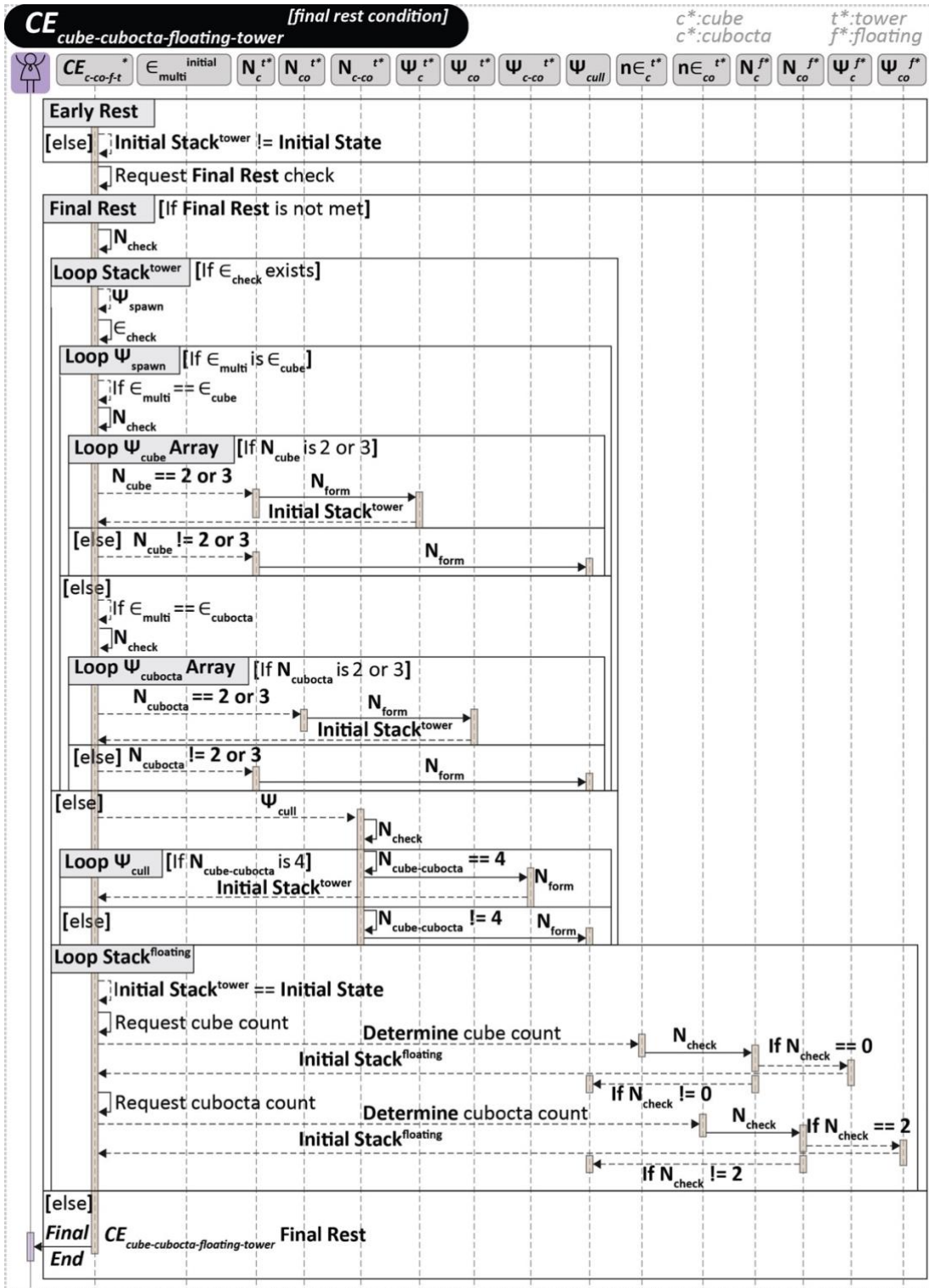


Figure 4.42 – The UML Sequence Diagram for a *CE*_{cube-cubocta-floating-tower} with the role, interaction and runtime of all the € entities and Ψ entities, and their dependencies for the Final Rest condition. Illustration and graphics by Author (November 2019).

As the $CE_{\text{cube-cubocta-floating-tower}}$ serves as a conceptual amalgam of the $CE_{\text{cube-cubocta-tower}}$, and the $CE_{\text{cube-floating-tower}}$, all the aforementioned state conditions are supported by the entire series of procedural sequences (with case studies, simulation and prototyping) performed on the previous taxa, and thus need not be tested again. However, regarding the new state conditions pertaining to the Ψ_{floating} entities, and the complexities generated by the multiplicity of both the ϵ and the Ψ parameters is unprecedented in the previous procedural sequences of this research and thus necessitates rigorous evaluation, testing, prototyping and versioning if required.

Moreover, it has been observed in the previous taxa (particularly in 4.4.4) that the symbiotic relation sought between the built form and the algorithm (and thus generated, maintained and taxonomized by means of all the CE that this thesis has demonstrated so far), has only been rigorously tested by means **Computational Design**, and not yet been empirically tested by means of **Digital Fabrication**. Thus, the prototyping of this taxon seeks to fulfill this objective by conducting a workshop that focuses on digital fabrication by means of additive manufacturing. Due to the ease of operation, readily available resources and absolute repeatability of iterations, FDM (Fused Deposition Modelling) printing has been considered as the chosen means of additive manufacturing.

Moreover, the CE^{MEME} is perfectly suited to be evaluated by means of fabrication owing to the presence of the multiplicities of both the ϵ and the Ψ parameters. Although effective 3D printability by means of FDM printing machines can also be considered as an interdependent Ψ parameter, such an entity (as it relies heavily on the type of printer and the material used for printing) was ignored at this stage. However, the concept of a floating tower in the form of the $CE_{\text{cube-cubocta-floating-tower}}$ could be directly considered for prototyping in a manner that the test confirms if the outcome survives several floors of stacked material while being able to float on water. Although, the built form cannot be constructed while the algorithm performs the computation (sadly, the kind of machinery isn't available as yet!), a 3D printed prototype could be evaluated for its performance as a floating tower.

To perform the evaluations in a didactic format, the author conducted a workshop which aimed at prototyping the **CE^{MEME}** (in the form of the **CE_{cube-cubocta-floating-tower}**) by performing computational simulations to test, evaluate, taxonomize and if required, amend and update the **CE**. The workshop, titled as '**Digital Fabrication Workshop**' was conducted in December 2019 at IES (Indian Education Society's) College of Architecture in Mumbai, India. It was attended by 10 candidates – 4 candidates practicing architecture in the AEC Industry in Mumbai, India and 6 candidates pursuing the B.Arch. degree (students of the 7th and 9th semesters).

For the testing, participants were first introduced to the concept of Computational Ecosystems and its research, and were provided with a lecture on Cellular Automata and its implementation in the research. Further, the participants were demonstrated with several results that were obtained in the previous iterations of similar workshops (i.e. *the Designing ways of designing workshop* conducted at IES, Mumbai – as elaborated in 4.2.3, and the *Computation as a Design tool workshop* conducted at RIT, Kottayam – as elaborated in 4.3.3). As the participants were not well versed with using any computational design software, they were tutored on using different tools and functionalities with Rhino 7 and Grasshopper 3D that were relevant to the research on Computational Ecosystems. The participants were also encouraged to test the already established **CE_{cube-tower}**, **CE_{cube-cubocta-tower}**, and **CE_{cube-floating-tower}** before explicitly evaluating the **CE_{cube-cubocta-floating-tower}**. Moreover, the participants were not particularly well versed in using FDM printers as a digital fabrication tool. Since the outcomes of the **CE_{cube-cubocta-floating-tower}** were to be prototyped on an FDM Printer of the make – Creality Ender 3 Pro¹⁴⁰ using PLA (Poly Lactic Acid – commonly used filament based 3D printing material) plastic, the participants were tutored on using different tools related to modelling 3D printable geometry, slicing 3D printing tool paths, handling a 3D printer for accurate and effective 3d prints and finally cleaning and post processing a 3D print.

¹⁴⁰ Ender 3 Pro (2018). Shenzhen, PRC: Shenzhen Creality 3D Technology Co, Ltd.

After establishing and upgrading their computational skills to a considerable level, the participants were individually tasked with testing the $CE_{cube-cubocta-floating-tower}$. Although they were asked to test the algorithm with $\epsilon_{multi}^{initial}$ Array of 10 to 12, the participants were encouraged to experiment with higher number of $\epsilon_{multi}^{initial}$ Array.

After the successful trials of the, fig. 4.43 illustrates the computational outcome of one of the tallest $CE_{cube-cubocta-floating-tower}$ with 15 $\epsilon_{multi}^{initial}$ that was generated during the '**Digital Fabrication Workshop**'. The figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), the rest of the ϵ_{multi}^{tower} denoted by ■ (R,G,B – 204,204,204) colored cube, and the $\epsilon_{multi}^{floating}$ denoted by ■ (R,G,B – 207,235,255).

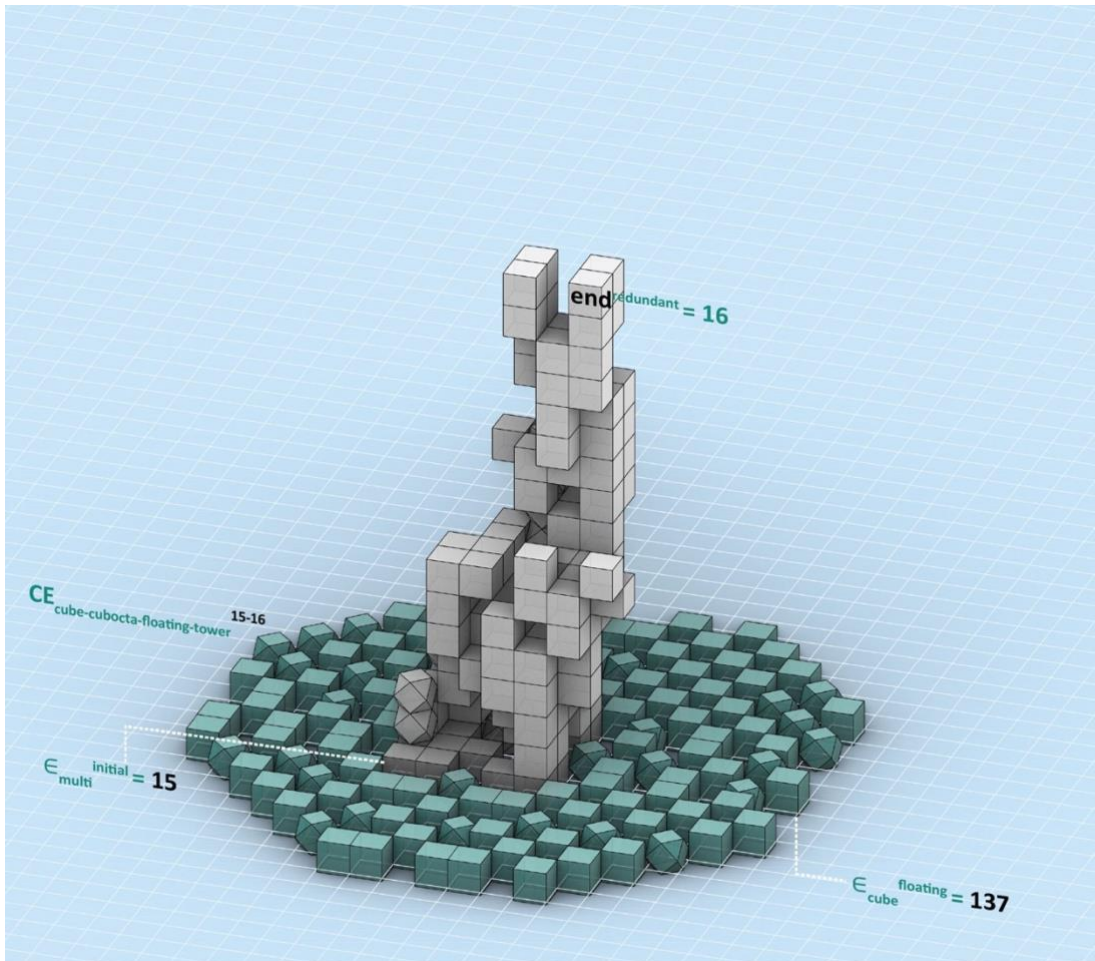


Figure 4.43 – Computational Design Prototype of a selected $CE_{cube-cubocta-floating-tower}$ with 15 $\epsilon_{cube}^{initial}$ that had a runtime of 16 t^{tick} before reaching the rest state. Model, algorithm, illustration, and graphics, by Author (December, 2019).

Whereas, fig. 4.44 shown below illustrates the outcome in the form of **Digital Fabrication** of the $CE_{cube-cubocta-floating-tower}$ demonstrated in its computational form in fig. 4.43 with $15 \epsilon_{multi}^{initial}$ generated during the '**Digital Fabrication Workshop**'. The digitally fabricated version photographed below has been considerably tweaked to convert edge conditions of ϵ_{cube} and $\epsilon_{cubocta}$ into 3D printable chords. However, care has been taken to keep the morphology intact.

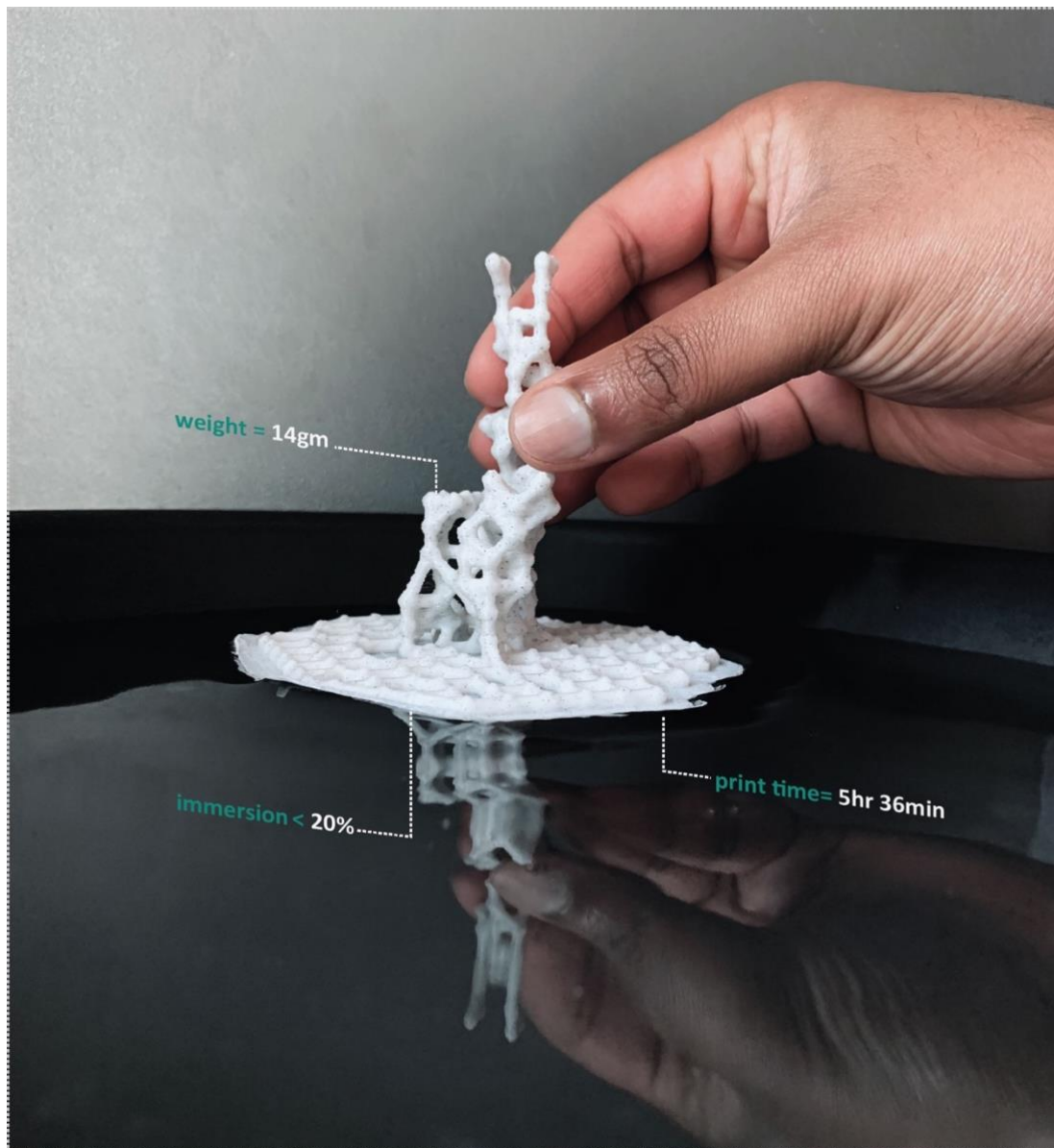


Figure 4.44 – Digital Fabrication Prototype of the selected $CE_{cube-cubocta-floating-tower}$ with $15 \epsilon_{cube}^{initial}$ that had a runtime of $16 t^{tick}$ before reaching the rest state. Model, algorithm, Illustration, graphics, and photograph by Author (December, 2019).

4.5.4 CE^{MEME}

Similar to the previous taxa (as explained in 4.2.4, 4.3.4 and 4.4.4), fig. 4.43 and fig. 4.44 sufficiently demonstrate the prototyping of the $CE_{cube-cubocta-floating-tower}$, however, more outcomes focusing on the taxonomy creation have been illustrated in the next chapter (5 | On the consequences of Computational Ecosystems) with further analysis comparing all the other taxa from the procedural sequences in chapter 6 (On the investigative analysis of Computational Ecosystems). However, as the CE^{MEME} serves as an amalgamation of the CE^{MESE} and the CE^{SEME} , the amendments made in its methodology should be addressed as the following –

- The multiplicity of both the ϵ and the Ψ parameters has dictated the CE^{MEME} to considerably simulate real-life built form, thus further moving away from the highly computational resemblance to the outcomes of 3D CA .
- Expanding on the previously assumed parameters for the different environments to multiple agents, the CE^{MEME} also demonstrates how a CE with higher complexity can be molded and programmed into performing the desired architectural intent.
- The introduction of the digital fabrication in prototyping by means of the successful results of the **Digital Fabrication Workshop** (more results in 5.4) provides a direct link in the workflow of Computational Design being used in unison with Digital Fabrication to form a CE that in turn serves as a link between the built form and the algorithm (thereby bypassing design).

Thus, the CE^{MEME} quite adequately handles the maximum complexity that could come out of the multiplicity in the distinct ϵ and Ψ entities (as long as the example scenario is followed). Although quite distinct and complex compared to the previous taxa, after methodically following the procedural sequences, similar outcomes could be generated.

4.6 Procedural sequences for Computational Ecosystems

All the four taxa - CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME} which were explicitly theorized, taxonomized, evaluated and fabricated as explained thus far, have undergone the primary objectives – **Case studies**, **Simulations**, and **Prototyping** in this chapter. Although each taxon has had a slightly modified, bespoke version for the entire methodology, following are the key methodological milestones in the **Architecture of Computational Ecosystems** so far –

- **Case Studies** – Documenting similar ecosystems (preferably observed in nature) with participating agents governed by simple rules that could be simulated computationally.
- **Analogical Assessment** – Identifying exact and precise analogies in terms of the ϵ and Ψ parameters between the case study and the desired **CE**. Also, identifying the taxon that the **CE** falls in (CE^{SESE} , CE^{MESE} , CE^{SEME} , or CE^{MEME}).
- **UML Class Diagram** – Establishing UML Class diagram for all the identified ϵ and Ψ parameters, and their specific morphological parameters, considering which will be user determined and which will be context dependent.
- **CA Identification** – Assessing all the different classes determined in the UML Class Diagram and identifying the appropriate **CA** model that could be relevant to perform the **CE**.
- **ϵ and Ψ entities** – Setting up **Neighbourhood Conditions** for all the specific ϵ entities and **State Conditions** for all the distinct Ψ entities identified in the UML Class diagram. A detailed illustration providing the various Ψ_{spawn} and Ψ_{cull} conditions for specific N_{check} and N_{form} routines is particularly helpful.

- **UML Sequence Diagram** - Establishing UML Sequence diagram for the distinct ϵ and Ψ entities determining their distinct roles, interactions and runtimes as compared to the sequence of the entire **CE**. Here, the role, interaction and runtime of the user and the introduction of possible partial intermediate **CE** can also be elaborated.
- **Computational Prototyping** – Simulating all possible initial conditions with a lower number of initial ϵ entities but for the entire runtime and rest states of the **CE**. Here, several iterations are spontaneously generated and can be used to offer an initial evaluation of the **CE**.
- **Computational Evaluation** – Evaluating the **CE** with 3rd party testers (in case of the research, this objective was achieved by conducting several workshops as mentioned previously) for bugs, redundancies and code compliances. A versioning of the **CE** could also be done at this stage if required.
- **Production Evaluation** – Eventually perform digital fabrication by means of predetermined production technique thereby testing the **CE** as a feedback loop between the **Built form** and the **Algorithm**.

Although the above stated methodological milestones are crucial in establishing any **CE**, the above list is not exhaustive and several more steps can be appended or removed as required by the user.

Results

5 | On the consequences of the Computational Ecosystems

Just like the previous chapter that defines, exemplifies and illustrates the procedural sequences for all the taxa by focusing on the methodology of this research, this chapter explicitly illustrates the different outcomes of the all the taxa by focusing on the results of this research. The structure of the thesis (refer 1.3.2), specifically choses to combine the methodology, results and discussions for all the taxa into separate chapters rather than dealing with separated methodologies, results and discussions per taxon. The decision comes with the following advantages –

- In this way, the methodologies, results and discussions can share the common semantics across taxa, without having to repeat the concepts several times.
- Apart from reducing redundancies, this also showcases the intuitive thought processes and computational decisions that were taken before establishing each taxon, as the complexity progresses.
- Moreover, when all the taxa are combined together, especially in the results and discussion sections, the outcomes and analysis for each taxon can be done comparatively while understanding the benefits of one over the other in a systematic manner.
- Finally, it serves as an analogy to the timeline adopted at the beginning of the research, where in the operational objectives of the research, all the taxa were theorized, taxonomized and prototyped simultaneously.

Apart from showcasing the results, which are absolutely graphical in nature, this chapter also introduces some folksonomies (user tagging system as explained in 3.2.1) that are associated to all the specific iterations for each of the predetermined 4 taxa – CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME} .

