ADAPTABILITY OF GENERATIVE ALGORITHMS: A MEANS TO SUSTAINING THE DYNAMIC DESIGN PROCESSES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EKİN DAMDERE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE INTERNATIONAL JOINT DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF ARCHITECTURE
IN
COMPUTATIONAL DESIGN AND FABRICATION
TECHNOLOGIES IN ARCHITECTURE
BY
MIDDLE EAST TECHNICAL UNIVERSITY
DELFT UNIVERSITY OF TECHNOLOGY

SEPTEMBER 2010

Approval of the thesis:

ADAPTABILITY OF GENERATIVE ALGORITHMS:
A MEANS TO SUSTAINING THE DYNAMIC DESIGN PROCESSES

submitted by **EKİN DAMDERE** in partial fulfillment of the requirements for the degree of **Master of Science in Architecture Department, Middle East Technical University, and Delft University of Technology** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Assoc. Prof. Dr. Güven Arif Sargın _____
Head of Department, **Architecture**

Assoc. Prof. Dr. Mine Özkar _____
Supervisor, **Architecture Dept., METU**

Examining Committee Members:

Prof. Dr. Ahmet Can Baykan _____
Architecture Dept., METU

Assoc. Prof. Dr. Mine Özkar _____
Architecture Dept., METU

Assoc. Prof. Dr. Şebnem Yalınay Çinici _____
Architecture Dept., Yıldız Technical University

Assoc. Prof. Dr. Arzu Gönenç Sorguç _____
Architecture Dept., METU

Assist. Prof. Dr. Bige Tunçer _____
Architecture Dept., TU Delft

Date: 13.09.2010

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Ekin Damdere

Signature:

ABSTRACT

ADAPTABILITY OF GENERATIVE ALGORITHMS: A MEANS TO SUSTAINING THE DYNAMIC DESIGN PROCESSES

Damdere, Ekin

International Joint M.S. Programme of Computational Design and Fabrication Technologies in Architecture, in the Department of Architecture
Supervisor: Assoc. Prof. Dr. Mine Özkar

September 2010, 92 pages

This thesis is an investigation focusing on the adaptability of generative systems in a dynamic design problem, where the problem definition changes according to the changing conditions of the environment and transforming needs of the architectural space. This thesis, instead of discussing the dynamicity of the design processes, investigates the use of an adaptable generative system in a case-specific dynamic design problem to sustain its changing problem definitions. The research mainly looks into the potentials of generative systems in terms of adaptability and develops a generative system that is able to transform its structure in accordance with the dynamic constraints of a complex design process.

Keywords: dynamic design processes, generative design systems, adaptability, computational design tools

ÖZ

TÜRETKEN ALGORİTMALARDA UYARLANABİLİRLİK: DİNAMİK TASARIM SÜREÇLERİNİN SÜRDÜRÜLMESİNDE ETKİLİ BİR ARAÇ OLARAK KULLANIMI

Damdere, Ekin

Mimarlıkta Sayısal Tasarım ve Üretim Teknolojileri Uluslararası Ortak Yüksek Lisans Programı, Mimarlık EABD
Tez Yöneticisi: Doç. Dr. Mine Özkar

Eylül 2010, 92 sayfa

Bu tez, uyarlanabilir türetken sistemlerin, problem tanımlarının değişen çevre koşulları ve mimari mekanın dönüşen ihtiyaçlarına göre değiştiği dinamik tasarım problemlerinde uygulanması üzerine yoğunlaşan bir araştırmadır. Bu tez, tasarım sürecinin dinamik olup olmadığını tartışmak yerine dinamik tasarım probleminin değişen problem tanımlarının uyarlanabilir türetken sistem kullanımıyla sürdürülebilir olmasını araştırır. Tez, bu tür algoritmaların türetilen tasarım ürünlerinin tekrar dönüştürülmesi konusundaki yeterliliklerini inceler ve kendi strüktürünü karmaşık tasarım süreçlerinin dinamik sınırlayıcılığına uygun olarak uyarlayabilen ve dönüştürebilen bir türetken sistem geliştirmektedir.

Anathar Kelimeler: dinamik tasarım süreçleri, türetken algoritmalar, uyarlanabilirlik, sayısal tasarım gereçleri, türetken tasarım sistemleri.

To My Family,

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor Assoc. Prof. Dr. Mine Özkar for her guidance, advice, constructive criticisims and most importantly patience throughout my studies.

I am also grateful to my jury members Prof. Dr. Can Baykan, Assoc. Prof. Dr. Arzu Gönenç Sorguç, Assoc. Prof. Dr. Şebnem Yalınay Çinici, and Assist. Prof. Dr. Bige Tunçer. And I also would like to thank my program coordinators Assoc. Prof. Dr. Arzu Gönenç Sorguç and Assoc. Prof. Dr. Rudi Stouffs.

I especially offer my sincere gratitude to Assist. Prof. Dr. Bige Tunçer for her endless support throughout my studies abroad. It would have been impossible for me to come this far without her encouragements. I also would like to thank Huib Plomp and Sander Mulders for their valuable advice.

I would like to thank all my friends who did not leave me alone even I am far away. And I especially thank my fellow housemates for sharing their opinions, being with me all the time and making it bearable to be far away from home.

Finally, I would like to thank to my family. I am forever indebted to my parents, Hülya and Ali Damdere. Nothing can be enough to express my gratefulness for their support during my life and my studies. They never leave me alone. I also would like to thank my sister İlke Damdere for her encouragements and helping me to stay calm every time I fall in despair.

Thank you all very much indeed.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

CHAPTER 1

INTRODUCTION

With the advancing discussions about design processes, problem definitions for design problems are considered to be dynamic and desired to be adaptable to the changing conditions of the environment and transforming needs of the architectural space. This thesis investigates the use of generative systems and emphasizes the adaptable potentials of such systems to support the dynamic design problems. These dynamic design problems where problem definitions are revised and redefined considering the changing environment and the architectural object needs to be redesigned according to these changes, form the foundation of the argument of this thesis.

The generative system in question should be able to detect the transformations of the changing problem definition. The ability to detect these changes can be provided by emphasizing their adaptable properties. Generative structures are capable of adapting to the new objectives when the environment changes and these potentials develop from the manageability of their reconfigurations.

Discussions about design problems with changing problem definitions have gained importance. These changing design problem definitions require adjustable processes where parameters and constraints of a problem change and as a result the course to pursue a design proposal is altered. This need for changes in the definition of the solution ends up in a recurrence of transforming the end product every time the definition changes. It can be exemplified by the changing functional descriptions of a space. A problem definition instructing to design a living room next to an entrance hall, with respect to aesthetical concerns can be changed to instruct

to design a kitchen between the living room and the hall. This thesis suggests the use of adaptable generative systems to carry out this matter of dynamic design problem definitions and their demand for transformation during their existence.