5.1 Results of CE^{SESE}

As the prototypical results of a $CE_{cube-tower}$ have already been illustrated in fig. 4.11 and fig. 4.12, the following fig. 5.1 and fig. 5.2 illustrate a few selected iterations of a $CE_{cube-tower}$ with $12 \text{ €}_{cube}^{initial}$. As described previously, the figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), and the rest of the €_{cube} in the $CE_{cube-tower}$ thus denoted by ■ (R,G,B – 204,204,204) colored cube.

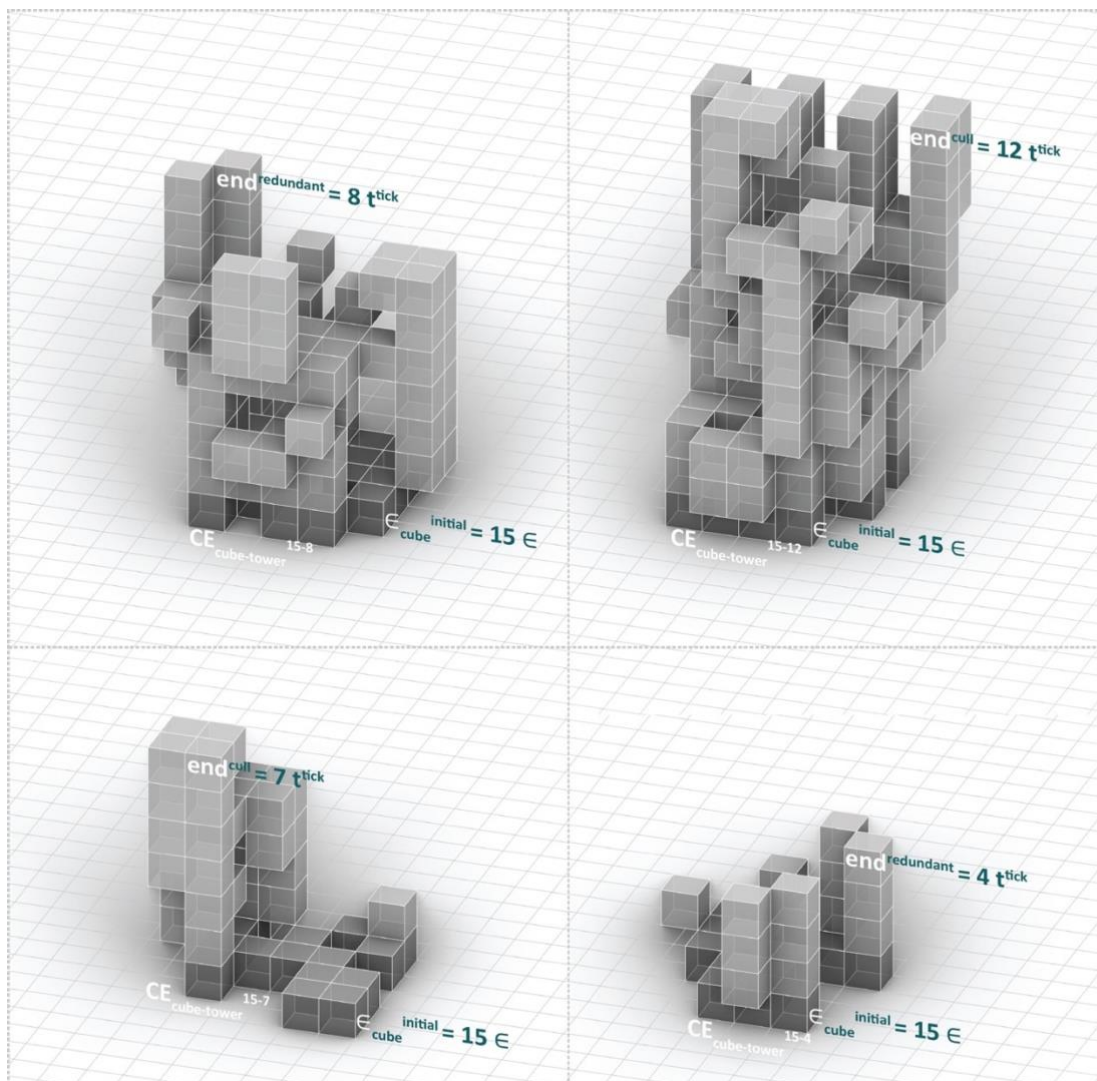


Figure 5.1 – Selected iterations of $CE_{cube-tower}$ with $15 \text{ €}_{cube}^{initial}$ with varying runtimes until individual rest state are reached. Model, algorithm, Illustration and graphics by Author (June 2018).

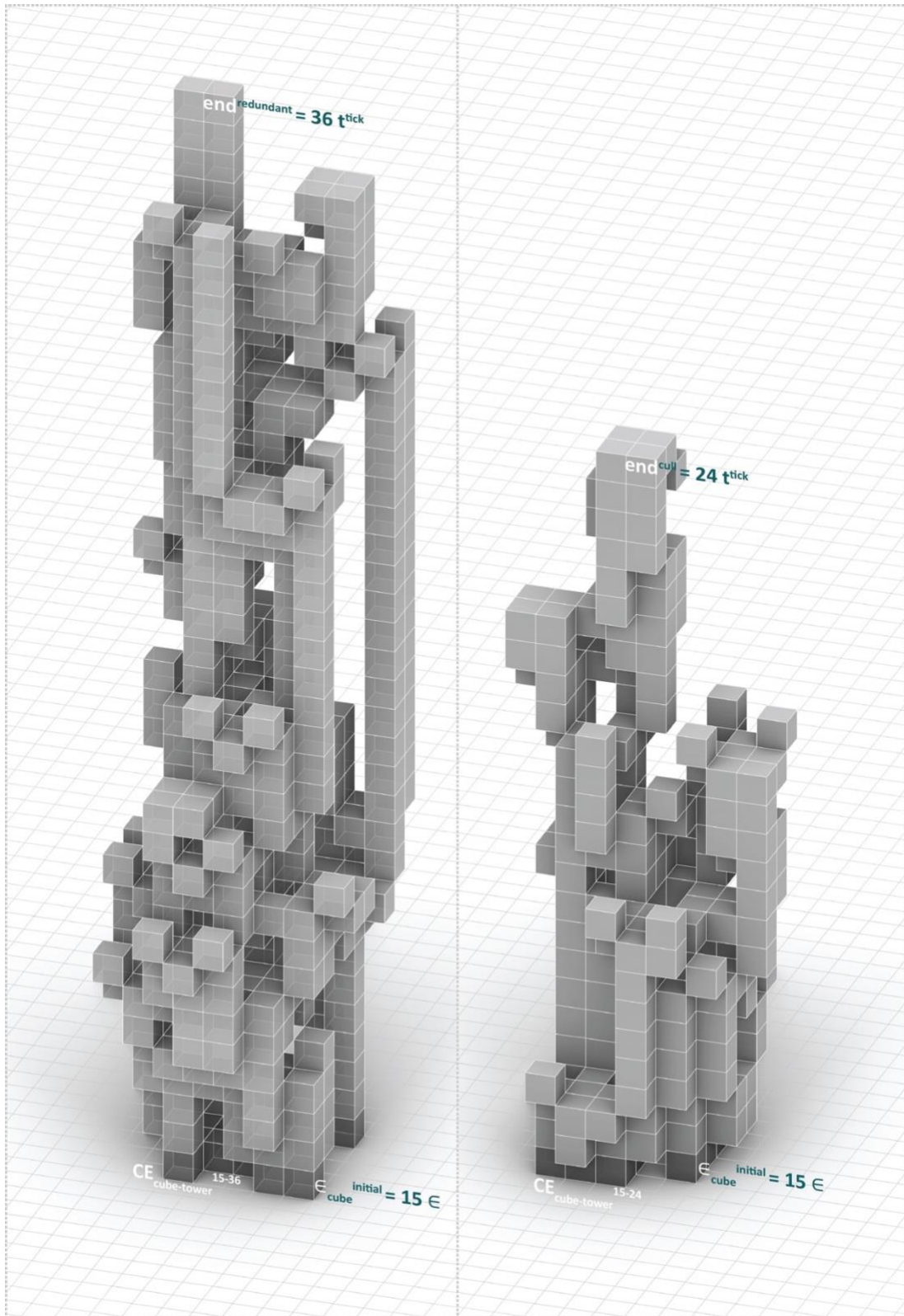


Figure 5.2 – Selected iterations of $CE_{cube-tower}$ with $15 \text{ €}_{cube}^{initial}$ with varying runtimes until individual rest state are reached. Model, algorithm, Illustration and graphics by Author (June 2018).

The 6 iterations of $CE_{cube-tower}$ illustrated in fig. 5.1 and fig. 5.2 clearly demonstrate how the different $\epsilon_{cube}^{initial}$ conditions affect the morphology of the $CE_{cube-tower}$. Here, all the $CE_{cube-tower}$ begin with 15 ϵ_{cube} when the designer initiates the algorithm however, depending on the N_{check} and N_{form} routines, and influenced by $CE_{cube-tower}$ rest conditions, they terminate in a wide range of shapes. Following are some observations for all the individual $CE_{cube-tower}$ illustrated in fig. 5.1 and fig. 5.2 –

- $CE_{cube-tower}^{15-8}$ – With 100 ϵ_{cube} , this $CE_{cube-tower}$ halts at the 8th level due to the **end^{redundancy}** condition. As it has a balanced combination of regular and irregular assemblages, it can be **#hospitality** or **#commercial** architecture.
- $CE_{cube-tower}^{15-12}$ – With 165 ϵ_{cube} , this $CE_{cube-tower}$ halts at the 12th level due to the **end^{cull}** condition. Because of some highly ordered assemblages, it can be used as a **#mixed-use** building for **#residential** and **#commercial** architecture.
- $CE_{cube-tower}^{15-7}$ – With 46 ϵ_{cube} , this $CE_{cube-tower}$ halts at the 7th level due to the **end^{cull}** condition. Due to the availability of a stark contrast in volumes it is ideal for **#hospitality**, **#healthcare** or **#educational** architecture.
- $CE_{cube-tower}^{15-4}$ – With a mere 29 ϵ_{cube} , this $CE_{cube-tower}$ manages only to rise up to 4 levels before it is halted by the algorithm owing to the **end^{redundancy}** condition. It can hardly be of any use in any architectural application.
- $CE_{cube-tower}^{15-24}$ – With 259 ϵ_{cube} , this $CE_{cube-tower}$ halts at the 24th level as it reaches the **end^{cull}** condition. Due to its decent height and a good proportion of irregular ϵ_{cube} stacking, it is ideal for **#high-density #residential** towers.
- $CE_{cube-tower}^{15-24}$ – The tallest in this selection with a whopping 507 ϵ_{cube} , it halts at the 36th level due to the **end^{redundancy}** condition. Owing to its considerable height and some highly ordered assemblages, it can also be used as a **#mixed-use** building for **#residential** and **#commercial** architecture.

5.2 Results of CE^{MESE}

As the prototypical results of a $CE_{cube-cubocta-tower}$ have already been illustrated in fig. 4.24 and fig. 4.25, the following fig. 5.3 and fig. 5.4 illustrate a few selected iterations of several $CE_{cube-cubocta-tower}$ with $15 \text{ } \epsilon_{multi}^{initial}$. As described previously, the figure shows the initial state denoted by ■ colored cube (R,G,B – 129,129,129), and the rest of the tower thus denoted by ■ (R,G,B – 204,204,204) colored cube.

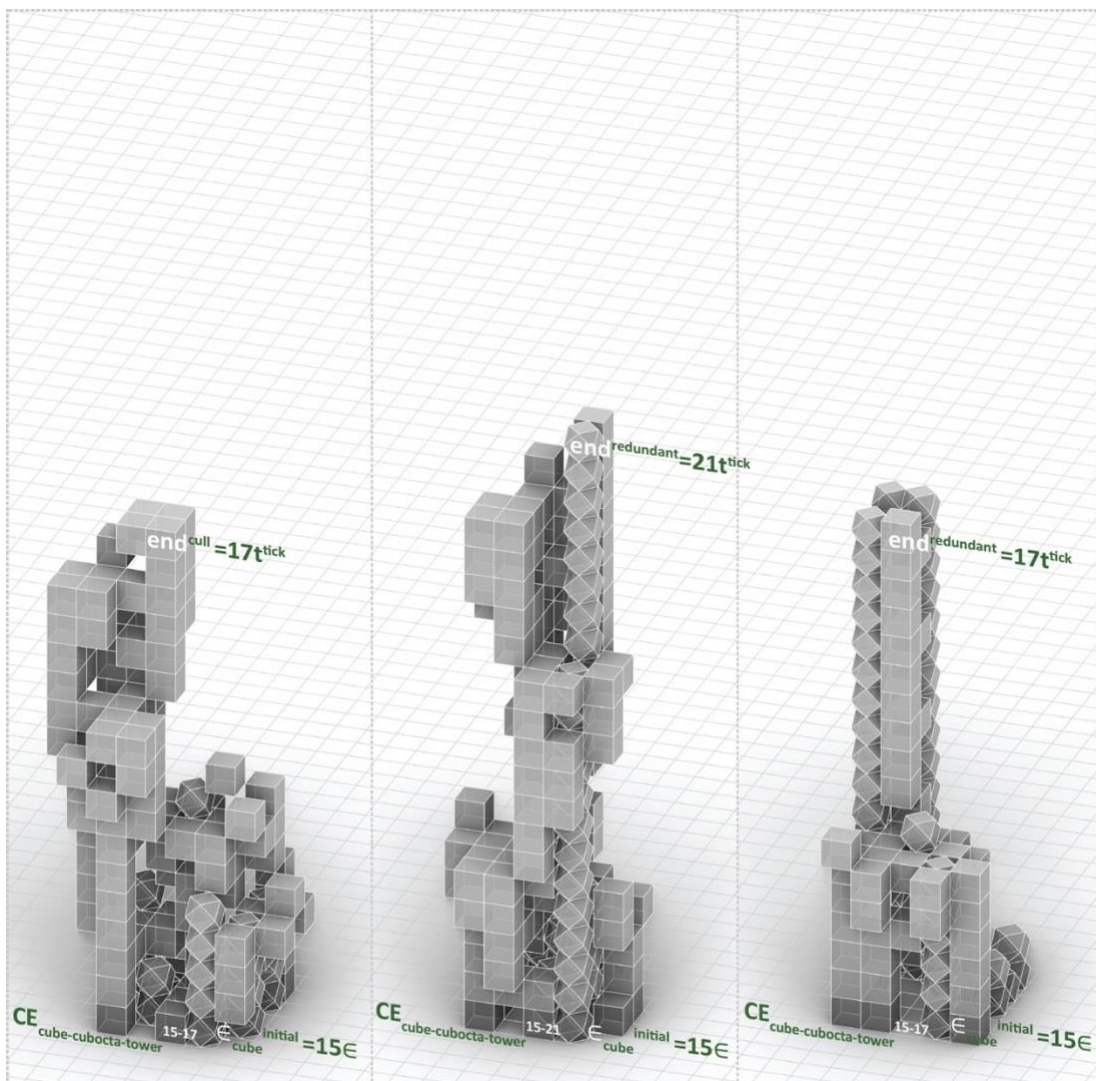


Figure 5.3 – Selected iterations of $CE_{cube-cubocta-tower}$ with $15 \text{ } \epsilon_{cube}^{initial}$ with varying runtimes until individual rest state are reached. Model, algorithm, Illustration and graphics by Author (July 2019).

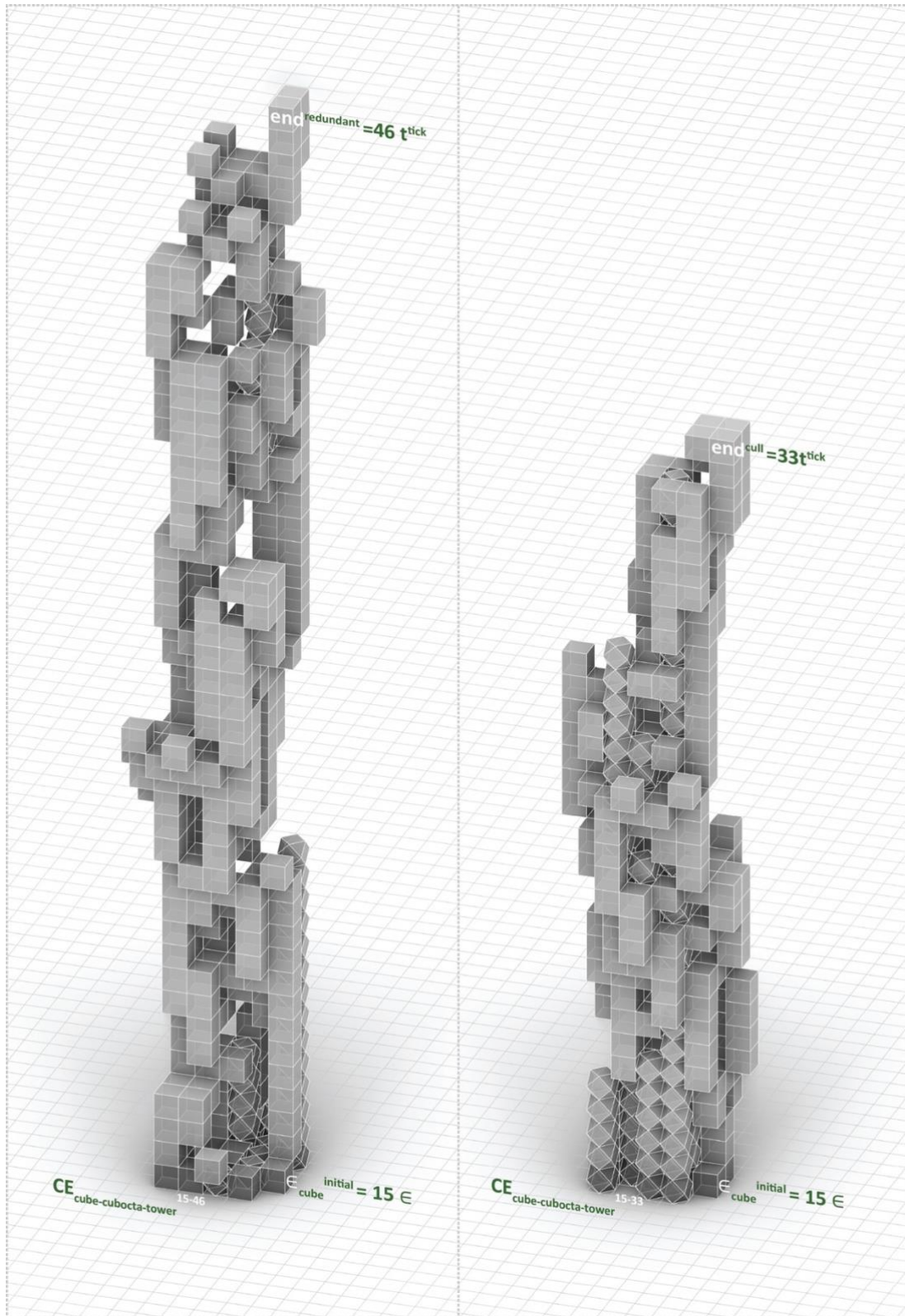


Figure 5.4 – Selected iterations of $CE_{cube-cubocta-tower}$ with $15 \text{ €}_{cube}^{initial}$ with varying runtimes until individual rest state are reached. Model, algorithm, Illustration and graphics by Author (July 2019).

The 5 iterations of $CE_{cube-cubocta-tower}$ illustrated in fig. 5.3 and fig. 5.4 demonstrate how the different $\epsilon_{cube}^{initial}$ conditions affect the morphology of the $CE_{cube-cubocta-tower}$. Here, all the $CE_{cube-cubocta-tower}$ begin with 15 ϵ_{cube} when the designer initiates the algorithm however, depending on the N_{check} and N_{form} routines, and influenced by the $CE_{cube-cubocta-tower}$ rest conditions, they terminate in varied shapes. Following are some observations for all the individual $CE_{cube-cubocta-tower}$ illustrated in fig. 5.3 and fig. 5.4 –

- $CE_{cube-cubocta-tower}^{15-17}$ – With 143 ϵ_{multi} , this $CE_{cube-cubocta-tower}$ halts at the 17th level due to the **end^{cull}** condition. Owing to a bulk of aggregation until the 8th level, and an isolated tower, this $CE_{cube-cubocta-tower}$ could be implemented as a **#mixed-use** building for **#residential** and **#commercial** architecture.
- $CE_{cube-cubocta-tower}^{15-21}$ – With 170 ϵ_{multi} , this $CE_{cube-tower}$ halts at the 21st level due to the **end^{redundant}** condition. Because of some ordered assemblages, it can be a **#mixed-use** building for **#residential** and **#commercial** architecture.
- $CE_{cube-cubocta-tower}^{15-17}$ – With 133 ϵ_{multi} , this $CE_{cube-cubocta-tower}$ halts at the 17th level due to the **end^{redundant}** condition. Owing to its slender verticality after the 6th level and the availability of a lot of open space on the 6th level, this CE could be implemented as an **#Urban #Hotel** architecture.
- $CE_{cube-cubocta-tower}^{15-46}$ – The tallest in this selection with 377 ϵ_{multi} , it halts at the 46th level due to the **end^{redundant}** condition. Owing to its considerable height and some highly ordered assemblages, it can also be used as a **#mixed-use** tower for **#residential**, **#commercial** and **#hospitality** architecture.
- $CE_{cube-cubocta-tower}^{15-33}$ – Another example of vertical growth, this tower with 329 ϵ_{multi} , halts at the 33rd level due to the **end^{cull}** condition. Owing to its decent height and a healthy blend of the ϵ_{multi} and ϵ_{multi} entities while retaining enough built form, this $CE_{cube-cubocta-tower}$ can be very effectively used as a **#mixed-use** tower for **#residential**, **#commercial** and **#hospitality**.

5.3 Results of CE^{SEME}

As the prototypical results of a $CE_{cube-floating-tower}$ have already been illustrated in fig. 4.36 and fig. 4.37, the following fig. 5.5 and 5.6 illustrate a few selected iterations of several $CE_{cube-floating-tower}$ with 15 $\epsilon_{multi}^{initial}$. The figure shows the initial state denoted by ■ (R,G,B – 129,129,129), the rest of the ϵ_{cube}^{tower} denoted by ■ (R,G,B – 204,204,204) colored cube, and the $\epsilon_{cube}^{floating}$ denoted by ■ (R,G,B – 207,235,255).

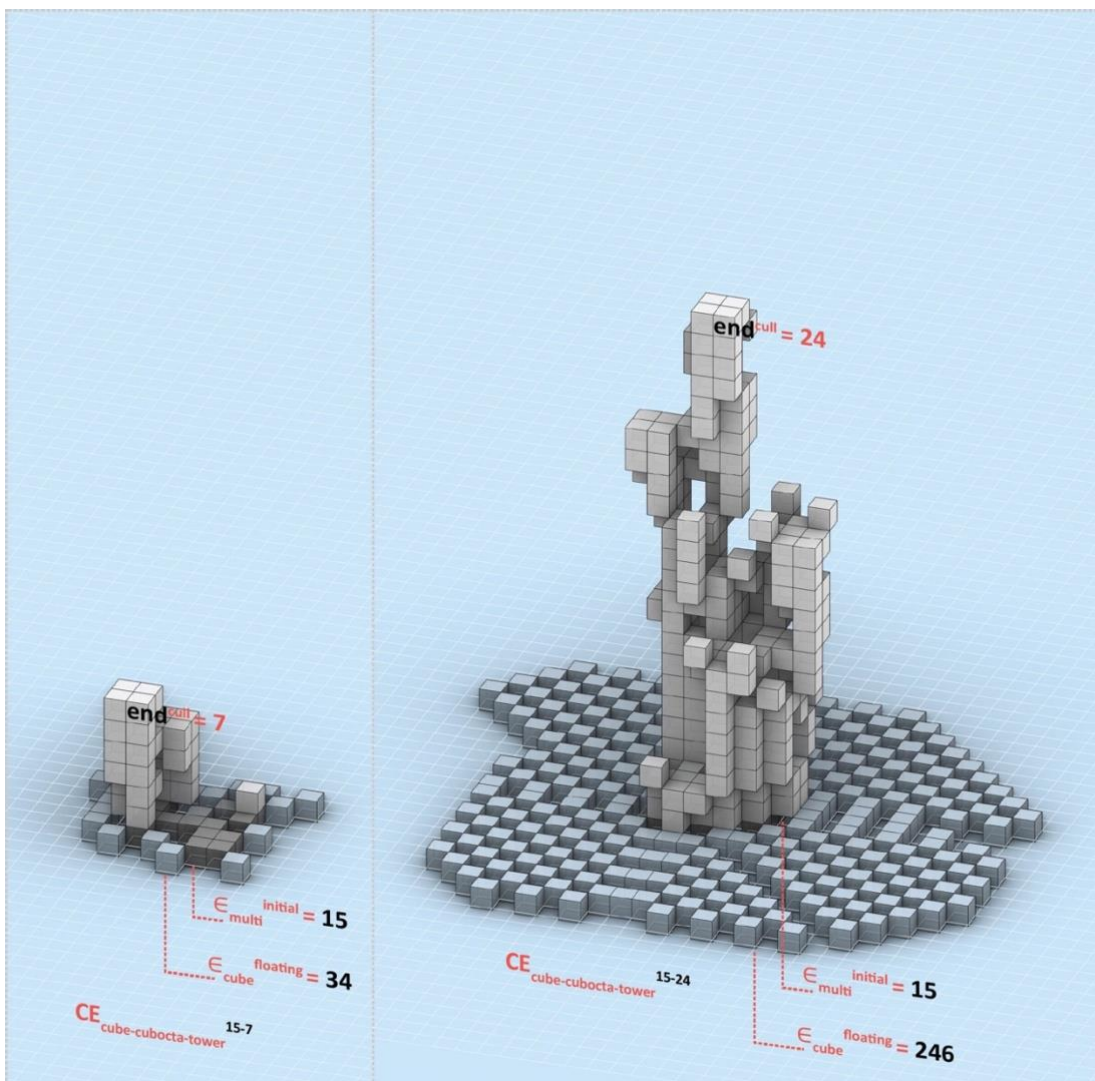


Figure 5.5 – Selected iterations of $CE_{cube-floating-tower}$ with 15 $\epsilon_{cube}^{initial}$ with varying runtimes until individual rest state are reached. Model, algorithm, Illustration and graphics by Author (July 2019).

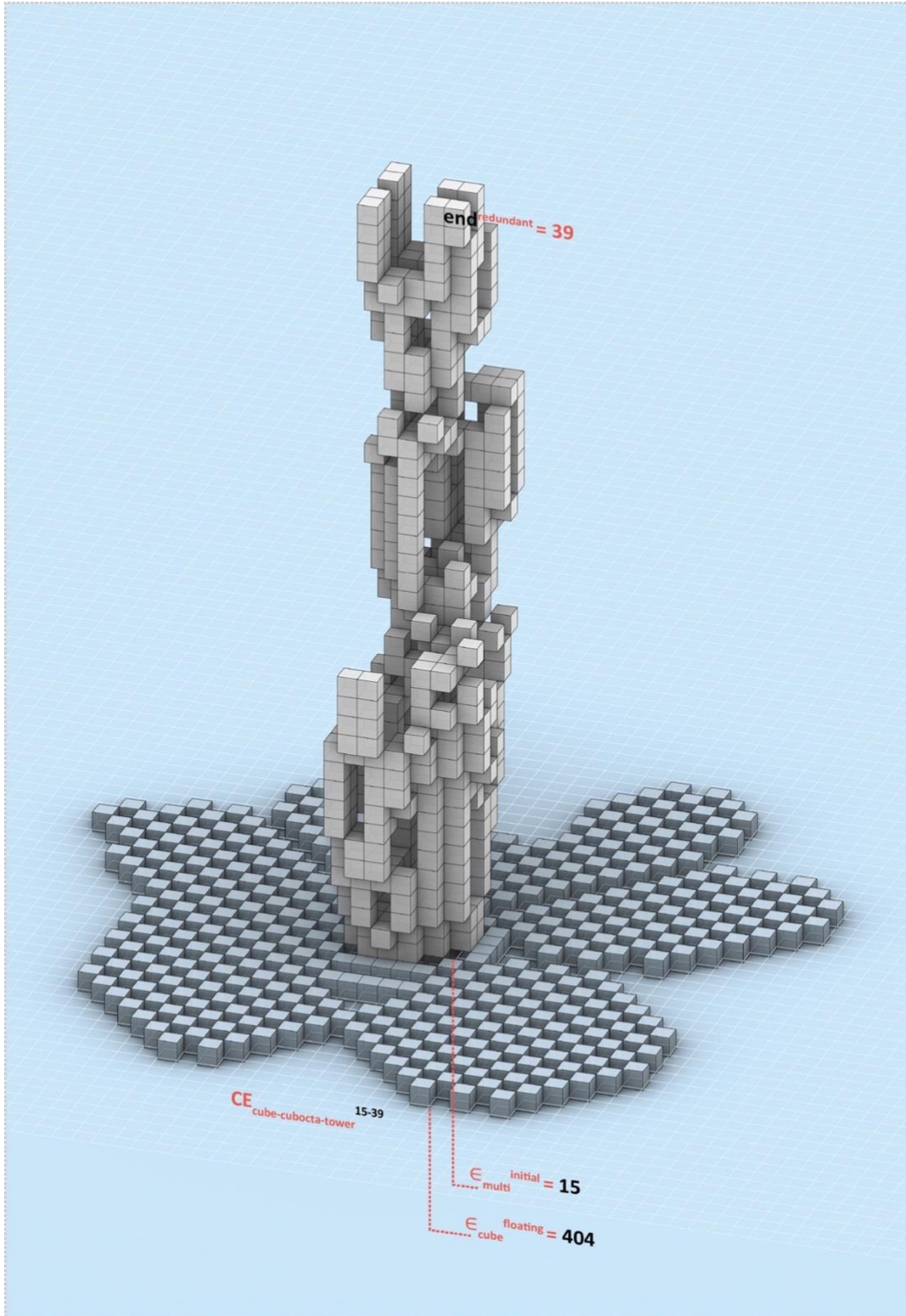


Figure 5.6 – Selected iterations of $CE_{cube-floating-tower}$ with 15 $\epsilon_{cube}^{initial}$ with varying runtimes until individual rest state are reached. Model, algorithm, Illustration and graphics by Author (July 2019).

The 5 iterations of $CE_{cube-floating-tower}$ illustrated in fig. 5.5 and fig. 5.6 demonstrate how the different $\epsilon_{multi}^{initial}$ conditions affect the morphology of the $CE_{cube-floating-tower}$. Here, all the $CE_{cube-floating-tower}$ begin with 15 $\epsilon_{multi}^{initial}$ when the designer initiates the algorithm, however, depending on the N_{check} and N_{form} routines, and influenced by the $CE_{cube-floating-tower}$ **rest** conditions, they terminate in a wide range of shapes. Following are some observations for all the CE illustrated in fig. 5.5 and fig. 5.6 –

- $CE_{cube-floating-tower}^{15-7}$ – With 46 ϵ_{cube}^{tower} and 34 $\epsilon_{cube}^{floating}$ this $CE_{cube-floating-tower}$ halts at the 7th level due to the **end^{cull}** condition. And to counter the weight of the 46 ϵ_{cube}^{tower} elements it spawns 19 additional $\epsilon_{cube}^{floating}$ elements to keep the tower afloat. Owing to a large open foreground that could serve as a docking station, this $CE_{cube-cubocta-tower}$ with a very organized and contained tower could be treated as a **#healthcare** or **#research** facility.
- $CE_{cube-floating-tower}^{15-24}$ – With an organized distribution of 259 ϵ_{cube}^{tower} elements forming the tower, and 246 $\epsilon_{cube}^{floating}$ elements as part of the floating raft, this $CE_{cube-floating-tower}$ rises up to 24 levels after halting due to the **end^{cull}** condition. The additional $\epsilon_{cube}^{floating}$ elements that it spawns are evenly distributed on all the sides of the tower. Because of several segregated assemblages, this $CE_{cube-cubocta-tower}$ could serve as several **#residential** towers.
- $CE_{cube-floating-tower}^{15-39}$ – With a whopping 520 ϵ_{cube}^{tower} elements in the tower and an additionally spawned 389 $\epsilon_{cube}^{floating}$ elements, this $CE_{cube-floating-tower}$ halts at the 39th level due to the **end^{redundant}** condition. This $CE_{cube-cubocta-tower}$ has an uncommon combination of assemblages that can trifurcate the entire tower. The bulky but organized base can be concentrated with **#commercial** activity. The central part branches into the tower and open space, where the tower can be **#residential** or **#offices**, and the open space can serve as an **#elevated #park**. Also, the raft has a unique assemblage that allows a lot of interesting niches that could be used to accommodate a plenty of **#public** and **#infrastructural** activities.

5.4 Results of CE^{MEME}

As the prototypical results of a $CE_{cube-cubocta-floating-tower}$ have already been illustrated in fig. 4.43 and fig. 4.44, the following fig. 5.7 and 5.8 illustrate selected iterations of the $CE_{cube-cubocta-floating-tower}$ with 15 $\epsilon_{multi}^{initial}$. The figure shows the initial state denoted by ■ (R,G,B – 129,129,129), the rest of the ϵ_{multi}^{tower} denoted by ■ (R,G,B – 204,204,204), and the $\epsilon_{multi}^{floating}$ denoted by ■ (R,G,B – 207,235,255).

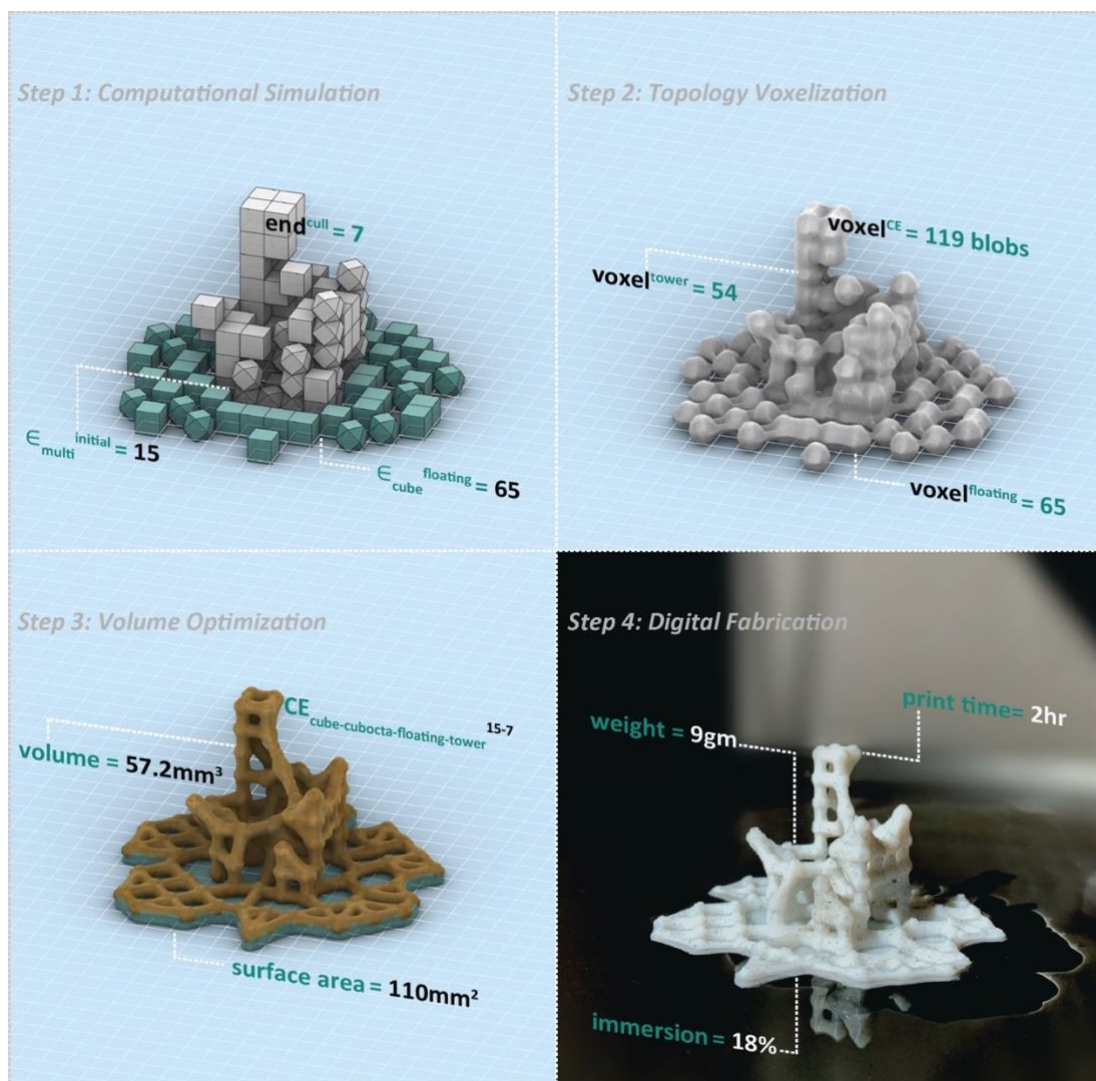


Figure 5.7 – Process of selected $CE_{cube-cubocta-floating-tower}$ with 15 $\epsilon_{multi}^{initial}$ being transformed for Digital Fabrication. Model, algorithm, Illustration and Photograph by Author (December 2019).

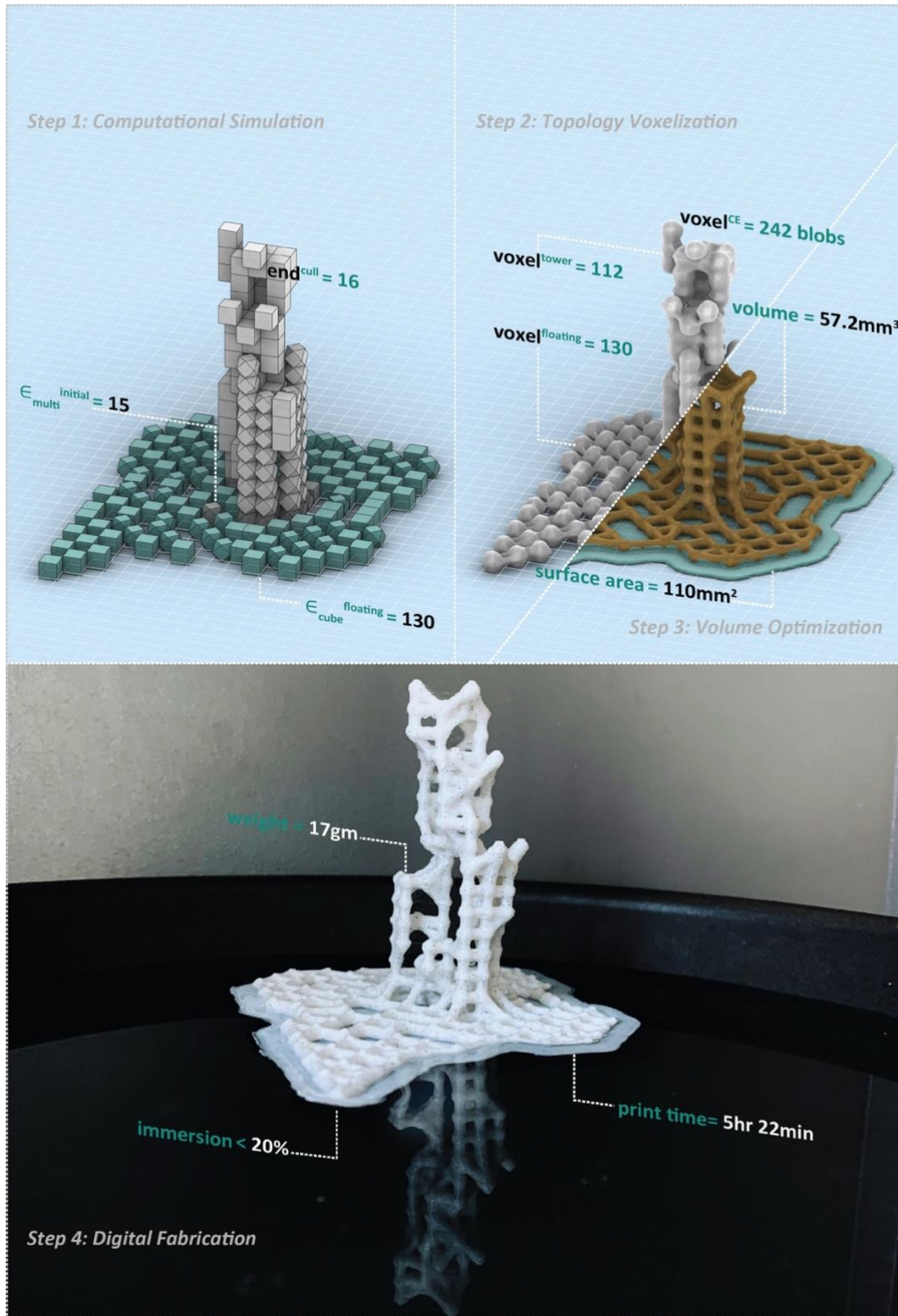


Figure 5.8 – Selected 3D printed iteration of $\text{CE}_{\text{cube-cubocta-floating-tower}}$ with 15 $\epsilon_{\text{multi}}^{\text{initial}}$ until individual rest state is reached. Model, algorithm, Illustration and graphics by Author (December 2019).

The iterations of $CE_{\text{cube-cubocta-floating-tower}}$ illustrated in fig. 5.7 and 5.8 demonstrate how the different $\epsilon_{\text{multi}}^{\text{initial}}$ conditions affect the morphology of the CE . Both the CE begin with 15 $\epsilon_{\text{multi}}^{\text{initial}}$ when the designer initiates the algorithm, but depending on the N_{check} and N_{form} routines, and influenced by the $CE_{\text{cube-cubocta-floating-tower}}$ rest conditions, they terminate into different results. Apart from illustrating the outcomes, however, fig. 5.7 elaborates the process of transforming an existing, performing and completed CE into a 3D-printable object in order to perform the evaluations in real-physical environments. The steps taken for the entire process are elaborated as following –

- **Step 1 – Computational Simulations** – Here, the finished result at the Final End Condition (or Early End Condition, although the former is preferred) of the CE is outputted and assimilated with all its specific components.
- **Step 2 – Topology Voxelization** – Here, all the ϵ_{multi} are considered merely as points (centered on the individual ϵ_{multi} agents). These points are then processed into forming voxels (3D pixels) by making sure that each voxel replicates the surface area of the original ϵ_{multi} agent.
- **Step 3 – Volume Optimization** – In this step the voxelization is further reduced to form optimum structures while making sure that ratio of the volume to bottom surface of are of the entire CE remains the same.
- **Step 4 – Digital fabrication** – After converting the volume optimization into a 3D printer tool path (G-code), the finished print is weighed, and evaluated. Like in the case of a $CE_{\text{cube-cubocta-floating-tower}}$ it is tested to ensure vertical spanning strength (by FDM printing without supports) and buoyancy (by dropping the print in water and checking its immersion percentage).

These four steps have also been used for the result shown in fig. 5.8, and any other prototype of $CE_{\text{cube-cubocta-floating-tower}}$ that would be done.

5.5 Conclusive thoughts on the results of all the taxa

Having illustrated some selected results for the prototyped executions of all the four taxa thus far, the research thus concludes the execution of the procedural sequences in performing the primary objectives for the CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME} . The chapter in this thesis also concludes the illustration of outcomes for the selected initial conditions.

Fig. 5.1 to fig. 5.8 help in illustrating some of the key findings in all the taxa, which will be elaborated upon by comparing the results across taxa in the next chapter (as in 6 | On the investigative analysis of the Computational Ecosystems). These findings can be defined as the following:

- Different morphologies of the outcomes are governed by the following factors –
 - The $\mathbf{\epsilon}$ entities – Naturally, the physical properties and the various assemblages of the $\mathbf{\epsilon}$ entities (either biotic, or abiotic, or both).
 - The $\mathbf{\Psi}$ entities – The distinct rule sets which in turn dictate the state conditions in the form of the $\mathbf{\Psi}$ entities (both distinct and symbiotic).
 - The Neighbourhood (\mathbf{N}) – One of the very important aspects of the computational environment which is the \mathbf{N} condition for each $\mathbf{\epsilon}$ and $\mathbf{\Psi}$.
 - The Initial State – By far the most divisive parameter, which is capable of generating various iterations within a predetermined taxon with all the above parameters ($\mathbf{\epsilon}$, $\mathbf{\Psi}$, and \mathbf{N}) unchanged.