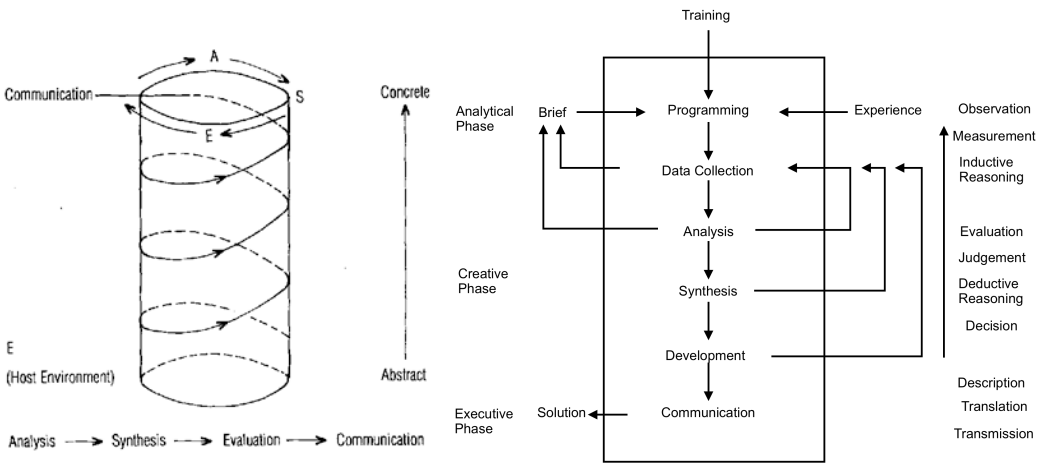
Changing design problem definitions and dynamic design processes can be exemplified in other scales. Generally a building is constructed in a given site where the external and contextual conditions are fixed for that specific period of time. When the site conditions change, considering these changes the building needs to be redesigned. For example when the land piece surrounding the building is transformed into a water piece, design data obtained from the previous site analyses change. So, problem definition changes. Therefore, the building needs to be redesigned considering the water piece around the building site. Defining, evaluating and redefining the design problem result in a dynamic design process.

The idea to handle the dynamic design problems and to investigate the adaptability of generative design systems in addition to the use of such tools only to serve as generation mechanisms came along with the problematic of my graduation project. The need arose to contemplate on the issue of changing definitions of a design problem and to emphasize the adaptability of generative structures to provide design solutions in dynamic design problems. This graduation project with a very suitable design task paved me the way to work with this topic. Therefore, the argument of this thesis is developed specifically for the dynamic design problem this graduation project introduces. Instead of stating and generalizing that all design processes are dynamic, this thesis deals with one specific dynamic design problem and suggests solutions to sustain this dynamicity.

Problem definition of the project is to develop a program best fit to serve to the needs of the site determined by the project. Under the project title “Architecture, Infrastructure, Mobility” the task is to individually develop the problem and program definitions relating to the issues discussed in the project site. I elaborate more on the project in chapter 2.

This thesis does not discuss whether the architectural object should be dynamic or not. Rather it investigates the dynamicity of a design process where through changing problem definitions the design proposal is transformed according to the new needs of the architectural space and the environment. The issue is not the ability of the architectural object to modify its built-scape responding to changes. As a result of the change in the design process, the design proposal is transformed.

The increasing demand for the utilization of computational tools leads the way to substitute these tools to support decision actions in design processes. Acknowledging this integration of computational tools in design processes, I argue that, it is possible to make use of the tools specifically generative systems to maintain the changing design problem definitions. To understand how this argument can be put into practice, it is important to make clear in which stages of design processes these computational tools are integrated. Rowe (1987) reviews structures of design processes and decision mechanisms developed between 1960s and 1980s. The “iconic model” in Figure 1a represents the design process as a linear sequence of stages where Archer’s model (Figure 1b) refers to the design process as continuous feedback loops. The continuity of feedback loops in Archer’s model brings a new perspective to how to understand a design process (Rowe, 1987). When we look at these two models of design processes, the notion of adaptability in the sense that it originates from the nature and the complex structures sustaining the evolutionary processes is not integrated in any of the stages specified. The adaptable definition of a synthesis stage is implemented between analysis and development. After defining the design problem, data collecting and analysis of the data obtained, in the synthesis phase where this data is processed into a design output, an adaptable generative system can be put to use (Figure 2).



a. b.