- Depending on the predetermined ϵ and ψ entities, a **CE** can be drastically varied to accommodate the purpose by varying the multiplicity of one or both of the ϵ and ψ entities.
- Moreover, although one of the ϵ and ψ entities is singular, the multiplicity of the latter will amount to a certain degree of multiplicity in the former.
- The ϵ entities are semantically capable of being variable parameters for a wide range of agents such as geometries, topologies, and organisms as demonstrated in the example scenarios.
- The ψ entities are semantically capable of being variable parameters for a wide range of functions such as structural assemblages, geological conditions, and even fabrication constraints as demonstrated in the example scenarios.

Observations

6 | On the investigative analysis of the Computational Ecosystems

Having theorized, simulated, taxonomized, versioned, and prototyped the proof of concept for all the four taxa – CE^{SESE} , CE^{MESE} , CE^{SEME} , and CE^{MEME} , the thesis can now elaborate upon the specific inferences generated by comparing all the outcomes and results of these four taxa against each other. It can now also be discussed how these inferences can be implemented in developing several more CE (conforming to either of the four taxa theoretically and semantically).

Thus, for the ease of comprehension this chapter (which serves as a conclusive end to all the possibilities within the four taxa, before diving into the conclusions and projections for CE) elaborates upon the similarities and differences across the four taxa under two different conditions of variability which can be described as –

- By considering the same initial state for different ϵ , Ψ , and N – Comparing the results for the same initial state while ϵ , Ψ , and N vary as per considerations of distinct taxa.
- By considering different initial states for the same ϵ , Ψ , and N – Comparing the results for different initials state while ϵ , Ψ , and N are considered for the already established CE^{MEME} – $CE_{cube-cubocta-floating-tower}$.

Moreover, this chapter shows how CE (built on the computational framework of CA) could perform as a dynamic, reciprocal, symbiotic feedback loop while having

A **built form** that was constructed, monitored and governed by an autonomous, unbiased **algorithm**

and an **algorithm** that was dynamically constructed, monitored and governed by the **built form**.

6.1 Effects of distinct ϵ , Ψ , and N

This thesis has already illustrated several selected **CE** for the example scenarios developed as part of the procedural sequences for all the four distinct taxa (chapters 4| and 5|). However, the results of these specific initial states were chosen for their distinct outcomes which were demonstrating possibilities to be implemented in a varied range of architectural typologies. Some, of these outcomes were also specifically chosen to demonstrate specific physical properties such as the tallest towers, towers with maximum number of ϵ_{multi} , or **CE** with specific assemblages useful for specific architectural typologies.

A comparative analysis of different **CE** with distinct ϵ , Ψ , and N considerations for the same initial states, however, has not been performed yet, and this section elaborates on the same. Fig. 6.1 shown alongside, illustrates the construction of four **CE** – **CE**_{cube-tower} (as the **CE**^{SESE}), **CE**_{cube-cubocta-tower} (as the **CE**^{MESE}), **CE**_{cube-floating-tower} (as the **CE**^{SEME}), and **CE**_{cube-cubocta-floating-tower} (as the **CE**^{MEME}) initiated with the same initial state with 15 $\epsilon_{\text{multi}}^{\text{initial}}$. Although semantically speaking, not all the taxa can share the same initial state due to the multiplicity of the ϵ , Ψ , and N considerations. However, the **CE**^{SESE}, and **CE**^{SEME} (due to their ϵ entity singularity) could have the same $\epsilon_{\text{multi}}^{\text{initial}}$ conditions. Similarly, the **CE**^{MESE}, and **CE**^{MEME} (due to their ϵ entity multiplicity) could have the same $\epsilon_{\text{multi}}^{\text{initial}}$ conditions. Although this is not true for all the **CE**^{SESE} – **CE**^{SEME} and **CE**^{MESE} – **CE**^{MEME} combinations always, but for the example scenarios (elaborated in 4|), as they have cube-cuboctahedra combinations.

Considering the results illustrated in fig. 6.1, it can be inferred that the predefined distinct ϵ , Ψ , and N considerations for different **CE** can be controlled to generalize a purpose for the **CE**, thus forming a crucial link between the **built form** and the **algorithm**. Conversely, the results also infer, that the same initial state can be subjected to accommodate multiple, predefined, ϵ , Ψ , and N considerations and create multiple **CE**.

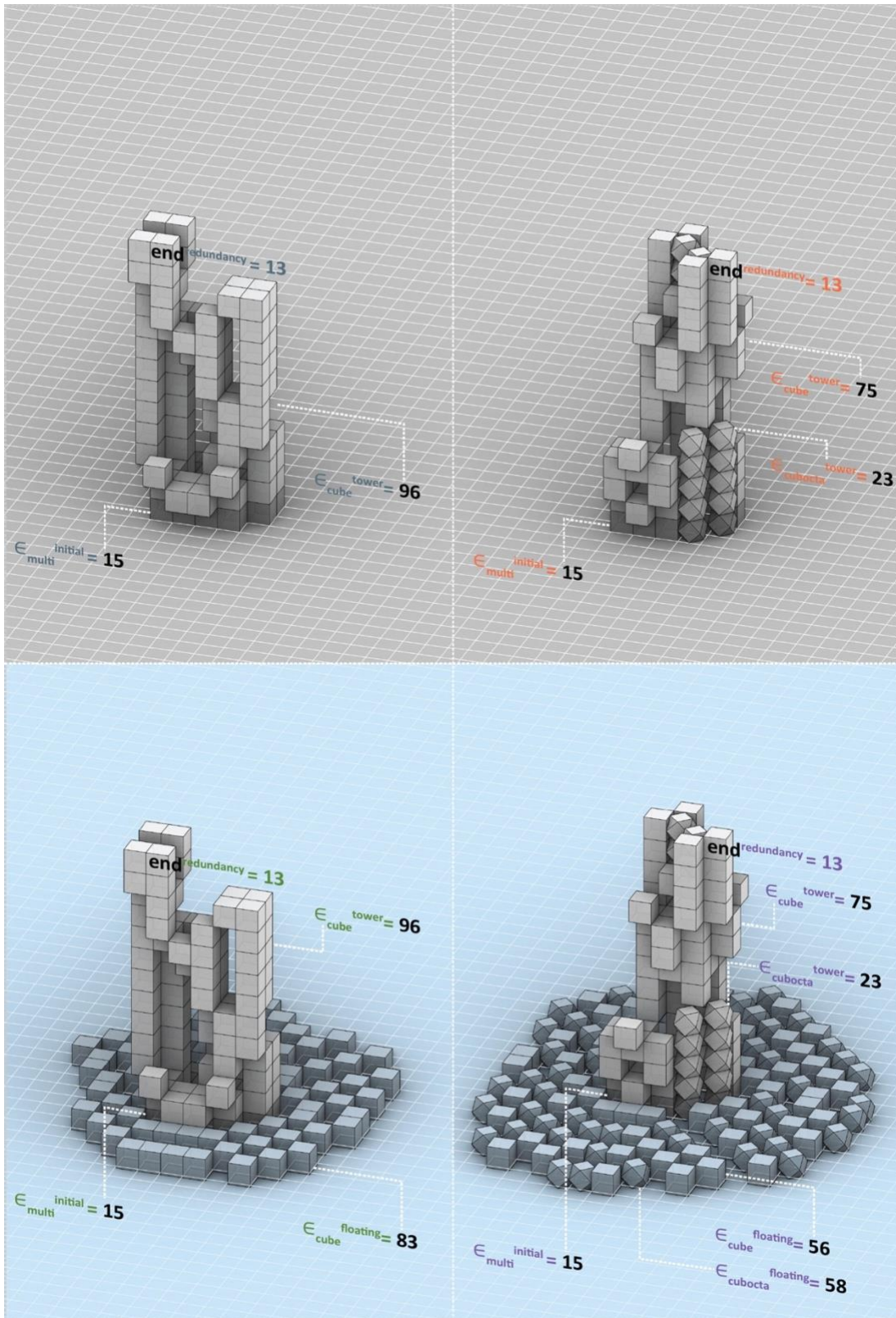


Figure 6.1 – Four different **CE** with the same $\epsilon_{\text{multi}}^{\text{initial}}$ conditions, but different ϵ , Ψ , and **N** considerations. Model, algorithm, Illustration and graphics by Author (Jan 2020).

6.2 Examples of possible distinct ϵ , Ψ , and N for CE

The ability of a CE to be manipulated differently for distinct ϵ , Ψ , and N considerations while conforming to the same (or similar) initial state ($\epsilon_{\text{multi}}^{\text{initial}}$) illustrated in fig. 6.1 can be used judiciously in the physical world (that is in the sense of actuality in non-computational environment) while generating and maintaining a feedback loop of structural coupling between the **built form** and the **algorithm**.

It is standard practice in the AEC community to conform to building bye-laws that are specific to every site (in terms of the urban/suburban/rural governing authority). Often these constraints restrict the site condition to follow a certain shape or form, which considerably changes the overall design strategy to accommodate the typology and counter other context specific issues. The intelligence employed by the architect into solving these issues, comes quite intuitively and cannot be replaced by any CE that aspires to replace the design between the built form and the algorithm (as explained in 1.1.4), however, some of these solutions could be programmed by intuitively varying the ϵ , Ψ , and N considerations (as elaborated in 4.6) while conforming to the same restrictions on the shape and form of sites (as in an unchanged $\epsilon_{\text{multi}}^{\text{initial}}$).

Fig. 6.2 illustrates how the results demonstrated in fig. 6.1, can be envisioned as different architectural solutions conforming to the same site conditions. While the $CE_{\text{cube-tower}}$ (as the CE^{SESE}) becomes a **#Residential** tower, the $CE_{\text{cube-cubocta-tower}}$ (as the CE^{MESE}) is envisioned as a **#Floating #Residential** tower accommodated within the same site boundaries surrounded by a hypothetical Ψ access street. The multiplicity in the Ψ entities of the $CE_{\text{cube-floating-tower}}$ (as the CE^{SEME}), and $CE_{\text{cube-cubocta-floating-tower}}$ (as the CE^{MEME}) shown in fig. 6.1 become **#mixed-use #residential #commercial** tower and **#floating #mixed-use #residential #commercial** tower respectively. CE^{MESE} and CE^{MEME} as their dual environments conform to the same site boundaries as shown in the figure. However, the boundaries of their floating pods are quite different.

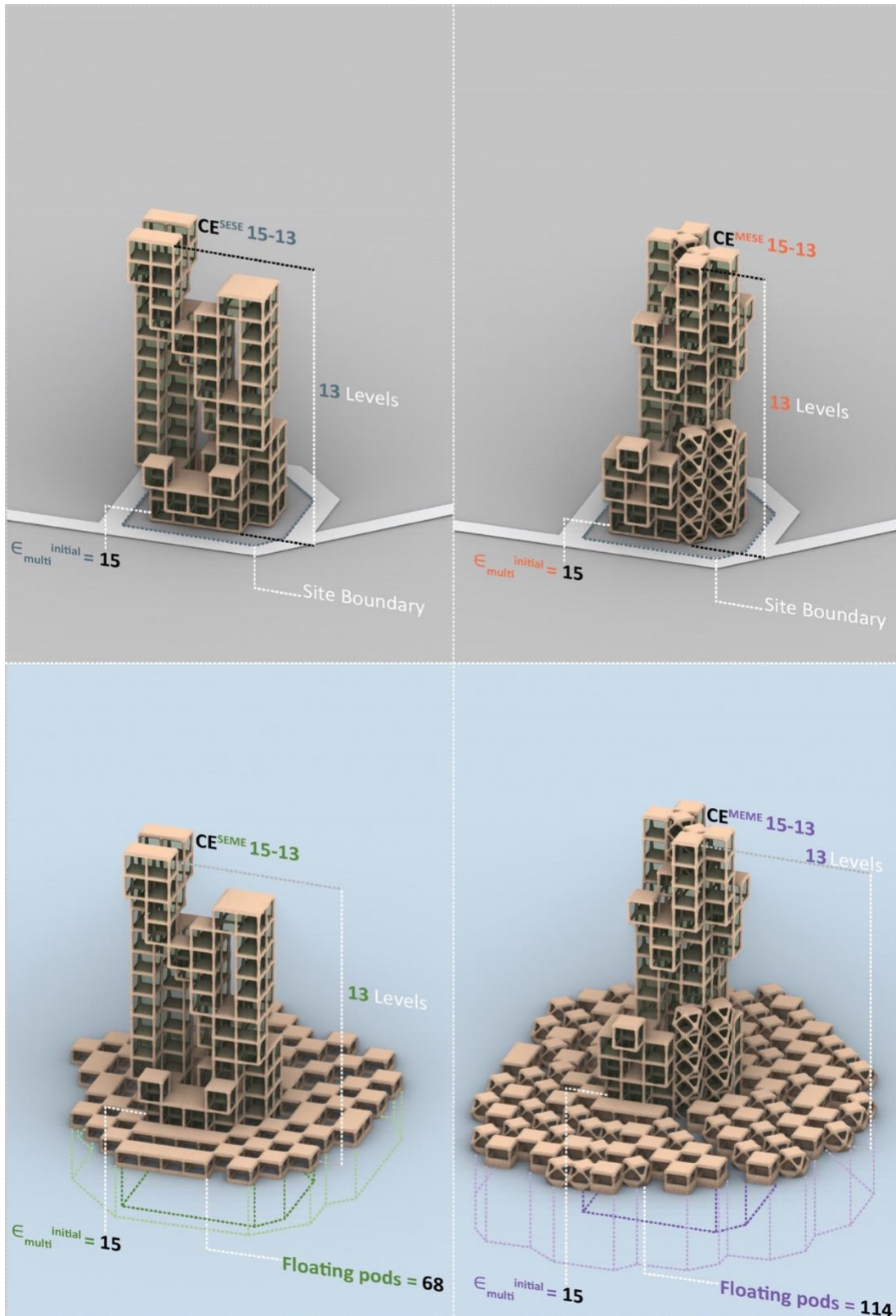


Figure 6.2 – Four different CE with the same $E_{multi}^{initial}$ conditions, envisioned as distinct architectural built forms. Model, algorithm, Illustration and graphics by Author (Jan 2020).

6.3 Effects of distinct Initial States

Contrary to the case explained in 6.1 (effects of distinct ϵ , Ψ , and N considerations), this scenario of having the same ϵ , Ψ , and N considerations (as in operating in the same predetermined considerations for the CE in question) albeit with distinct Initial States has already been sufficiently illustrated and elaborated upon (extensively demonstrated in 5). However, the results have been very superficially observed for their physical characteristics such as height, total number of ϵ achieved by the CE , and/or the (%) immersion in the liquid, and a comparative analysis considering the manner in which the Initial States have an enhanced control over the physical properties of the outcomes of the CE hasn't been really elaborated yet.

Fig. 6.3, demonstrates 4 distinct outcomes of a specific CE that has already been established in this thesis – $CE_{cube-tower}$ (as the CE^{SESE}). As sufficiently done by its predecessors (in 5.1) the figure illustrates how the same ϵ , Ψ , and N considerations ensure that the CE serves the same purpose (as in, in this case the purpose is to construct a tower out of the stacking assemblages of cubes), however the initial states help in generating different iterations that could be manipulated into serving a wide range of functional purposes (as in, the different architectural typologies that the cube tower can serve based on the specific configuration of its assemblages).

As shown in fig. 6.3, and although emphasized sufficiently thus far, it should be noted that the algorithm can in no possible manner control the morphological outcome of the CE purely based on the initial conditions. The initial conditions, which serve as mere random aggregation of the ϵ entities have no bearing to the final outcome that is returned to the user as part of the **Final Rest** condition. This observation could serve as an extension to the halting problem (Berlekamp, Conway and Guy, 2001).¹⁴¹

¹⁴¹ Berlekamp, E. R., Conway, J. H. and Guy, R. K. (2001). *Winning Ways for Your Mathematical Plays*. 2nd ed. Wellesley, Massachusetts: A K Peters, Ltd., p. 276.

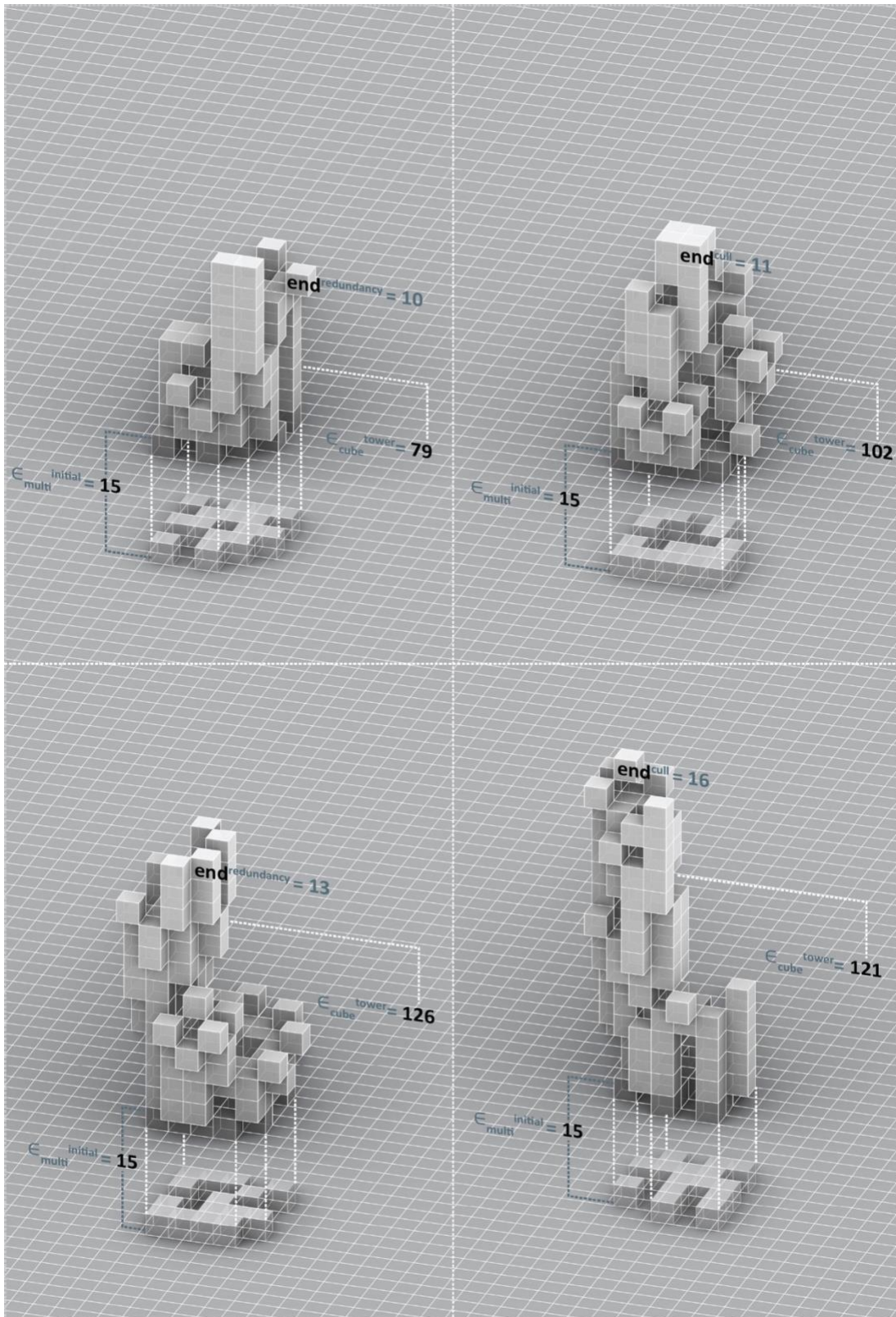


Figure 6.3 – Four $CE_{\text{cube-tower}}$ with the same ϵ , ψ , and N considerations, but different $\epsilon_{\text{multi}}^{\text{initial}}$ conditions. Model, algorithm, illustration and graphics by Author (Jan 2020).

6.4 Examples of possible distinct Initial States for CE

However, this exact property of unpredictability of the actual physical properties of the eventual outcome of a **CE** as compared to its Initial State, can be run through an artificial neural network (ANN), where each permutation and combination within the domain of the possible initial states could be run through an ANN and the **CE** could be equipped to possibly predict which random Initial State could produce the desired outcome. Although performing ANN over the **CE** has not been the objective of this research hypothesis, a possible research trajectory for the further development of **CE** has been suggested in the next chapter (7|).

Although absolutely unpredictable, the Initial State is still highly instrumental in shaping the **CE** (as has been repeatedly discussed in this thesis). Apart from the specific building bye-laws for every context, an AEC project must also conform to a wide range of topographical and geological conditions specific to a given site. Fig. 6.4 illustrates the same four Initial State consideration demonstrated with their outcomes in fig. 6.3. Albeit, in fig. 6.4, the Initial States are envisioned as four distinct topographical formations (in terms of the possible availability of bedrock) for the same given site. Depending on these varied soil conditions, fig. 6.4 illustrates how the **CE_{cube-tower}** (in fact, any **CE** for that matter) can provide a wide range of possible outcomes. As soil analysis is one of the essential prerequisites in any AEC project, this data is quite readily available in the industry. Thus, it can be quite effectively procured and implemented in outputting more rich and perfect results from a **CE**.

Moreover, (following the observations made previously in 4.2.4) the final results of the **CE_{cube-tower}** and the $\epsilon_{\text{cube}}^{\text{initial}}$ conditions were run through the Evolutionary Solver of the Galapagos component in Grasshopper. This addition to the **CE_{cube-tower}** (theorized and taxonomized in 4.2) serves as an essential addition in ensuring that the final outcomes of the **CE** could be ranked in terms of either their height or total number of pods.