Figure 1 a) An iconic model of a design process, Rowe (1987) reviews Asimov's speculation of a design process in relation to the "iconic model" referencing to Watts (1966) and Mesarovic (1964) b) Archer's model of a design process (Rowe, 1987)

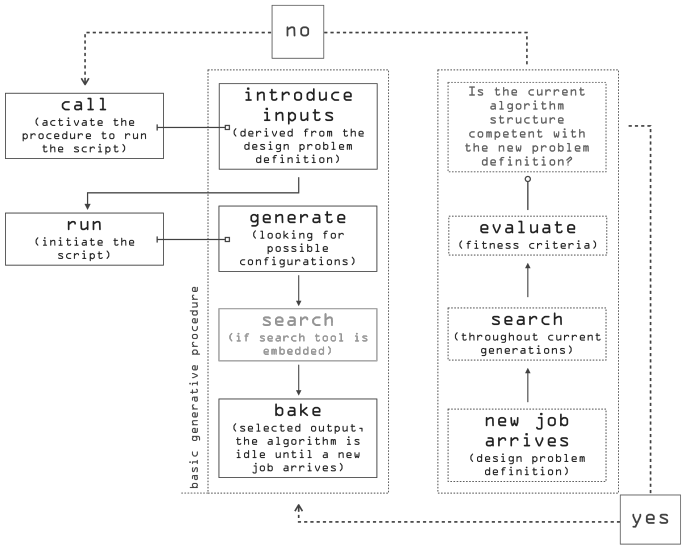


Figure 2 This adaptable process is embedded between the analysis and the development stages in design processes.

The demand to develop an adaptable generative system fit to cope with changes throughout the lifespan of this generative system already exists in recent software engineering research. The idea to adapt these adaptable generative systems in dynamic design processes to provide the dynamicity of such design problems is the main concern in this thesis. Therefore, the adaptable generative algorithm in question is built with regard to the examples of existing adaptable structures. The adaptability of generative systems is investigated through studying the use of artificial generative systems, autonomous iterative systems, and biological systems. These models are looked through to relate design processes and natural processes in generation to one another, and to find ways to support the transformation of a generated system as a way to sustain the emergence of new alternatives and to adapt alternatives to the dynamic constraints of a complex design process in addition to the issue of the adequacy of such systems to come up with a series of possible design alternatives.

The main aim of this study is to design a generative structure so that it is able to adapt and transform its structure in relation to the changing parameters of a design problem. These transformations are the processes, which the generated output is going to be subjected to in accordance with adjustments of the problem definition specific to the design problem. I propose to develop a generative system to manage the transformation of the solution space and to maintain the adaptability of the design problem definition to changing environmental impact, where environment refers to a mechanism constantly changing the external stimuli.

How to achieve the adaptability of a generative structure is investigated through the discussions of designing a modular system to manage adaptability and to control how this generative system responds to its changing environment. In relation to swarm intelligence and self-management of such systems, with decentralized local interactions between the components of the generative system and the emergence of a global behavior as a result of these interactions, it is easier to manage adaptability of such a system. This thesis also investigates the integration of mobile elements –agents- in a generative structure to handle these local communications between the components of the system and set up call mechanisms to deal with transformations of the generative parts. These communication agents are

implemented in the generative system in collaboration with the generative modular structure of the system and they provide information flow between the components. Besides communicative properties, they are used for “begin” and “end” commands to control the running of the generative parts in the algorithm.

This thesis aims to develop a generative system that can adjust its structure and transform the output, which is generated and selected, according to the predefined parameters. These parameters are determined by the changing design problem definition, which is controlled by environmental factors. These factors are external influences and they are contextually determined, specific to the problem definition. Environmental factors are not specified only by climatic conditions. In addition to them, environmental factors are relevant to the site definitions –soil conditions, border conditions, and scenery of the assigned land piece, functional relations and user needs. Functional relations are remodeled in order to satisfy the changing environmental factors and user needs. The user/inhabitant/observer, according to interpretation of space-function relationships, is able to adjust the designed mechanism –in this case the output, to fulfill what is needed for the design mechanism to fully operate. In short, the structure, which is constantly changing, while changing its structure, also transforms the outcome.

I also suggest that the customization of adaptable generative structures can be practiced as a means for the user to adapt these generative systems into their own design processes. The structure in question can be designed so that it can be customized with respect to the user needs. Also, this structure can be generalized as a system consisting of modular units, so that it can be practiced in any design problem. As a result, the issue of customization is handled as a parameterization of these generative structures to be modified by/for the user. The issue of generalization is discussed through the adaptability of these systems to the requirements of any selected design problem. The discussion about the issue of customization and generalization of generative systems is not delved into but it is suggested as a further study to be conducted.