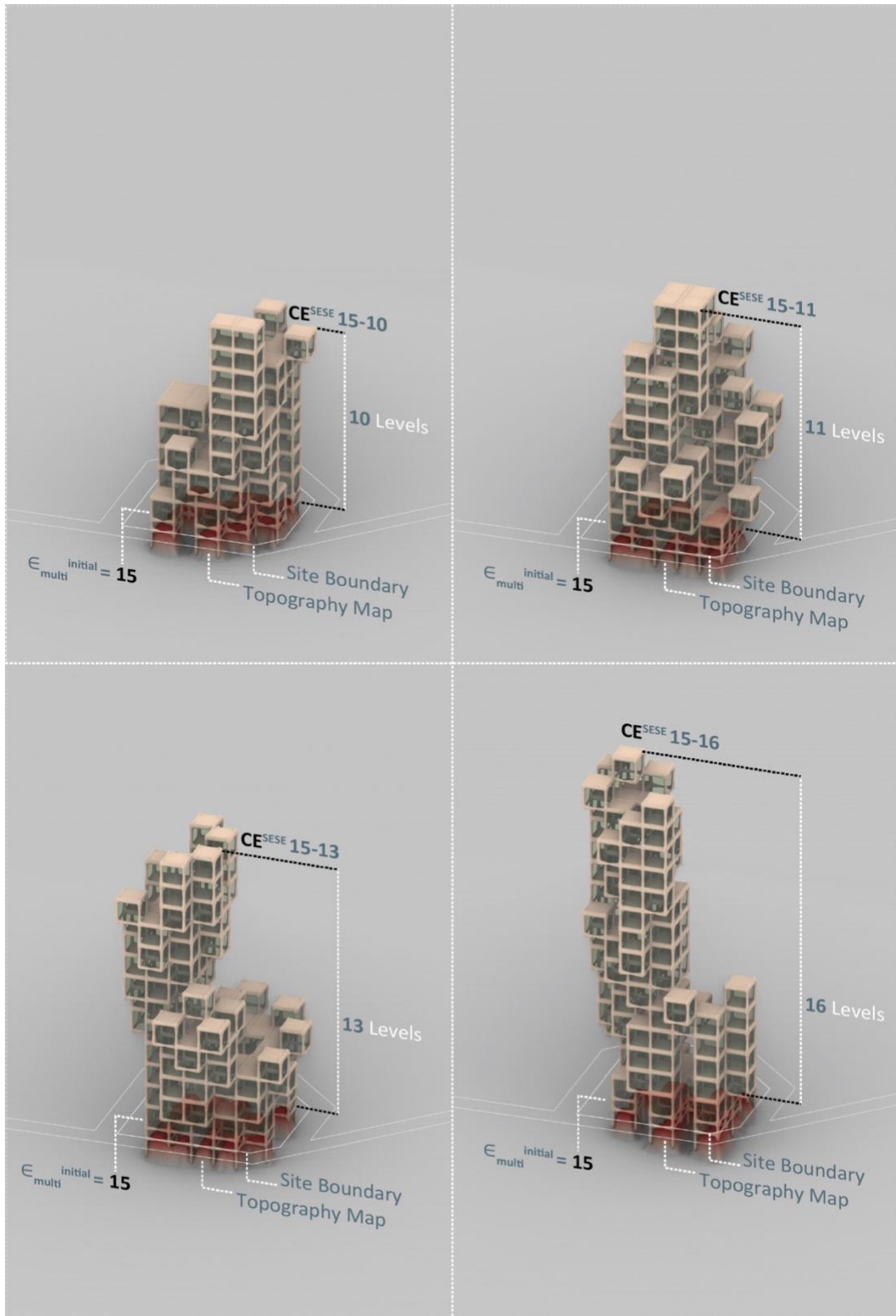


Figure 6.4 – Four $CE_{cube-tower}$ with different $\epsilon_{multi}^{initial}$, envisioned as distinct architectural built forms. Model, algorithm, Illustration and graphics by Author (Jan 2020).

6.5 The inferences of the analysis

This chapter provides the first series of visuals for a comprehensive end-use for the incorporation and implementation of **CE** in the AEC industry. Although these images provide visuals which are quite simulated and computational, these images paint a picture of how a **CE** can be used to express architecture that serves wide range of architectural intent. Moreover, the next chapter (7|) helps elaborate how exactly a **CE** can generate more concrete and realistic built forms in a more visual manner.

This chapter also comes the closest thus far to the practical application of a **CE**. That is, the analysis doesn't just use the computational semantics introduced in the Literary objectives of the CE, but also mentions practical terminology pertaining to the AEC industry. Thus, it would now be prudent to state the following –

- The lexical semantics considered for the theoretical establishment of the concept of **CE** can now be correlated to their counterparts in the AEC industry.
 - **€** considerations – Can be considered as elements similar to several physical elements used in the AEC industry. These could be building materials such as bricks, panels, and tiles. Also building components such as columns, beams, and slabs. Or enclosures as exemplified in 6.2 and 6.4, such as pods of different sizes, or enclosures for various services and equipment. Or, these could also be a wide range of combinations of the above-mentioned examples.
 - **Ψ** considerations – Can be considered organizing principles similar to several physical constraints that dictate built forms in the AEC industry. These could be constraints that could be structural, functional, topographical, geological, economic, related to energy conservation rules, related to services or equipment, or related to local bye-laws.

Pertaining to these correlations, the observations made in the previous pages (as in 6.1, 6.2, 6.3, and 6.4) based on the similarities and differences across the four taxa under the conditions of variability, can be summarized as followed –

- Variability of the ϵ , Ψ , and N –
 - Different **CE** can be implemented over the same site conditions to generate different outcomes depending on the distinct, predetermined considerations of the **CE**.
 - While keeping the site conditions same, implementing different **CE** (in terms of applying distinct, predetermined ϵ , Ψ , and N considerations) can be crucial in evaluating which **CE** develops the most optimum result, thereby concluding which variable ϵ , Ψ , and N parameters can be suited best for a site.
 - In short, a wide range of ϵ considerations mentioned in the correlation could be optimized for those mentioned for the Ψ considerations in the same site conditions.

- Variability of the **Initial States** –
 - The same **CE** (that is the same ϵ , Ψ , and N considerations) could be interpolated for different initial states thereby generating a series of outcomes, which could be ranked in terms of various different parameters such as their height, number of outputted ϵ entities, volume, or other similar parameters concerning the built form.
 - The initial states could be representative of a wide range of site conditions such as topographical data, infrastructural data, and data for the location of services and equipment.

Conclusions

7 | On the prospective projections for the Computational Ecosystems

After carefully and meticulously analyzing all the predetermined taxa through observations made while performing the procedural sequences independently, the thesis shall now focus on how the concept of **CE** can be implemented in various fields within the AEC industry. The research trajectories mentioned in this chapter do not necessarily refer directly to the operations of the AEC industry, but all the mentioned trajectories affect the AEC industry in one way or another.

The chapter includes empirical results of research methodologies that have already been established, performed, and documented by the Author while pursuing the research. Thus, although not performing exactly as proof-of-concept, these examples serve as a foundation for potential research that can be pursued in these directions. As some of these directions exploit the ubiquity of Industry 4.0, a lot of technology that is mentioned in this chapter already exists and does not require any additional research (in that particular field) to perform the mentioned potential advancement of **CE**. However, some of these examples and research trajectories, have not been entirely prototyped by the Author (as the said trajectories are not completely in the scope of this research), and thus will be based on speculation. Nonetheless, these research trajectories serve as potential sanctuaries for **CE** to flourish to an extent where it can be incorporated in the AEC industry as a novel design automation technique that would be a robust link between the **built form** and the **algorithm**.

Apart from exemplifying how **CE** can be implemented in these industries, the empirical results mentioned in this chapter also serve as an example of how modelling, analysis and fabrication can be performed simultaneously while generating form, space and enclosure for a predetermined architectural intent. Moreover, as these results were performed in unison with the respective industries, it is quite evident how flexibly and effectively **CE** can be implemented in these industries for a wide range of purposes.

7.1 Probable research trajectories in Algorithm-aided-design

A large portion of this research that has already been established in the previous chapters (especially 3.3) dwells on the software infrastructure that has been used by the AEC community for the past few decades - CAD. Moreover, the research takes advantage of the software infrastructure that has gained prominence with the advent of Industry 4.0. As already elucidated in the previous chapters (as in 1.1.4 and 1.2.4) a new paradigm of software infrastructure that revolves around the concept of Algorithm-aided-design (AAD) has been implemented in the incorporation of the research methodology (as seen in 4). The research very much stems from the assumption (refer 1.1.4) for the field of computational design that –

As computational design becomes more autonomous, the role of design as the mediator of a construction project becomes more redundant.

This assumption has influenced the methodology of this research into implementing a significant amount of autonomy in computational design by employing **CA** as a computational framework to generate, taxonomize and prototype design automation algorithms in the form of **CE**. The previous chapters (especially 4 |, 5 |, and 6 |) have demonstrated how architectural design can be generated autonomously by inputting predetermined data and constraints pertaining to structure, functional arrangement, and topographical data. Similarly, additional contextual parameters such as climatic data, schedule of services and equipment can also be appended to this design automation algorithm. With the implementation of AAD as an advanced design tool, “that equips designers to design a process rather than just a product” (Tedeschi, 2014)¹⁴². The example illustrated in the upcoming pages demonstrates how a typical Architectural project was developed using a **CE**.

¹⁴² Tedeschi, A. (2014). *AAD_Algorithms-Aided Design - Parametric Strategies using Grasshopper®*. Brienza, Italy: Le Penseur Publisher, p. 495.

The Architectural project was undertaken as a simple design exercise to evaluate and understand how a **CE** could be implemented in a traditional design workflow. The project was conducted within the confines of a conceptual solution provided to the Institute for Biodigital Architecture and Genetics (iBAG) at UIC, Barcelona on the massive immigration that happens in Barcelona, Spain. The solution was proposed in the form of a residential tower, which would be on the sea-front of the city. This proposed tower had the following predetermined constraints –

- It would be situated on a Site that would be a circular floating platform of 200m in diameter, situated on the coast of Barcelona.
- It would be 150m in height with a 1:8 (diameter: height) proportion.
- It would have a structural core possessing two fireproof staircases, and four elevators while accommodating the other services.

The above constraints were converted in the following **CE** parameters –

- **CE** – Owing to the multiplicity in its constraints and parameters, the project would undoubtedly be constructed by employing a **CE^{MEME}**, however the initial setup could be done for a **CE^{SESE}**. The entire workflow would be **CE_{biodigital-tower}**.
- **€** – Considering that the tower needs residential pods, the initial **€** entities could be cubes, thus **€_{cubes}**.
- **Ψ** – Although this **CE** would eventually have a multiplicity in the **Ψ** entities, in the initial considerations, this would just be the standard **Ψ_{tower}**.
- **N** – As the above parameters essentially set up a **CE_{cube-tower}** (as demonstrated in 4.2 and 5.1), the CE would be performed in the previously determined **hypothetical revised infinite 3D Square grid**.

Fig. 7.1 thus demonstrates the entire workflow of the $CE_{\text{bi}digital\text{-tower}}$ as shown below.

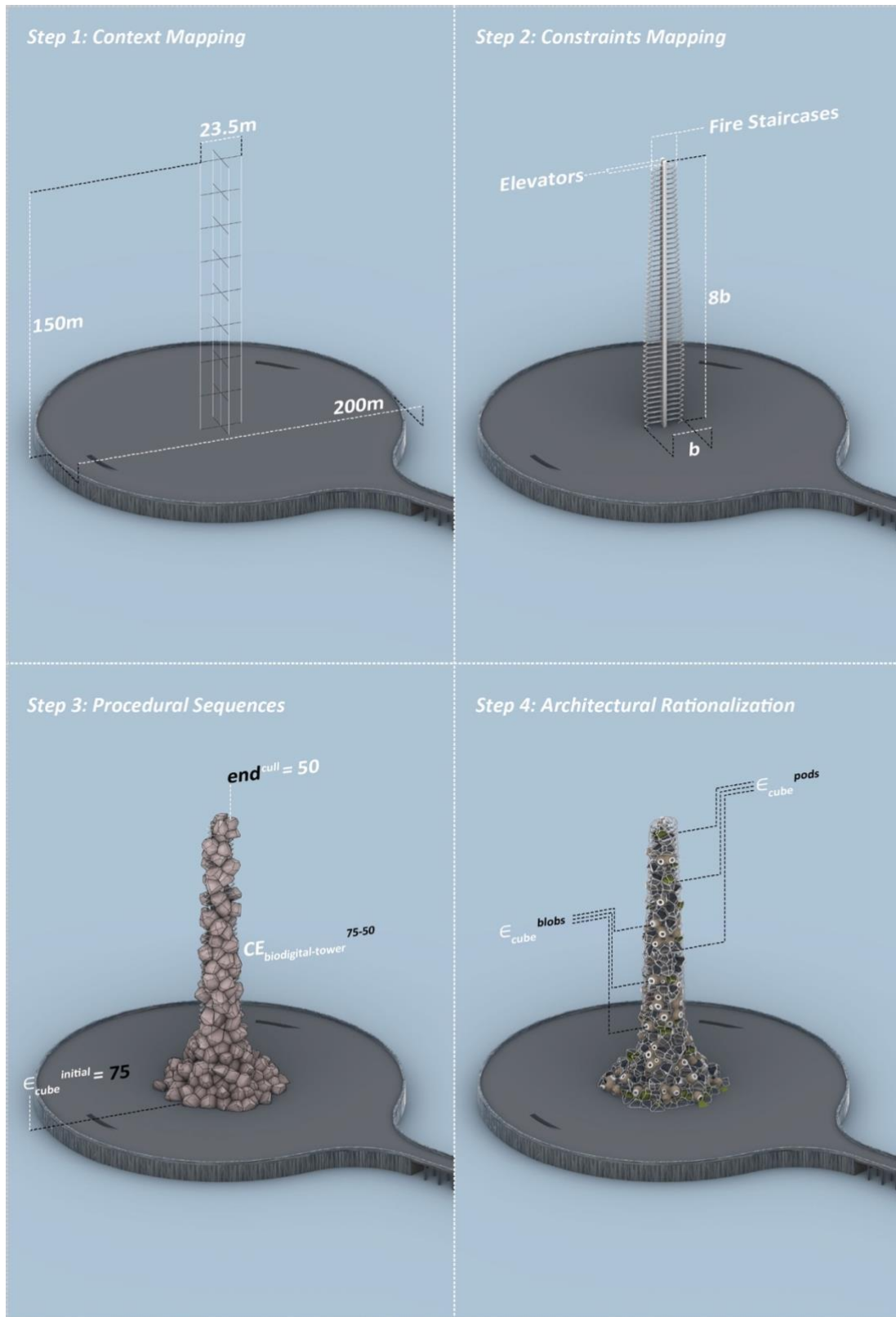


Figure 7.1 – Workflow of $CE_{\text{bi}digital\text{-tower}}$. Model, algorithm, and graphics by Author (May 2020).

The steps involved in the workflow of the $CE_{\text{biodigital-tower}}$ were –

- Step 1 – **Context Mapping** – Here, all the entities pertaining to the context have been mapped. This step serves as a prelude to setting up the \mathbf{N} .
- Step 2 – **Constraints Mapping** – Here, all the constraints related to the design brief have been mapped. This step further establishes the \mathbf{N} and Ψ_{tower} .
- Step 3 – $CE_{\text{biodigital-tower}}$ **Procedural Sequences** – Here, the CE undergoes all the procedural sequences to generate and rank the outcomes as assemblages that could potentially serve as $CE_{\text{biodigital-tower}}$.
- Step 4 – $CE_{\text{biodigital-tower}}$ **Architectural Rationalization** – Here, the selected iteration undergoes assimilation of architectural components ensuring that $CE_{\text{biodigital-tower}}$ can be visualized as a habitable structure.

These steps aid in generating an Architectural Outcome that can be further worked upon to generate a standard AEC construction document (comprising of plans, sections, elevations, and details). Moreover, if the construction strategy and building material are predetermined, they can also be considered as Ψ entities, and repeating steps 2, 3, and 4 would generate a richer outcome. However, the above four steps should not be considered canonical to the CE .

The example of the construction of a $CE_{\text{biodigital-tower}}$ shown above serves as a further proof-of-concept for the implementation of CE in AAD. As the emphasis of AAD is on the process than the product it serves as an ideal, contemporary design system (already in use in the AEC industry), which can serve as a testing ground for the incorporation of CE . Here, the designer still has sufficient control over the **built form** and the **algorithm**. The design (well, actually computational design) merely gets automated while providing outcomes that would be both tedious and overly complex if pursued with traditional design systems (those mentioned in 1.1).

7.2 Probable research trajectories in Additive Manufacturing

Similar to the dependency of this research on the operational infrastructure provided by Computational Design in the form of AAD, the research relies on Digital Fabrication in the form of Additive Manufacturing (AM) for its production infrastructure. Moreover, the research anticipates the rise of Industry 4.0 which could transform architecture back from an allographic profession into an autographic craft, where the architect would be a digital craftsman (as in 1.1.3). The research also emanates from the assumption (as stated in 1.1.4) for the field of **digital fabrication** that –

As digital fabrication becomes more data-driven, the role of design as a medium of generating the construction document becomes more redundant.

Although **digital fabrication** has been sufficiently mentioned in the introduction of the research, it hasn't been exploited in prototyping all the taxa of **CE** mentioned in this thesis, albeit the FDM printed prototyping of the **CE_{cube-cubocta-floating-tower}** (as demonstrated in 4.5 and 5.4). The unavailability of advanced additive manufacturing techniques at the iBAG, UIC, Barcelona has been one of the key reasons that has to a certain extent restricted this research to perform digital fabrication with FDM printing alone. However, the research-stay undergone by the author within the AEC industry for a sufficient duration of three months, has been instrumental in developing a systematic attempt into incorporating **CE** within the discipline of AM.

AM has already been implemented by the AEC industry in various forms such as Robotic Building (Thermoplastics), Robotic Building (Pneumatic Systems), Robotic Stacking, Wire Arc Additive Manufacturing (or Metal Printing) and FDM (Delta printers). Many startups have cropped up globally which are reconfiguring decommissioned Robotic Arms into establishing successful AM Labs as thriving business ventures. The following example is part of the research stay undergone by the Author at one such startup – **Studio RAP** based in Rotterdam, The Netherlands.

Studio RAP is an architecture firm that makes a wide range of bespoke designs across different scales by implementing computational design, often many of which are produced in-house through digital fabrication. One such is the design and production of robotically fabricated, ceramic, monolithic coffee tables. As part of the research-stay, the Author collaborated with the design workflow already implemented at Studio RAP and attempted to incorporate **CE** in the design and fabrication of the coffee tables. Following were the constraints as part of the design brief –

- **Design constraints** – The coffee tables had to be produced and post-processed as monolithic artefacts. This was partly due to the use of potter’s Clay which must be built uniformly, consistently, and not in parts.
- **Material and finish** – The material chosen was a specific potter’s Clay with 20% grog (residual clay that has been fired up and then pulverized to be mixed back into clay, to increase its strength and setting consistency). Further, this material was to be post-processed in the traditional Delft Blue style of glazing.
- **Production constraints** – To be fabricated using a KUKA p6 robotic arm, using a canister-based pneumatic deposition system developed by Studio RAP.

The above constraints were converted in the following **CE** parameters –

- **CE** – The project would need a CE^{SEME} , say a $CE_{clay-table}$.
- ϵ – As the tables are to be built using a robotic arm, ϵ would be the planes defining the eventual tool path of the robotic arm, thus ϵ_{plane} .
- Ψ – The rules for the top would be the Ψ_{top} , and those for the trunk as Ψ_{trunk} .
- **N** – The table would be printed upside down, and thus the **CE** can be performed in the previously determined **hypothetical revised infinite 3D Square grid**.

Fig. 7.2 thus demonstrates the entire workflow of the $CE_{\text{clay-table}}$ as shown below.

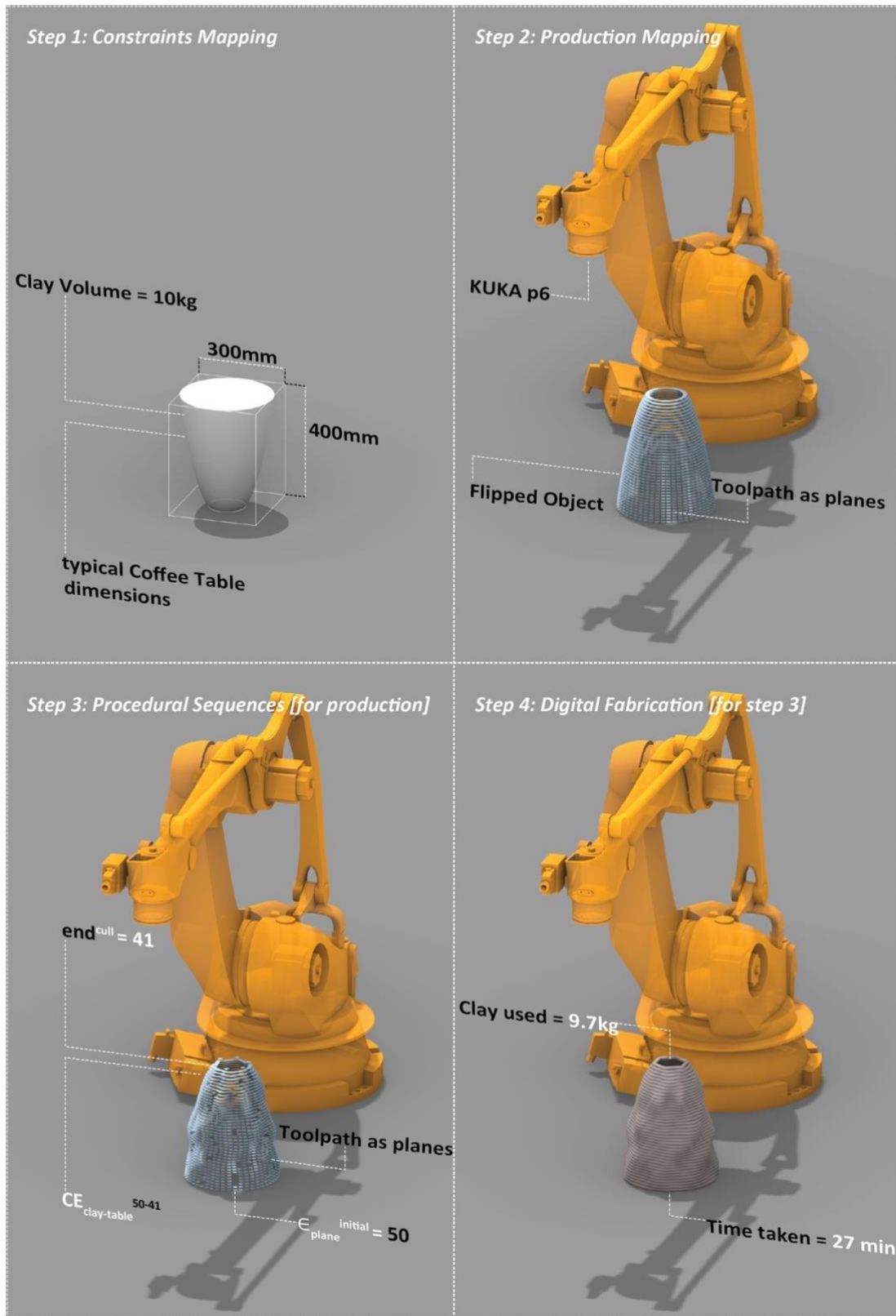


Figure 7.2 – Workflow of $CE_{\text{clay-table}}$. Model, algorithm, and graphics by Author (August 2020).

The steps involved in the workflow of the $CE_{clay-table}$ can be explained as –

- Step 1 – **Constraints Mapping** – Here, all the constraints related to the design brief have been mapped. The table height is optimized to accommodate both the ergonomics of a coffee table and the volume of clay that can be deposited by the canister. This step establishes the N , Ψ_{top} and Ψ_{trunk} .
- Step 2 – **Production Mapping** – Here, all the constraints related to the production have been mapped. The use of ϵ_{plane} as the ϵ entities helps in effectively transitioning into the robot's tool path (as a robotic tool path is determined by planes and not points). This step also helps in mapping how the beautiful process of the Delft Blue glaze can enhance the design.
- Step 3 – $CE_{clay-table}$ **Procedural Sequences** – Here, the CE undergoes all the procedural sequences to generate and rank the outcomes as assemblages that could potentially serve as $CE_{clay-table}$.
- Step 4 – $CE_{clay-table}$ **Digital Fabrication** – Here, the selected iteration undergoes actual production ensuring that $CE_{clay-table}$ can be ceramic coffee tables.

Although the above steps demonstrate how a CE can be implemented as a process to simultaneously design and fabricate a built form with the means of an algorithm, the above steps are not exhaustive, and can vary from CE to CE . However, the example of the $CE_{clay-table}$ serves as substantial empirical evidence of how CE can be crafted into simulating constraints of a certain digital fabrication technique, to design a specific built form and eventually build it using the algorithm. As this project involved a client and end-user, the $CE_{clay-table}$ was prototyped several times before performing the post-processing. Fig. 7.3 further demonstrates the stages in the fabrication of the $CE_{clay-table}$. The images only demonstrate how the $CE_{clay-table}$ was fabricated following the toolpath and ergonomic constraints of the KUKA P6. As the post-processing was not done in-house by Studio RAP, there is no documentation for it.



Figure 7.3 – Fabrication stages for the *CE_{clay-table}*. Illustration, and photographs by Author (August 2020).

As demonstrated in fig. 7.3, the images show how the **CE_{clay-table}** was fabricated using the freedom offered by the robot, as well as the constraints of the material and the extrusion method. The canister-based pneumatic deposition system developed by Studio RAP, involves an aluminum container that can hold 10kg of potter's clay. This severely restricts the size of objects that could be built using the system, as compared to a tube-based continuous material extrusion system that is often used in AM with clay (and similar water-based materials). The drying process of clay also had to be systematically done, as the object shrinks to approximately 85% of its original size (generally the shrinkage is much more, but because the project used clay filled with 20% grog, the shrinkage was minimal). Nonetheless, the object had to be dried for 14 days in an isolated dehumidified container to avoid inconsistent shrinkage and possible warping, before sending it for post-processing (that is firing and glazing).

However, all of the above-mentioned constraints were incorporated in the **CE_{clay-table}**, chiefly by means of the Ψ entities. Thus, this example demonstrates how a CE can accommodate not just the functional requirements of a built form, but also the production constraints offered by the material and fabrication process employed to conceive the built form. In a traditional design-based workflow, here the process would have had to follow the process of analyzing, modelling, and prototyping several times to eventually generate the best possible iteration, with **CE** however, the analysis, modelling and prototyping happens simultaneously and dynamically.

Moreover, the implementation of **CE** in the workflow of an already established design practice performing at the cutting edge of Industry 4.0 (as in Studio RAP), opens up very ambitious prospects for the involvement of **CE** in AM. In this potential future, the design or the designer would no more be archivists of construction documents, as digital fabrication would itself strive a dialogue between the built form and the algorithm. Perhaps, in the distant future (with the 5th Industrial revolution) a robotic arm could be monitored dynamically by a cloud-based **CE**. Perhaps this **CE** could ensure that the robotic arm could build a **CE_{clay-table}** autonomously for a wide range of predetermined parameters and not just monolithic, ceramic coffee tables.

7.3 Probable research trajectories in Pedagogy

As **CE** is a novel approach in design automation, it has to be tested and evaluated in as many industries related to design as possible. This would not only make it robust, but its failures in one industry can also boost its success in others. To ensure that **CE** can be applied in any **built form – algorithm** relationship, the theory explained in this thesis (especially 2| and 3| that emphasize on accomplishing the literary objectives set up in 1.2.1) is very essential. Moreover, any further study in the field of **CE**, could be based on the theoretical foundations laid down in this thesis.

The theory that had actually culminated from the ambitiousness of the futuristic and possibly sensational hypothesis **of addressing the increasing redundancy of design within the built form and the algorithm**, has been quite methodically formulated while considering already established semantics (both lexical and logical) from a wide range of domains such as Biology, Ecology, Computational Sciences, Applied Mathematics, Applied Mechanics, Fabrication, Manufacturing and Economics relevant to the research. Moreover, before acknowledging them as the core theoretical structure of this research, these semantics have also been considered for the contradictions they might offer while being placed aside each other. For example, the halting problem that is associated with the operations and results of any **CA**, has been theoretically and operationally countered by applying the concept of decay to eventually halt a **CE** by means of exhaustion of resources (as elaborated in 3.2). Additionally, while establishing these semantics, the research also derives a consistent methodology, with step-by-step instructions on installing a system within the **built form – algorithm** relationship that can guide future researches into strengthening the theoretical foundations of **CE**.

This very aspect of the research (the manner in which it has a considerable theoretical foundation), opens a new direction in the understanding, implementation and eventual propagation of **CE - Pedagogy**.

In the course of accomplishing the preliminary and operational objectives by means of the procedural sequences (refer 1.2.1) the research has also undergone plenty of evaluation and testing within the confines of academic environment in the form of workshops. Although these workshops were aimed at testing the computational robustness of a wide range of **CE**, often the participants had to be trained and informed about various aspects related to the research. The following were some of the key pre-evaluation tutelage, that had to be done during the workshop –

- **Theoretical Background** – To acclimatize all the different participants to the relevance, purpose, and background of the research, the workshop began with presentations on the semantics and theoretical foundations of the research. Often after the presentations, the participants were encouraged to elaborate on their understanding of the concept of **CE**, and their perception of the application of **CE** in the operations of the AEC industry (before undertaking the evaluations and thereby the Author's directions for **CE**).
- **Methodological Background** – After developing a sufficient knowledge-inventory based on the semantics and theory, the participants were introduced to the methodology adopted by the research in conducting the procedural sequences for **CE**. Here, participants were provided with relevant tool-kits developed for the specific **CE**, and trained into using the tool kit to perform any preliminary tests as relevant to the **CE** in question.
- **Technological Background** – The research relies on the implementation of computational design and digital fabrication, and however ubiquitous these technologies are in the AEC industry, their lack of theoretical background denies their acceptance in design curricula in many educational institutes. Thus, the workshop comprised of providing tutoring for Rhinoceros3D and Grasshopper 3D (with its plugins that were relevant in the functioning of the **CE** in question). As required for the evaluation of a specific **CE**, the workshop also included training on the use and applications of FDM printing.

Theorizing, Taxonomizing and Prototyping operational Ecosystems in Computational environments

Fig. 7.4 shows a collage of photographs taken while several different workshops were conducted by the Author throughout the course of the research to develop, evaluate and demonstrate all the four taxa of **CE** mentioned in the thesis. The image also shows some outcomes developed by the participants during different workshops.



Figure 7.4 – Collage of different workshops conducted by the Author for developing, evaluating and demonstrating the four **CE** taxa. Photographs by Author (from July 2018 to Dec 2020).

As illustrated in fig. 7.4, following are the details of the different workshops –

- **Designing Ways of Designing Workshop** – As elaborated previously (in 4.2.3), the workshop was conducted in June 2018 at the IES College of Architecture in Mumbai, India. Attended by 20 candidates (16 Students and 4 Architects), it was a 3-day workshop. As it was setup for the evaluation of a $CE_{cube-tower}$ (refer 4.2), it involved physical modelling by means of stacking acrylic cubes and gluing them on an acrylic sheet. The results generated during the course of this workshop have helped establish a foundation for the methodology while providing enough evidence that the state conditions derived for the Ψ_{tower} rules (as per 4.2.2) were performing as expected.
- **Computation as a Design tool Workshop** – As elaborated previously (in 4.3.3), the workshop was conducted in July 2019 at RIT in Kottayam, India. Attended by 20 candidates (20 selected Students of the 7th and 9th semesters pursuing BArch), it was also a 3-day workshop. As it was setup for the evaluation of computational prototyping of two different CE – $CE_{cube-cubocta-tower}$ (refer 4.3) and $CE_{cube-floating-tower}$ (refer 4.4), it only involved generating and evaluating computational results for the two CE . The results of this workshop have helped in providing empirical results that the multiplicity of the ϵ and Ψ entities can be managed distinctly with the application of the right CE .
- **Digital Fabrication Workshop** – As elaborated previously (in 4.5.3), the workshop was conducted in December 2019 at the IES College of Architecture in Mumbai, India. Attended by 10 candidates (6 Students, 3 Architects and 1 Design Engineer), it was also a 3-day workshop. As it was setup for the evaluation of a $CE_{cube-cubocta-floating-tower}$ (refer 4.5), it involved evaluating the computational results and digitally fabricating selected CE by means of FDM printing. The results generated during the course of this workshop have helped in providing empirical results for the amalgamation of computational design and digital fabrication in the workflow of CE .

- **Procedural Systems Workshop** – This workshop was conducted from January 2021 to March 2021 for the Biodigital Architecture Master program held at UIC Barcelona. Attended by 8 candidates (8 Students enrolled in the Biodigital Architecture Master Program 2021), originally it was a 6-week long workshop, but the contribution towards this research was not during the entire time and was scattered throughout the course of these 6 weeks. The workshop was not conducted to provide any evaluation of specific CE, but was employed to evaluate some of the independent Ψ entities which have been eventually implemented in the functioning of the *CE_{biodigital-tower}* (refer 7.1).
- **Permutable Morphologies Workshop** – This workshop was conducted from January 2020 to February 2020 also for the Biodigital Architecture Master program held at UIC Barcelona. Attended by 11 candidates (11 Students enrolled in the Biodigital Architecture Master Program 2020), originally it was a 3-week long workshop, but the contribution towards this research was not during the entire time and was scattered throughout the course of these 3 weeks. The workshop was not conducted to provide any evaluation of specific CE, but was employed to evaluate some of the independent ϵ entities which have been implemented in the functioning of the *CE_{biodigital-tower}* (refer 7.1).

As mentioned in the detailed explanations of all the conducted workshops above, the fundamental objective of these workshops was to generate and evaluate the said *CE*. However, the workshops also provided with an additional purpose of rationalizing the theoretical constructs *CE* was established upon. Instructing and developing *CE* in an institutional environment has also provided valuable insights into how the community perceives *CE* as a robust link between built form and algorithm from different vantage points. This perception has and could develop many more varied prospective projections for the application and implementation of *CE*. Moreover, expanding the concept of *CE* to pure and applied sciences could also help make it more mainstream.

While it has several benefits for the development of **CE**, design education can also benefit from the theory of **CE**. The semantics and case studies formulated in each of the four taxa, has a strong bearing in pursuing **CE** as a subset of Bio-inspired computational algorithms, and can be researched upon while generating several different **CE** within the confines of **CE^{SESE}**, **CE^{MESE}**, **CE^{SEME}**, and **CE^{MEME}**. Perhaps, further research could also develop some more taxa besides the above four. As explained in the examples of the **CE_{biodigital-tower}** (refer 7.1) and **CE_{clay-table}** (refer 7.2), the concept of **CE** could have promising, and yet optimized outcomes as Industry 4.0 computational design gets more autonomous and digital fabrication becomes more data driven. And it is about time, the design curricula updated accordingly.

Biodigital Architecture as a domain of architectural design has been “*theorizing and prototyping novel methodologies and technologies at the intersection of the biological and the digital for the past couple of decades*” (Estevez, 2015)¹⁴³. As the concept of **CE** is based on this very idea, Biodigital Architecture is and would be a perfect niche that the **CE** could occupy for its application in pedagogy. Meaning, **CE** could very well perform as a discipline existing within **Biodigital Architecture**, and be further researched within the operational activities of a research group.

However, unlike the Genetic Architectures Research group and iBAG, UIC, Barcelona, not all design institutions share openness for futuristic and interdisciplinary didactic domains. Throughout the course of interactions within the educational community for conducting several more workshops in India and Europe, the Author has faced a fair share of discouragement to incorporate **CE** (or even computational design and digital fabrication) in the design curriculum. Design schools emphasizing on the traditional approach in architecture, have often unequivocally rejected the inclusion of any form of computation in the process of architectural design. Thus, any form design automation (such as **CE**) would not be effortlessly accepted in this industry.

¹⁴³ Estévez, A. T. (2015). *Arquitectura Biodigital Y Genética*. Barcelona, Spain: ESARQ (Escola Tècnica Superior d'Arquitectura, Universitat Internacional de Catalunya), p. 296.

7.4 Probable research trajectories in Software Development

Apart from pursuing literary objectives which eventually resulted in the theoretical foundations for **CE**, the research has also followed a thoroughly methodical approach in creating and documenting all the algorithms. The following steps, have been pursued quite consistently for every taxon, as part of the procedural sequences –

- **Pseudo code** – The State conditions established for every taxon have consistently been derived from the Cellular Automata models mentioned in this thesis (as in 2.3). Moreover, all the components of the algorithm (for the respective taxon) have been consistently defined in reference with the lexical and logical semantics (as in 3.1, 3.2, and 3.3).
- **UML Class Diagram** – All the components of each algorithm have then been methodically illustrated in the form of UML Class Diagrams where all possible **€** and **Ψ** entities have been identified while defining their respective attributes, operations, and inheritances (refer 4.2.2, 4.3.2, 4.4.2, and 4.5.2).
- **UML Sequence Diagram** – To understand and further execute the relationships between these classes all the algorithms have then been methodically illustrated in the form of UML Sequence Diagrams where all possible **€**, **Ψ**, and **N** considerations have been identified while defining their respective role, interaction and runtimes (refer 4.2.3, 4.3.3, 4.4.3, and 4.5.3).
- **User Testing** – After writing the algorithms through visual programming (in Grasshopper3D), the algorithms have been extensively tested by inviting users (in the form of students and practitioners in the AEC industry) by conducting workshops. This quite vital step has also assisted in generating several versions for the specific algorithms in all the taxa, while user issues and minor bugs have been fixed in the base code.

By performing the aforementioned steps, the research has also followed a workflow that is adopted by the software development industry. To a certain extent, this validates **CE** to be identified and implemented as a stand-alone software that could serve its purpose of creating a dynamic, reciprocal, symbiotic relationship between the built form and the algorithm. While CE cannot be identified as one single software but a category of software which can perform a series of certain predetermined tasks. However, the research objective behind the **Architecture of Computational Ecosystems** has never been about developing and releasing a software in the market, and thus this has not been pursued in the course of this research. But it can serve as a possible research direction in the development of **CE**. In fact, it is quite inevitable through the development of **CE** that it would eventually evolve into an implementable software that could possibly be used by the AEC industry.

CE could be used as a computational framework for game development. The example shown below in fig. 7.5 is a single-player city building game called Block'hood¹⁴⁴. It involves building vertical neighbourhoods with cubes that serve different purposes such as resources, services, residential, commercial, structure, and many more. While being an entertaining game it also provides insights into architectural design.



Figure 7.5 – Block'hood on Steam. Source:

<https://store.steampowered.com/app/416210/Blockhood/>.

¹⁴⁴ Block'hood (2017). Los Angeles, USA: Plethora Project.

Although Block'hood has not been developed using **CE**, it can be used as a template to determine how **CE** can be implemented in game development of similar simulation games. Moreover, interesting outcomes could possibly be achieved in Block'hood and similar simulation games if some aspects of **CE** were applied in the game engine. However, as **CE** is based on the computational framework of **CA**, and most **CA**, such as Conway's Game of life (refer 2.3.2) are zero-player games (as in, they don't really require sentient participants for it to run or end), it can be stated syllogistically, that games generated using **CE** could be zero-player games. However further research in this industry could possibly be pursued to develop bio-inspired simulation games.

Apart from using **CE** as computational framework for software and game development, **CE** could also be used as a data source for generating Artificial Neural Networks (**ANN**). As introduced in 6.4, a **CE** if appended with an **ANN**, could benefit from its predetermined learning paradigm. Although any of the paradigms can be considered, self-learning based on "Crossbar Adaptive Array" (Bozinovski, 2014)¹⁴⁵ which relies on one input **situation** and one output **behavior** could be a good place to start exploration. In the context of **CE**, the input situation could be the **initial state**, while the output behavior could be **End Rest** Array of the **CE**. Thus, a correlation could be deduced between the situation and the behavior, and with further research a mechanism could possibly be developed to predict the result of an initial state, or even to dictate a certain preferred result to determine the possible initial state.

Contrary to the above direction, several results of **ANN**, however, could also be used as a data source for generating various distinct **CE**. The data source provided by the **ANN** could be any architectural parameter pertaining to its typology or context. Thus, **ANN** could be an interesting possible direction to further research on **CE** and its applications.

¹⁴⁵ Bozinovski, S. (2014) Modelling Mechanisms of Cognition-Emotion Interaction in Artificial Neural Networks, since 1981. In: *BICA 2014. 5th Annual International Conference on Biologically Inspired Cognitive Architectures*. [online]: Procedia Computer Science, pp. 255-263. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050914015567> [Accessed 15 Jun. 2021]

7.5 Concluding Statements

“Humans have been historically horrible at predicting the pace of progress”

(Kurzgesagt – In a Nutshell, 2020, 07:55).¹⁴⁶

And this research, while being based on the prophetic assumptions for both the ubiquity of the algorithm and the redundancy of design delicately relies on speculations from the future of the AEC industry. However, the technology (as in Industry 4.0) that it relies on, and the scientific theory that it is built upon, ensures that (granted the openness of the industry) the ***Architecture of Computational Ecosystems*** can certainly help establish a dynamic, reciprocal, symbiotic relationship between the ***built form*** and the ***algorithm*** by making computational design more autonomous and digital fabrication more data driven. While this relationship is based on the premise of theorizing, generating, taxonomizing, and prototyping ***Hybrid Bio Plausible Bio-inspired Stochastic Optimization Algorithms*** as functioning ***Computational Ecosystems*** which perform as ***autonomous, autopoietic, context aware feedback loops*** between the ***built form*** and the ***algorithm*** (refer 2.4), it supports the hypothesis that, ***Cellular Automata can be employed as a computational framework to generate, taxonomize and prototype design automation algorithms*** with empirical evidence (refer 7.1).

Moreover, the procedural sequences performed to develop the empirical evidence over the four possible taxa for Computational Ecosystems – ***CE^{SESE}*** (refer 4.2), ***CE^{MESE}*** (refer 4.3), ***CE^{SEME}*** (refer 4.4), and ***CE^{MEME}*** (refer 4.5) has helped in generating a robust workflow that can be incorporated in the AEC industry while proving that, ***fabrication data in the form of G-code can be used as a fitness condition to generate, taxonomize and prototype digitally generated built forms*** (refer 7.2).

¹⁴⁶ Kurzgesagt – In a Nutshell (2020) *Can You Upload Your Mind & Live Forever?*. 10 December. Available at: <https://www.youtube.com/watch?v=4b33NTAuF5E> (Accessed: 18 Dec. 2020).

The inception and execution of the research had followed a systemic trajectory that can be summarized as stated below –

- **Establishing semantic syntax** – for the lexical and logical semantics (refer |2)

- **Performing procedural sequences for each of the four taxa** (refer |4)
 - **Case Studies** – bio-based theoretical and operational analogies.

 - **Simulations** – transcription of the analogies into the ϵ , Ψ , and N considerations.

 - **Prototyping** – evaluation and taxonomizing by means of industry participation (through workshops and research collaborations).

This has not only helped establish a standard, canonical methodology for the **Architecture of Computational Ecosystems** (refer 5.5 and 6.5) but also assisted in abrogating the sensationalism involved with predicting the pace of progress. Nonetheless, apart from establishing an unprecedented relationship between the **built form** and the **algorithm**, the **Architecture of Computational Ecosystems** opens a plethora of research opportunities in the domains of AAD, AM, pedagogy, and software development (refer |7). Although not an exhaustive list, it certainly ensures that the pursuit documented in this thesis is a mere beginning.

After all, as Einstein famously quoted (about Thermodynamics) –

“A theory is the more impressive the greater the simplicity of its premises, the more varied the kinds of things that it relates and the more extended the area of its applicability” – Albert Einstein (Schilpp, 1959).¹⁴⁷

¹⁴⁷ Schilpp, P. A. (1959). *Albert Einstein: Philosopher-Scientist*. New York: MJF Books, pp. 32 (Autobiographical Notes).

7.5 Concluding Statements (in Spanish)

“Los seres humanos han sido históricamente horribles al predecir el ritmo del progreso” (Kurzgesagt – In a Nutshell, 2020, 07:55).¹⁴⁸

Y esta investigación, si bien se basa en los supuestos proféticos tanto de la ubicuidad del algoritmo como de la redundancia del diseño, se basa delicadamente en especulaciones del futuro de la industria AEC. Sin embargo, la tecnología (como en la Industria 4.0), y la teoría científica en las que se basas, asegura que (dada la apertura de la industria) la Arquitectura de Ecosistemas Computacionales ciertamente puede ayudar a establecer una dinámica, recíproca, simbiótica relación entre la forma construida y el algoritmo al hacer que el diseño computacional sea más autónomo y la fabricación digital más impulsada por los datos. Si bien esta relación se basa en la premisa de teorizar, generar, taxonomizar y crear prototipos de algoritmos de optimización estocásticos bioinspirados bio-plausibles híbridos como ecosistemas computacionales en funcionamiento que funcionan como bucles de retroalimentación autónomos, autopoyéticos y sensibles al contexto entre la forma construida y el algoritmo (2.4), apoya la hipótesis de que **Cellular Automata puede emplearse como un marco computacional para generar, taxonomizar y prototipar algoritmos de automatización de diseño con evidencia empírica** (7.1). Además, las secuencias de procedimiento realizadas para desarrollar la evidencia empírica sobre los cuatro posibles taxones de ecosistemas computacionales – CE^{SESE} (4.2), CE^{MESE} (4.3), CE^{SEME} (4.4), and CE^{MEME} (4.5) ha ayudado a generar un sólido flujo de trabajo que se puede incorporar en la industria AEC y que, al mismo tiempo, se demuestra que **los datos de fabricación en forma de G-Code se pueden utilizar como condición de aptitud para generar, taxonomizar y crear prototipos de formularios construidos generados digitalmente** (7.2).