Below are the keywords essential to the development of the argument in this thesis. *Adaptability* in the sense that it is an adjustment of a mechanism to subsist in its

environment can be defined as the ability to gain crucial qualities to sustain its existence. Oxman and Oxman (1992) state that adaptation is modification through transformations on specific design descriptions and they talk about adaptation as the adjustment of the end product to fit to the changing environment. It is the adaptation of the end product. I describe the adaptability as the ability of a system to adjust its own structure to the changing conditions. As a result of the adaptation of the generative system, the end product is redesigned according to the new rules. It is not the ability of the end product to adapt to the changing conditions, but it is the ability of the generative design system.

According to Boer and Canamero (1999), adaptation is a system's potential for transforming its behaviors according to its context and refining its performance over time. Adaptability in other terms relates to the property of being able to accept future changes over time by altering their overall designs spontaneously according to the instant feedback from a number of agents – simple units made use in the system (Narahara, 2009). The definition of adaptability as the competence of a system to adjust according to changing conditions is taken into consideration throughout this thesis.

Fundamentally, adaptability is closely associated with the effort to survive under the changing conditions of the environment. Here, the *environment* is connoted as the overall structure, which the definitions of dynamic design problems govern and it serves as the settings defined for a generative system. *Dynamic design problem* in its definition has the notion of dynamicity since the architectural object in question is constantly changing with transformations of the design task. Therefore, the architectural object has to survive while satisfying transforming needs of the design problem. Adaptability is the key for a system to maintain its existence. It is not in the scope of this thesis that the architectural object should also be responsive and adaptable to the changing conditions of the environment. But the use of adaptable generative tools to handle such changing conditions is the main concern of this thesis.

In order to study an adaptable generative system, I develop a structure for a generative algorithm where an *algorithm* is a tool made up of individual elements,

which have their unique statements and behavior to accomplish an objective given. It is not handled as in the sense of a mathematical algorithm, but it is still a process consisting of determinate steps to achieve a goal. The user introduces a definition of the problem to an algorithm as its definition. The problem definition is changing according to defined parameters determined by the environment. According to Rocker (2006) an algorithm is a finite set of discrete and well-defined instructions for accomplishing a predefined task, and they essentially organize and determine the computer's sequence of operations when a program is running. According to Terzidis (2006, pg. 65) "... an algorithm is the abstraction of a process and serves as a sequential pattern that leads towards the accomplishment of a desired task." Along with these definitions, an algorithm is defined as a tool, which is instructed with determinate and specific task definitions to accomplish desired objectives.

I also investigate adaptability in relation to *evolution.* Genetic algorithms function on the inheritable properties of the individuals where these inheritable properties are the fundamental elements of evolution. The issue of evolution is linked to adaptability, where evolution refers to the adjustment in the inheritable properties of individuals composing a population in order to survive in the changing environment conditions. Renner and Ekárt (2003) also refer to evolution as an emergent property of evolutionary systems, in which new individuals inherit and adapt to survive for the next generation. The concept of subsistence in nature therefore necessitates the ability to adapt. As well, this thesis states that in order to speak of an adaptable system it is crucial to find the links between evolution and adaptability.

Fitness criteria in relation to the issue of evolution are values according to which the solutions generated for the problem definition are evaluated and selected if they satisfy these criteria. They are defined and introduced as dynamic constraints in the generative system used in this system and they refer to the changing specifications of the design problem definition that a proper outcome should satisfy

In order to make the scope of this thesis clear, this thesis is structured in three parts. The first part of this thesis is to mention the project at hand and defines the boundaries of the scope of this thesis. It comprises the introduction chapter and chapter 2 introducing the project, which is the main influence on developing the

thesis argument.