¹⁴⁸ Kurzgesagt – In a Nutshell (2020) *Can You Upload Your Mind & Live Forever?*. 10 December. Available at: <https://www.youtube.com/watch?v=4b33NTAuF5E> (Accessed: 18 Dec. 2020).

El inicio y ejecución de la investigación había seguido una trayectoria sistémica que se puede resumir como se indica a continuación –

- **Establecimiento de sintaxis semántica** - para la semántica léxica y lógica (|2)

- **Realización de secuencias de procedimientos de los cuatros taxones** (|4)
 - **Estudios de caso**: analogías teóricas y operativas de base biológica.

 - **Simulaciones**: transcripción de las analogías en las ϵ , Ψ y N .

 - **Prototipado** - evaluación y taxonomización mediante la participación de la industria (a través de talleres y colaboraciones de investigación).

Esto no solo ha ayudado a establecer una metodología canónica estándar para la **Arquitectura de Ecosistemas Computacionales** (5.5 y 6.5), sino que también ayudó a abrogar el sensacionalismo involucrado en la predicción del ritmo del progreso. No obstante, además de establecer una relación sin precedentes entre la **forma construida** y el **algoritmo**, la Arquitectura de Ecosistemas Computacionales abre una plétora de oportunidades de investigación en los dominios de AAD, AM, pedagogía y desarrollo de software (|7). Aunque no es una lista exhaustiva, ciertamente asegura que la búsqueda documentada en esta tesis es un mero comienzo.

Después de todo, como citó Einstein (sobre la termodinámica):

“Una teoría es más impresionante cuanto mayor es la simplicidad de sus premisas, más variadas son las cosas que relaciona y más amplia es el área de su aplicabilidad.” – Albert Einstein (Schilpp, 1959).¹⁴⁹

¹⁴⁹ Schilpp, P. A. (1959). *Albert Einstein: Philosopher-Scientist*. New York: MJF Books, pp. 32 (Autobiographical Notes).

8 | Bibliography

8.1 References

1. AA School of Architecture (2015) John Frazer - An Evolutionary Architecture. 02 May. Available at: <https://www.youtube.com/watch?v=58ZUhdKaRC8> (Accessed: 18 Dec. 2017).
2. Allen, C. and Shakantu, W. (2016). The BIM revolution: a literature review on rethinking the business of construction. In: 11th International Conference on Urban Regeneration and Sustainability, Bilbao: WITconferences pp. 919-930 Available at: <https://www.witpress.com/elibrary/wit-transactions-on-ecology-and-the-environment/204/35716> [Accessed 17 May 2019].
3. Antunes, R. F. (2016). Human Crowd Simulation: What can We Learn from ALife? In: ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems. [online] Cancun: MIT Press Direct, p. 8. Available at: <https://direct.mit.edu/isal/proceedings/alif2016/38/99500> [Accessed 26 Apr. 2021].
4. Antunes, R. F., Leymarie, F. F., Latham, W. (2016). Computational Ecosystems in Evolutionary Art, and Their Potential for the Future of Virtual Worlds. In: Y. Sivan, ed., Handbook on 3D3C Platforms, 1st ed. Cham, Switzerland: Springer International Publishing, pp. 441-473.
5. Aschwanden, G.D.P.A., Wullschleger, T., Müller, H., Schmitt, G. (2009). Agent based evaluation of dynamic city models: A combination of human decision processes and an emission model for transportation based on acceleration

- and instantaneous speed. *Automation in Construction*. [online] 22, pp. 81-89. Available at:
<https://www.sciencedirect.com/science/article/pii/S0926580511001415>
[Accessed 22 May 2020].
6. Bellman, R.E. and Dreyfus, S.E., (1962) *Applied Dynamic Programming*, London: Oxford University, 362 pp.
 7. Berlekamp, E. R., Conway, J. H. and Guy, R. K. (2001). *Winning Ways for Your Mathematical Plays*. 2nd ed. Wellesley, Massachusetts: A K Peters, Ltd., p. 276.
 8. Białyński-Birula, I., Białyńska-Birula, I. (2004). *Modelling Reality - How Computers Mirror Life*. Oxford: Oxford University Press, p.188.
 9. Binitha, S. and Sathya, S. S. (2012). A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2), pp. 137-151.
 10. Bickel, T. (2000). Tournament Selection. In: T. Bäck, D. B. Fogel, and Z. Michalewicz, ed., *Evolutionary Computation 1: Basic Algorithms and Operators*, 1st ed. New York: Taylor & Francis Group, pp. 181-186.
 11. Bowles, S. & Choi, J-K. (2019). The Neolithic Agricultural Revolution and the Origins of Private Property. *Journal of Political Economy*, 127(5), pp. 2186-2228.
 12. Boyle, R. (1661). *The Sceptical Chymist*. London: J. Cadwell
 13. Bozinovski, S. (2014) *Modelling Mechanisms of Cognition-Emotion Interaction in Artificial Neural Networks, since 1981*. In: BICA 2014. 5th Annual International Conference on Biologically Inspired Cognitive

Architectures. [online]: *Procedia Computer Science*, pp. 255-263. Available at: <https://www.sciencedirect.com/science/article/pii/S1877050914015567> [Accessed 15 Jun. 2021]

14. Caplat, P., Anand, M., Bauch, C. (2007). Symmetric competition causes population oscillations in an individual-based model of forest dynamics. *Ecological Modelling* 211. 3(4), pp. 491-500.
15. Carpo M (2011). *The Alphabet and the Algorithm*. The MIT Press, pp. 10-19.
16. Chen H. and Zhu Y. (2008). Optimization based on symbiotic multi-species coevolution. *Applied Mathematics and Computation*, 205(2008), pp. 47-60.
17. Codd, E. F. (1968). *Cellular Automata*. PhD Thesis. Academic Press, New York.
18. David, A. (2020). Special report: The simulations driving the world's response to COVID-19. *Nature*, [online]. Available at: <https://www.nature.com/articles/d41586-020-01003-6> [Accessed 12 Jan. 2021].
19. De Bary, A. (1879) *Die Erscheinung der Symbiose: Vortrag, gehalten auf der Versammlung deutscher Naturforscher und Aerzte zu Cassel (In English - The Phenomenon of Symbiosis)*, Strassburg: Karl J. Trübner, 30 pp.
20. De Castro, L.N. (2007) *Fundamentals of Natural Computing – basic concepts, algorithms, and applications*, Boca Raton: Taylor & Francis group, 638 pp.
21. Dorigo, M., Maniezzo, V. and Colorni A. (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26(1), pp. 1-13.

22. Douglas, A. E. (2010) *The Symbiotic Habit*, Princeton, NJ: Princeton University Press, 232 pp.
23. Estévez, A. T. (2015). *Arquitectura Biodigital Y Genetica*. Barcelona, Spain: ESARQ (Escola Tècnica Superior d'Arquitectura, Universitat Internacional de Catalunya), p. 296.
24. Flynn, M-J. (1972). Some Computer Organizations and their Effectiveness. *IEEE Transactions on Computers*, C-21(9), pp. 948-960.
25. Frazer, J. H. (1995). *Themes VII: An Evolutionary Architecture*. London: Architectural Association, p. 127.
26. Frazer, J. H. (2001) *The Cybernetics of Architecture: A Tribute to the Contribution of Gordon Pask*. *Kybernetes. The International Journal of Systems & Cybernetics*. 30(5/6). pp. 641-651.
27. Freitas Jr., R. A., Merkle, R. C. (2004). *Kinematic Self-Replicating Machines*. [online] www.molecularassembler.com. Available at: <http://www.molecularassembler.com/KSRM/2.1.3.htm> [Accessed 18 Nov. 2017].
28. Gad-el-Hak, M. (2003) *A New Kind of Science - Review*. *A New Kind of Science*, by S. Wolfram. *Applied Mechanics Reviews*, 56 (2), pp. B18-B19.
29. Gardner, M. (1970). *MATHEMATICAL GAMES - The fantastic combinations of John Conway's new solitaire game "life"*. *Scientific American*. [online] 223(4), pp. 120-123. Available at: <https://web.stanford.edu/class/sts145/Library/life.pdf> [Accessed 31 May 2020].

30. Grasse´, P.-P. (1963). Les phe´nome`nes sociaux chez les Animaux. Cahiers de l'Institut de Science e´conomique applique´e. Suppl. 139, V, pp. 7–23.
31. Greenberg, J. M., Hastings, S. P. (1978). Spatial Patterns for Discrete Models of Diffusion in Excitable Media. *SIAM Journal on Applied Mathematics*, 34(3), pp. 515–523.
32. He, S., Wu, Q. H. and Saunders, J. R. (2006). A Novel Group Search Optimizer Inspired by Animal Behavioral Ecology. In: *IEEE International Conference on Evolutionary Computation*. Vancouver: IEEE, pp. 1272-1278.
33. Herr C.M., Kvan T. (2005) Using Cellular Automata to Generate High-Density Building Form. In- Martens B., Brown A. (eds) *Computer Aided Architectural Design Futures 2005*. Dordrecht: Springer, p. 10.
34. Herr, C. M., Ford, R. C. (2015). Adapting Cellular Automata as Architectural Design Tools. In: *Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2015*. Hong Kong: The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA), pp. 169-178.
35. Hilbert, D. & Ackermann, W. (1928). *Principles of Mathematical Logic*. Providence, Rhode Island, USA: AMS Chelsea Publishing, pp. 113-134.
36. Ilachinski, A. (2001). *Cellular Automata A Discrete Universe*. Singapore: World Scientific Publishing Co. Pte. Ltd. Pp. 808.
37. James, P. (2014). *Urban Sustainability in Theory and practice: Circles of Sustainability*. London: Routledge. Pg 53

38. James, P. (2015). *Urban Sustainability in Theory and Practice – Circles of Sustainability*. New York: Routledge, pp. 260.
39. Jencks C (1984). *The language of post-modern architecture*. Rizzoli International Publications.
40. Koehler, D., Saleh, S. A., Li, H., Ye, C., Zhou, Y., Navasaityte, R., (2018). *Mereologies - Combinatorial Design and the Description of Urban Form*. In: *GENERATIVE DESIGN - Volume 2 - eCAADe 36*. Łódź, Poland: eCAADe, Faculty of Civil Engineering, Architecture and Environmental Engineering Lodz University of Technology, cop. 2018. pp. 85-94.
41. Krawczyk, R. J. (2002). *Architectural Interpretation of Cellular Automata*. *Generative Art 2002*. pp. 7.1-7.8.
42. Kurzgesagt – In a Nutshell (2020) *Can You Upload Your Mind & Live Forever?*. 10 December. Available at: <https://www.youtube.com/watch?v=4b33NTAuF5E> (Accessed: 18 Dec. 2020).
43. Langton, C. G. (1984). *Self-Reproduction in Cellular Automata*. In: *Physica 10D*. Amsterdam: Elsevier Science Publishers B.V. (North-Holland Physics Publishing Division), pp. 135-144.
44. Langton, C. G. (1995). *Artificial Life: An Overview*. Cambridge: MIT Press, p.341.
45. Loizeau, Nicolas (2016). *Building a computer in Conway's game of life*. [online] www.nicolasloizeau.com. Available at: <https://www.nicolasloizeau.com/gol-computer> [Accessed 05 May 2020].

46. Magliocca, N. R. (2020). Agent-Based Modelling for Integrating Human Behavior into the Food–Energy–Water Nexus. *Land* 2020[online] Volume 9(519), p. 25. Available at: <https://www.mdpi.com/2073-445X/9/12/519/> [Accessed 24 Apr. 2021].
47. Martinez, L. M., Viegas, J. M. (2017). Assessing the impacts of deploying a shared self-driving urban mobility system: An agent-based model applied to the city of Lisbon, Portugal. *International Journal of Transportation Science and Technology*. [online] 6(2017), pp. 13-27. Available at: <https://reader.elsevier.com/reader/sd/pii/S2046043016300442?token=0D63F683EEF2FFDA49A010FD6D7BD77A18B8CAC02BDE12BCA11F14B0F0702DA369F02E3B46DAB8B7E279C516B24D40CC&originRegion=eu-west-1&originCreation=20210703000320> [Accessed 22 May 2020].
48. Maturana, H. R. (2002). Autopoiesis, Structural Coupling and Cognition: A history of these and other notions in the biology of cognition. *Cybernetics & Human Knowing*, 9(3-4), pp. 5-34.
49. Maturana, H. R. and Varela, F. J. (1980). *Autopoiesis and Cognition: The Realization of the Living*. Berlin: Springer Science & Business Media, p.146.
50. McCormack, J. (2001). Eden: an evolutionary sonic ecosystem. In: *Advances in Artificial Life, 6th European Conference*. Berlin: Springer - Verlag, p. 10.
51. Minsky, M. (1967). *Computation: Finite and Infinite Machines*. New Jersey: Prentice-Hall Inc.p. 334.
52. Mlot, N. J., Tovey, C. A., Hu, D. L. (2011). Fire ants self-assemble into waterproof rafts to survive floods. In: *Proceedings of the National Academy of Sciences of the United States of America*. [online] Washington DC: PNAS, p 6. . Available at: <https://www.pnas.org/content/108/19/7669.full>

53. Moreno, D., Grinda, E. G. (2004). Soft Metropolitanism [Apartments in Micro-Skyscrapers]. In: F. Marquez Cecilia, and R. Levene, ed., EL CROQUIS 118: CERO 9, ABALOS & HERREROS, NO.MAD, 1st ed. Madrid: El Croquis, pp. 140-147.
54. Odum, E. P. (1971). Fundamentals of Ecology. Philadelphia: Saunders.
55. OMG – Object Management Group (2017). OMG® Unified Modelling Language® (OMG UML®) Version 2.5.1. Milford, Massachusetts: OMG Group, pp. 754
56. O' Rourke K.H. and Williamson, J. (2002). When did globalization begin? European Review of Economic History, 6(1), pp. 23-50.
57. Oxman, N. (2010). Material-based Design Computation. PhD Thesis. Massachusetts institute of Technology.
58. Paracer, S. and Ahmadjian, V. (2010) Symbiosis: An Introduction to Biological Associations, Princeton: Oxford University Press, 304 pp.
59. Parpinelli, R. S. (2013). An Ecosystemic View for Developing Biologically Plausible Optimization Systems. PhD Thesis. Federal University of Technology Paraná.
60. Parpinelli R. S. and Lopes, H. S. (2014). A computational ecosystem for optimization: review and perspectives for future research. Memetic Computing, 7(1), pp. 29-41.
61. Parpinelli, R. S. and Lopes, H. S. (2011). An Eco-inspired Evolutionary Algorithm Applied to Numerical Optimization. In: Third World Congress on, Nature and Biologically Inspired Computing. [online] Salamanca: IEEE, pp.

- 466-471. Available at: <https://ieeexplore.ieee.org/document/6089631>
[Accessed 12 Oct. 2020].
62. Parpinelli, R. S. and Lopes, H. S. (2012). Biological plausibility in optimisation: an ecosystemic view. *International Journal of Bio-Inspired Computation*, 4(6), pp. 345-358.
63. Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*. 22(3), pp. 52-67
64. Peters, I. (2009). *Folksonomies. Indexing and Retrieval in Web 2.0*. Berlin: De Gruyter Saur, pp. 445.
65. Phillips, J. (1931). The Biotic Community. *Journal of Ecology*, [online] 19(1). p 1-24. Available at: www.jstor.org/stable/2255934 [Accessed 12 May 2021].
66. Railsback, S. F., & Grimm, V. (2011). *Agent-based and individual-based modelling: A practical introduction*. Princeton: Princeton University Press., p.329.
67. Reid, C. R., Lutz, M. J., Powell, S., Kao, A. B., Couzin, I. D., and Garnier, S. (2015). Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. In: *Proceedings of the National Academy of Sciences of the United States of America*. [online] Washington DC: PNAS, p 6. . Available at: <https://www.pnas.org/content/pnas/early/2015/11/18/1512241112.full.pdf>
68. Ricklefs, R. E. (2008). *The Economy of Nature*. 6th ed. New York: W. H. Freeman and Company, 620 pp.
69. Schilpp, P. A. (1959). *Albert Einstein: Philosopher-Scientist*. New York: MJF Books, pp. 32 (Autobiographical Notes).

70. Schrandt, R. G., Ulam, S. M. (1967). On Recursively Defined Geometrical Objects and Patterns of Growth. [online] Los Alamos, New Mexico: Los Alamos Scientific Laboratory of the University of California, p. 19. Available at:
https://digital.library.unt.edu/ark:/67531/metadc1027179/m2/1/high_res_d/4573212.pdf [Accessed 05 May 2018].
71. Simon, D. (2009). Bio-geography based optimization. *IEEE Transactions on Evolutionary Computation*. 12(6), pp. 702-713.
72. Sipser, M. (2006). *Introduction to the Theory of Computation*. 2nd ed. Boston: Thomson Course Technology, 431 pp.
73. Situngkir, H. (2004). *Epidemiology Through Cellular Automata: Case of Study Avian Influenza in Indonesia*. [online]. Available at:
<https://arxiv.org/abs/nlin/0403035> [Accessed 24 Apr. 2021].
74. Situngkir, H. (2010). *Exploring Ancient Architectural Designs with Cellular Automata*. [online]. Available at:
https://www.researchgate.net/publication/2146550_Epidemiology_Through_Cellular_Automata_Case_of_Study_Avian_Influenza_in_Indonesia [Accessed 31 Aug. 2019].
75. Soare R. I. (1996). Computability and Recursion. *The Bulletin of Symbolic Logic*, [online] Volume 2(3), pp. 284-321. Available at:
<https://www.jstor.org/stable/420992> [Accessed 8 Apr. 2020].
76. Sommerer, C., Mignonneau, L. (1994). A-Volve: A real-time interactive environment. In: *ACM Siggraph Visual Proceedings*. pp. 172–173.
77. Tansley A.G. (1935). The use and abuse of vegetational concepts and terms. *Ecology*, 16(3), pp. 284-307.

78. Tedeschi, A. (2014). AAD_Algorithms-Aided Design - Parametric Strategies using Grasshopper®. Brienza, Italy: Le Penseur Publisher, p. 495.
79. Turing, A. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, 2(42), pp. 230-265.
80. Von Neumann, J. and Burks, A. W. (1966). Theory of Self-Reproducing Automata. Illinois: University of Illinois Press, p.387.
81. Warang, A. (2017). Towards the Architecture of Computation. MS Architecture Thesis. Universitat Internacional de Catalunya (UIC) Barcelona.
82. Warming, E. (1895). Plantesamfund - Grundtræk af den økologiske Plantegeografi. Copenhagen: P.G. Philipsens Forlag, 335 pp.
83. Wenegreen, J. J. (2004). Endosymbiosis: Lessons in Conflict Resolution. PLoS Biology, 2(3), pp. 345-358.
84. Wirth, E., Szabó, G., Czinkóczy, A. (2016). Measure of Landscape Heterogeneity by Agent-Based Methodology. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. III(8), pp. 145-151.
85. Wolfram, S. (1984). Computation Theory of Cellular Automata. Communications in Mathematical Physics, 96 (1984), Pp.15-57.
86. Wolfram, S. (2002). A New Kind of Science. Champaign, IL: Wolfram Media.
87. Wolfram, S. and Packard, N. H. (1985). Two-Dimensional Cellular Automata. Journal of Statistical Physics, 38, pp. 901-946

88. Wood, D., Grönquist, P., Bechert, S., Aldinger, L., Riggenbach, D., Lehmann, K., Rüggeberg, M., Burgert, I., Knippers, J., and Menges, A. (2020). From Machine Control to Material Programming Self-Shaping Wood Manufacturing of a High Performance Curved CLT Structure – Urbach Tower. In: Fabricate 2020 Making Resilient Architecture, London: UCL press pp. 50-57 Available at: <https://www.uclpress.co.uk/products/154646> [Accessed 15 Jun. 2020].
89. www.conwaylife.com, (2018). Forums for Conway's Game of Life. [online] Available at: <https://www.conwaylife.com/forums/viewtopic.php?f=2&t=3303> [Accessed 17 May. 2019].

8.2 Software and Hardware references

1. Anemone 0.4 (2015). Poznań, Poland: Mateusz Zwierzycki.
2. AnyLogic. (2000). France: The AnyLogic Company.
3. Block'hood (2017). Los Angeles, USA: Plethora Project.
4. Ender 3 Pro (2018). Shenzhen, PRC: Shenzhen Creality 3D Technology Co, Ltd.
5. Golly. (2005). England: Andrew Trevorrow and Tom Rokicki.
6. Grasshopper 3D (2007). Seattle: David Rutten, Robert McNeel & Associates.
7. Lucidchart (2008). Utah: Lucid Software Inc.
8. Lunchbox for Grasshopper (2012). Omaha, USA: Proving Ground Apps.
9. Mendeley. (2008). London: Elsevier.

10. NetLogo. (1999). Illinois: Northwestern University Center for Connected Learning and Computer-Based Modelling
11. Repast. (2006). Chicago: Repast HPC.
12. Rhinoceros, version 1 (1998). Seattle: Robert McNeel & Associates.
13. V-Ray for Rhino (1997). Sofia, Bulgaria: Chaos Group.