The second part of the thesis is a general introduction of generative design systems and examples related to the issue of adaptability to provide an overall discussion about the potentials of such systems. It also specifically talks about generative structures in design processes to elaborate more on these structures to sustain dynamicity of design problems. To investigate generative design systems and specifically focus on generative algorithms, literature based analyses will be made in order to attract attention to the adaptability of such systems. Examples and issues related to characteristics of these generative systems that can be associated with the notion of adaptability are studied. Prominent features these systems possess are pointed out. This part comprises chapter 3 and chapter 4.

The third part of the thesis is concerned with how to develop an adaptable generative algorithm to support the dynamicity of design problems and also give clues about how this algorithm is implemented to a realizable project. This part also includes the results, the discussion and conclusions of this thesis about the validation of the developed generative algorithm in terms of adaptability. The third part comprises the chapters 5 to 7.

Chapter 1 provides an introduction to this thesis and the problem definition, while clarifying the vocabulary.

Chapter 2 focuses on the argument discussed over the graduation project and describe the design problem definition and the design process. This chapter tries to depict the correlations between the argument behind the thesis and the concept of the design task.

Chapter 3 provides a general introduction to the generative design systems and examples related to the issue of adaptability and an overall discussion about the potentials of such systems. This chapter investigates some evolutionary models such as genetic algorithms, cellular automata and swarm intelligence and their biological references followed with discussions about issues of self-management, autonomicity and emergence associated with the issue of adaptability. This chapter

provides an introduction to the following chapter and prepares the ground for discussing the adaptability of generative algorithms in the chapters that are to follow.

Chapter 4 focuses on the use of generative algorithms in design processes. While examining different applications of generative algorithms, this chapter tries to inquire the adaptability of such systems in relation to dynamicity and autonomicity of these systems. Autonomicity in the sense that it is the ability of an application to initiate events without affecting the overall structure of tool is important in terms of achieving self-manageability in the generative system. This chapter also prepares the ground for developing an adaptable generative algorithm and talking over the potentials of this designed system when the adaptability is the main question.

Chapter 5 presents the structure for an adaptable generative algorithm, which is developed and designed over the arguments and discussions introduced in previous chapters. Specific examples of generative tools and algorithm structures are investigated and studied in detail to form a framework about how to structure a generative system. In this chapter, generative algorithm structure and which elements form the organization of this algorithm is explained in detail. Integration of the concept of agents to the generative system and as a result of this assembly the composite structure of the algorithm is presented. Algorithm parts, its modular structure and implementation of different types of agents referencing back to the examples examined are set forth and the discussion of adaptability is pursued over this generative algorithm design. Specifically, levels of the generative process are described and how the agents are implemented in the algorithm structure is presented. A sequential flow developed for the adaptable algorithm is presented. How to integrate this adaptable generative algorithm to the graduation project is also proposed as a further study in this chapter. This chapter also investigates how an adaptable generative algorithm can be personalized and generalized.

Chapter 6 provides results and discussion of the arguments constructed in this thesis. This chapter also talks about the potentials and limitations of the study and draws some conclusions and presents a summary of the study. It evaluates the methodology of the thesis, contributions of this study and suggests further studies.

CHAPTER 2

ADAPTABILITY IN AN ARCHITECTURAL DESIGN PROCESS

2.1. Introduction

The arguments of this thesis are attained and developed from the concept of an architectural project. This project for designing “A Research Institute for Environmental Conditions” in the Haarlem / Spaarnwoude region in the Netherlands is proposed for the graduation requirements of the graduate program this thesis is also a requirement of.

Design task determined by the project definition indicates that the design solution will serve to the needs of the project site. I find it intriguing to study the transformations of the landscape while analyzing the site and its past. Dynamic structure of the land use patterns and correspondingly dynamic needs of the land, lead me to think about how to compensate these transformations on the land-scale and convert these needs into an architectural project. This understanding of the project entails that the design problem I will define for that piece of land will change after a period of time, when the land transforms again.