9 | Appendix

9.1 Grasshopper definitions – CE^{SESE}

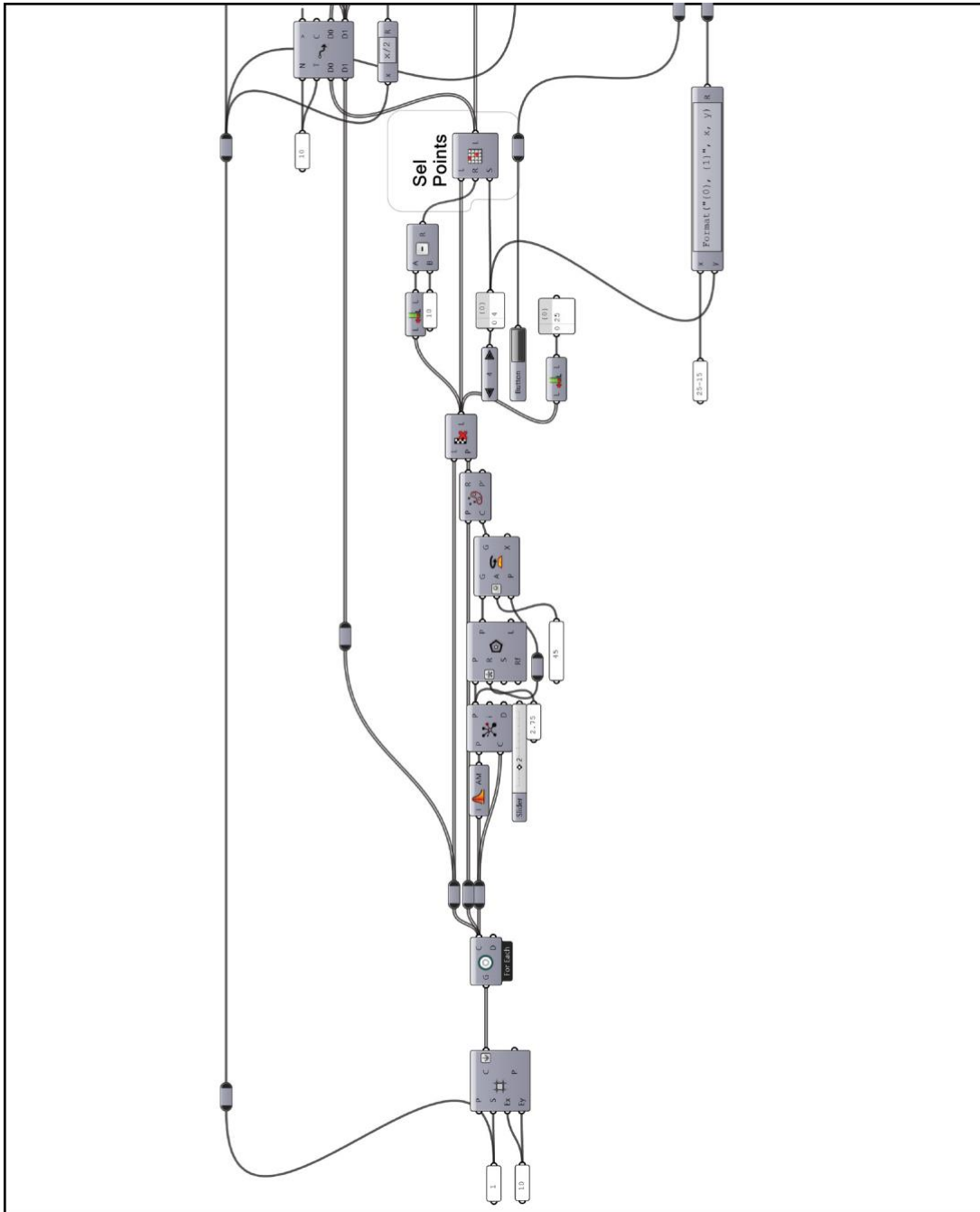


Figure 9.1 – Part 1 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-tower}$ as per 4.2

9.2 Grasshopper definitions – CE^{MESE}

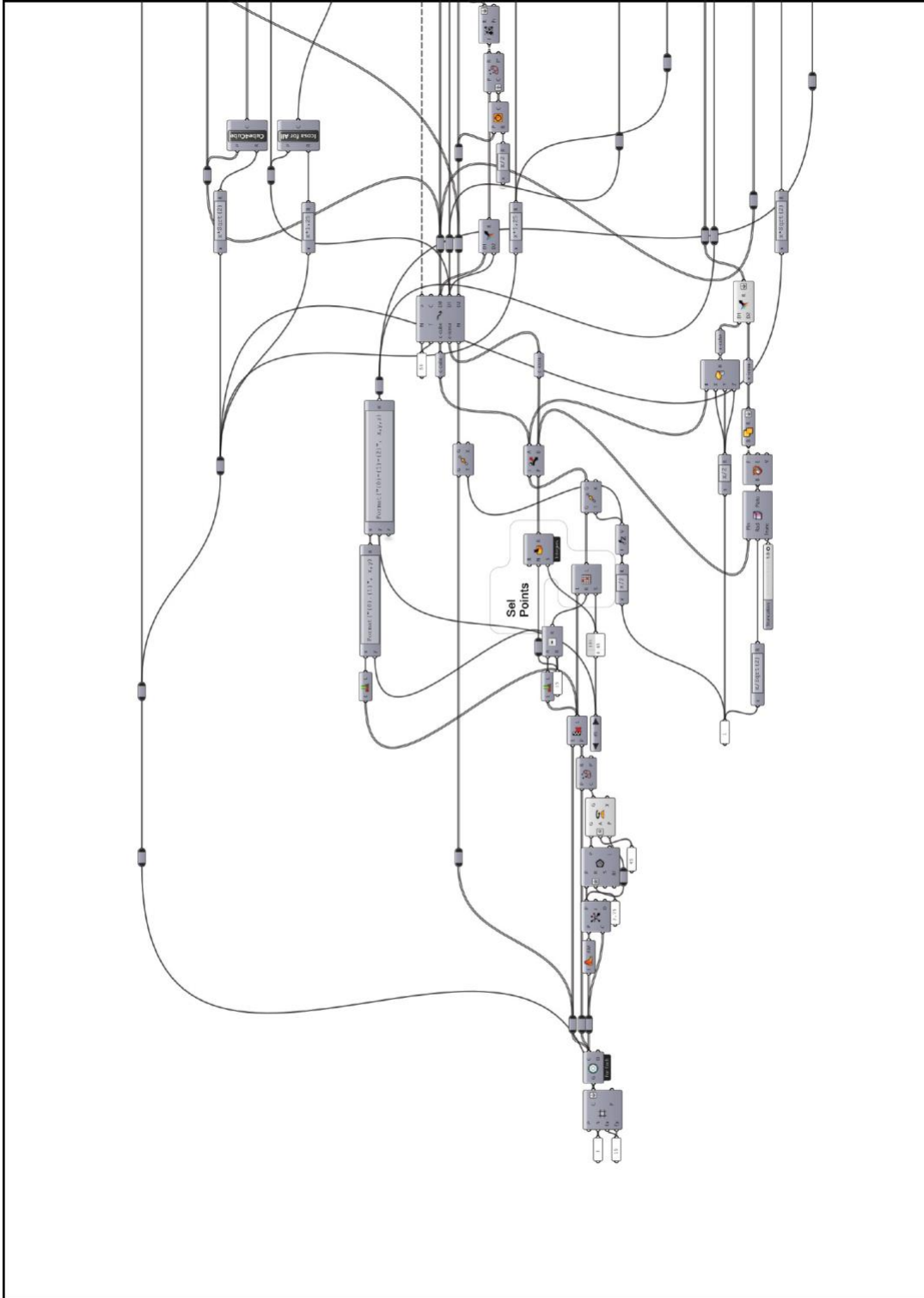


Figure 9.3 – Part 1 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-cubocta-tower}$ as per 4.3

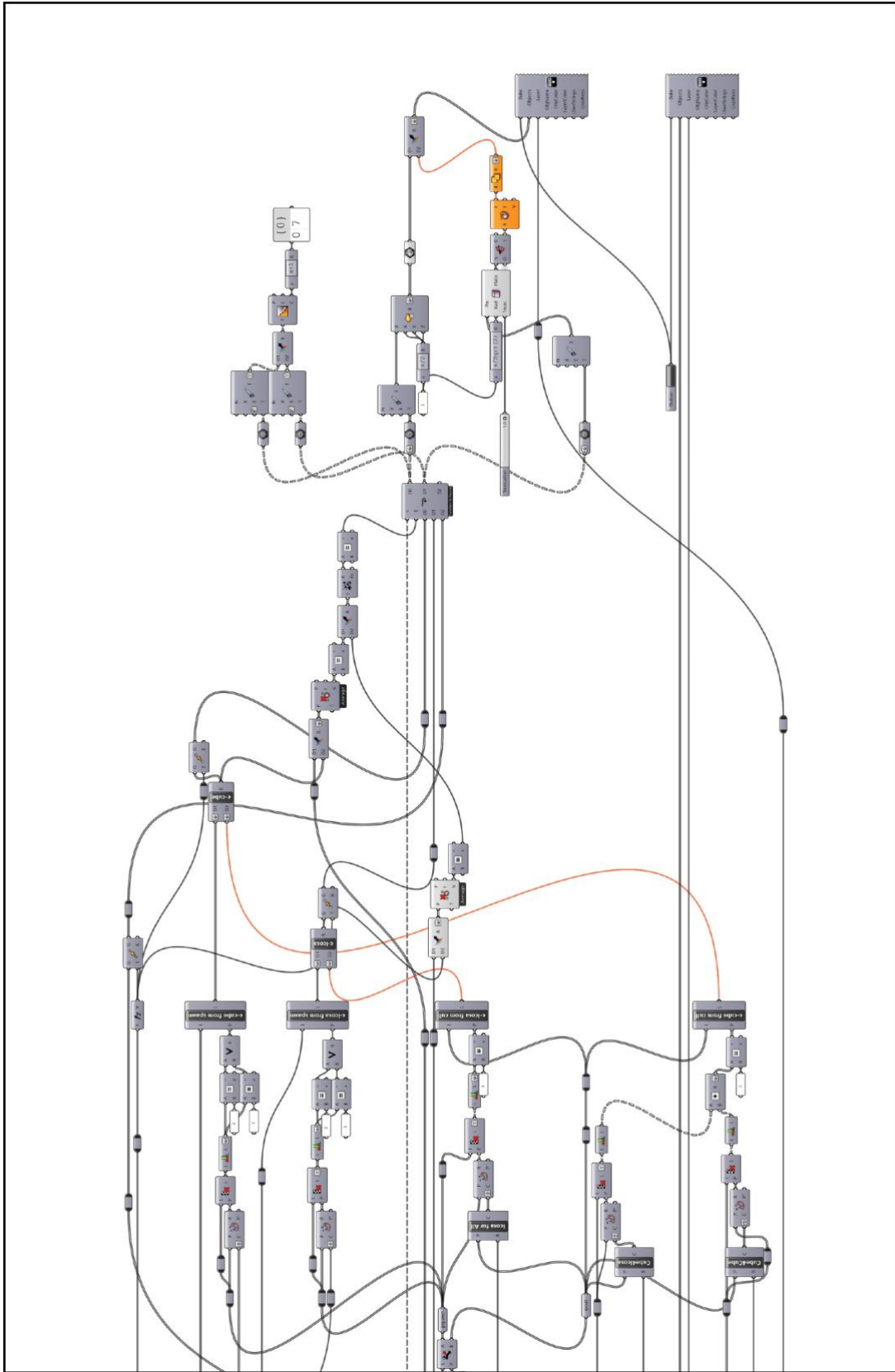


Figure 9.4 – Part 2 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-cubocta-tower}$ as per 4.3

9.3 Grasshopper definitions – CE^{SEME}

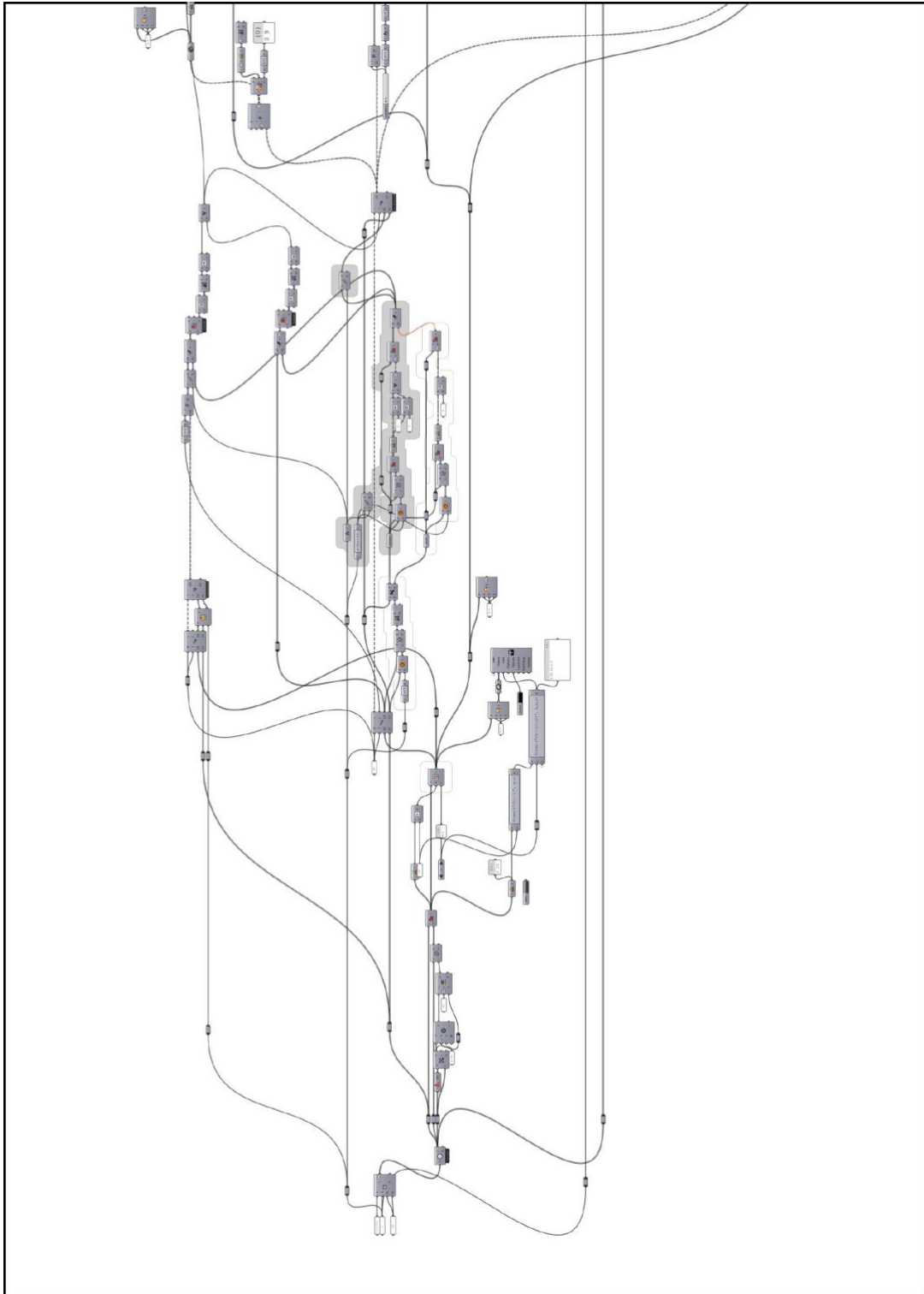


Figure 9.5 – Part 1 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-floating-tower}$ as per 4.4

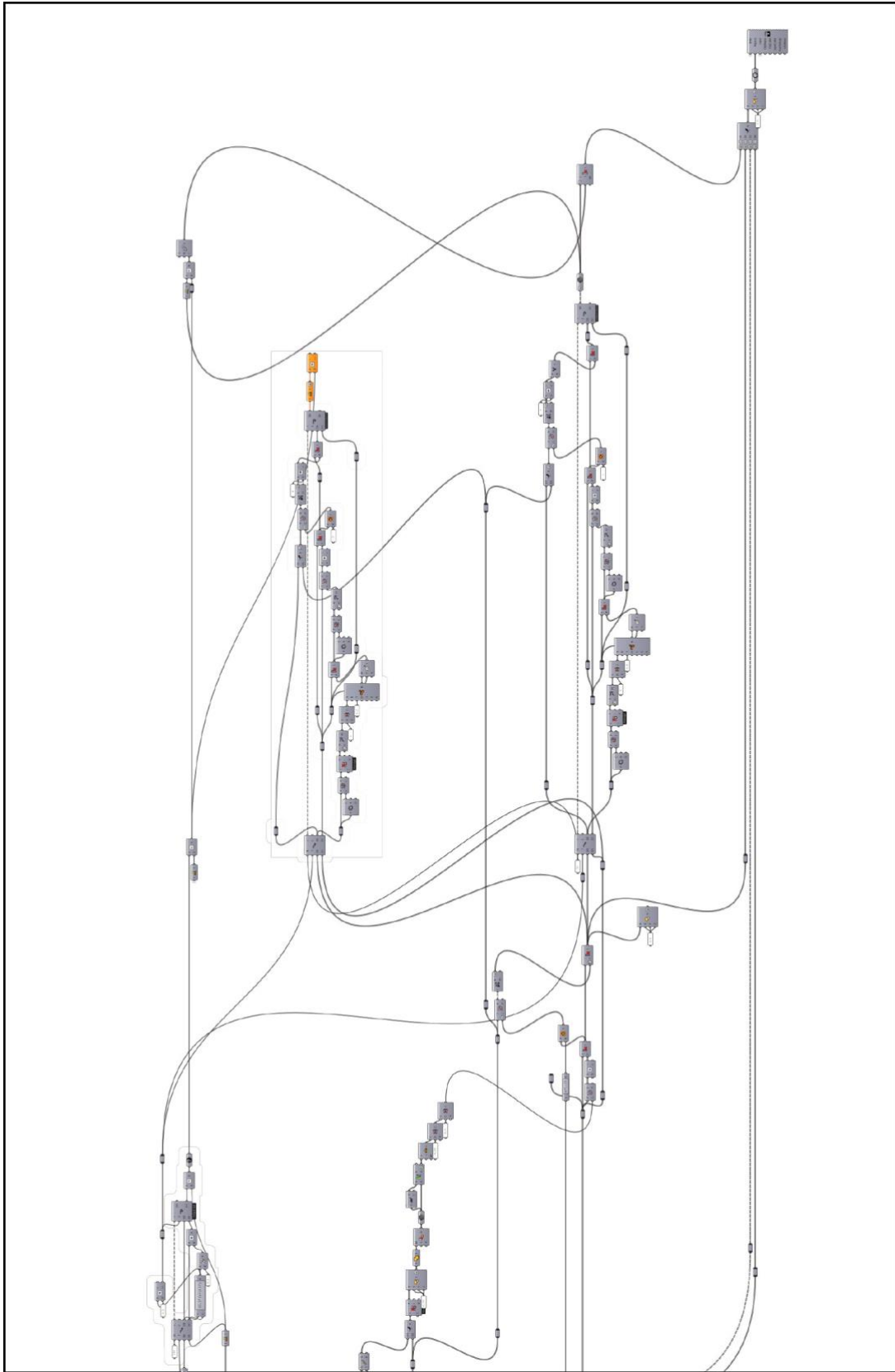


Figure 9.6 – Part 2 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-floating-tower}$ as per 4.4

9.4 Grasshopper definitions – CE^{MEME}

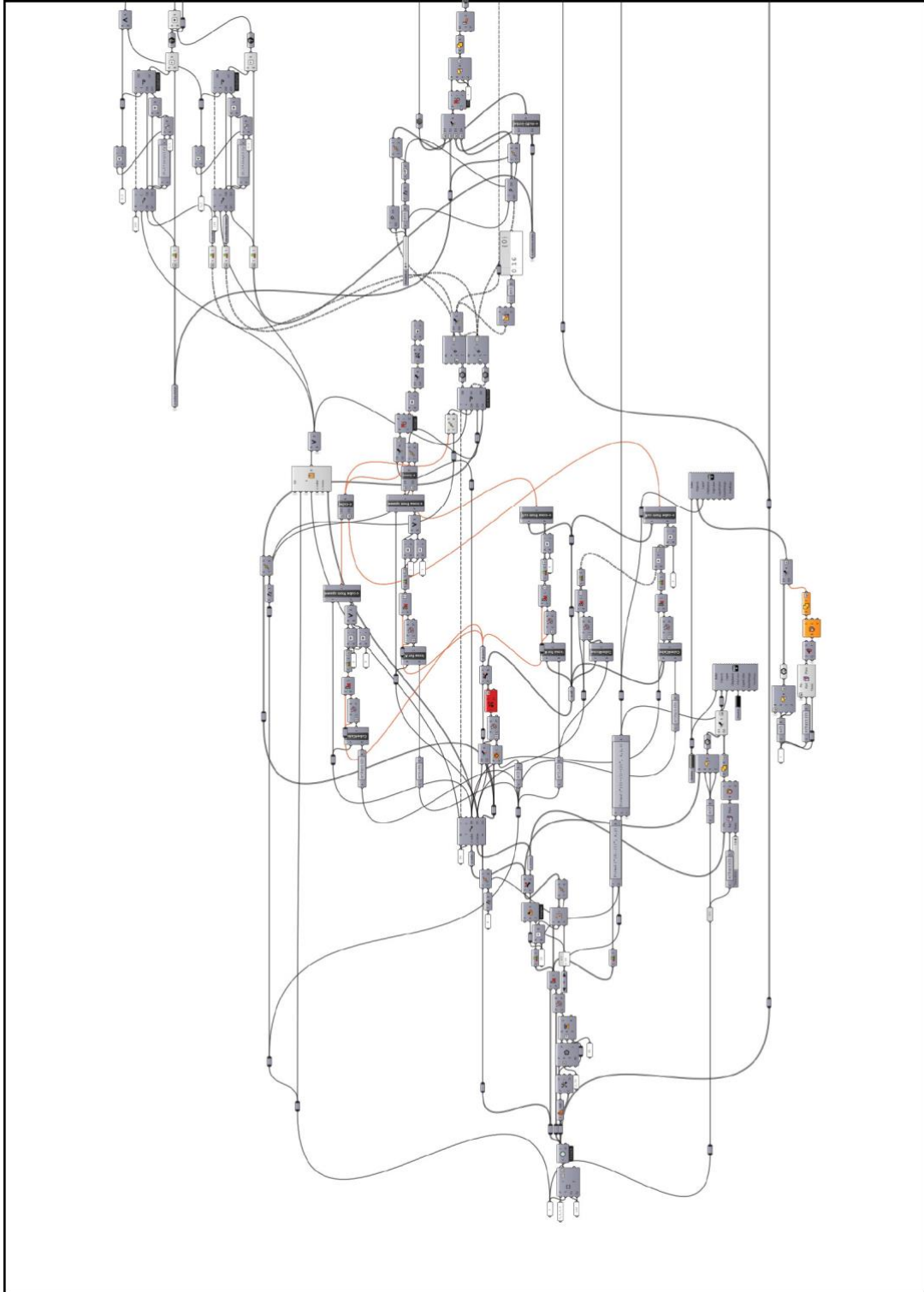


Figure 9.7 – Part 1 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-cubocta-floating-tower}$ as per 4.5



Figure 9.8 – Part 2 (of 2) grasshopper definition that was employed to generate, taxonomize and prototype the $CE_{cube-cubocta-floating-tower}$ as per 4.5