Since dynamicity necessarily brings along the notion of time, adjustment of the design system to the changing conditions in time is inevitable. All of these statements assert that any architectural object designed for the land needs to be altered after a period of time with the transformation of the land. Therefore, the problem definition also needs to be dynamically designed. This brings me to think about how to provide a continuous problem defining course and a dynamic design process. A generative design system seems suitable as a first step towards

attaining the dynamic character required in the design. As it is going to be discussed in the following chapters, generative approaches are beneficial in terms of self-managing their behavior in order to adapt to the changing environment. The question is not about the adaptability of an architectural object according to the changing conditions or not about transforming it every time the problem definition changes with the environment. Rather, it is to develop an adaptable generative design system to be introduced in the very early phases of the design process, to design with an adaptable system, which itself will be transforming to sustain this dynamic design process. It is possible that all the design input obtained from the analyses and put in the problem definition after a period of time is invalid because the environment providing the design input can be transformed into some other environment, which is totally different than before. It means that since the design input is changing, the problem definition also needs to be changing. This is the essence of a dynamic design process.

This graduation project forms the foundations of the argument this thesis puts forward. I wanted to contemplate on the issue of dynamic definitions of a design problem and investigate on how to provide design solutions to sustain this dynamicity. This graduation project with a very suitable design task provided me the problem definition to work with this topic. To maintain changing definitions of the design problem, the issue of adaptability is the connecting piece between the dynamic design problems and the use of generative algorithms in architectural field. As a conclusion, I investigate how generative algorithms can be utilized in dynamic design processes so that the dynamicity of the design problem is sustained and the transformation of the design solution is achieved.

2.2. The Design Problem: A Research Institute for Environmental Conditions

The main aim in this project, is to study the existing landscape through ongoing land transformations and design a research centre, which will house research and experiments on the major dangers threatening the landscape, climate and soil conditions. This institute holds experiments to overcome environmental problems in the Netherlands on a real-time land-scale. Inspired from the land transformations in the Netherlands, the fascination is to represent these transformations of the Dutch

lowlands in relation to the transformations and developments of technology. The story of constantly shifting landscape / shifting language of the Dutch landscape is exhibited and the memory of the lowlands is represented through keeping the track of the transformations of the landscape. The theme is based on the components that Dutch landscape introduced to the literature, the unique language that it generated over time. Experiments are applications of scaled experiments to a land scale to take measures for the possible future scenarios. Referencing these possible future scenarios its objective is to predict what will happen if these disasters occur. It is a prediction system, which will be realized on real land scale.

2.3. The Design Process

This graduation project is conducted with a number of phases. Starting with defining the constraints and parameters of the design problem and identifying what the design problem indicates, all design data obtained from the analyses are put together to define the design problem considering all aspects of the design task. These design data are the information about the site[1], visual, cultural, functional and performance expectations of the final architectural object, mainly all the input obtained from the design analyses.

After problem definition, information obtained from the problem inputs are analyzed and knowledge is created accordingly. These analyses cover the soil conditions in Spaarnwoude and the transformation of the land during its existence. The intriguing point also constituting the concept of this project is the constant transformation of the land still going on all through the lowlands. This knowledge of transformation is converted to transformation mappings to grasp the concept of change as much as possible (Figure 3). As a design concept, to present the transformations of the landscape in relation to the transformations and developments of technology throughout the built-scape, the argument of a built-scape having its own lifecycle

[1] This project has a specific site determined through the design task. It is not this thesis' concern if there is no site defined by the design problem. Also the generative model assumes that there is site input to be converted to computational data for the mapping module to run. In a case where there is no site determined by the project, it is still possible and necessary to introduce site data for the final built-scape to be designed.

changing its diagram like the land changes its structure is adopted. The built-scape in question while reacting to the transformation of the landscape and the environmental changes, it also transforms its structure redefining the design problem every time a transformation occurs in the landscape.

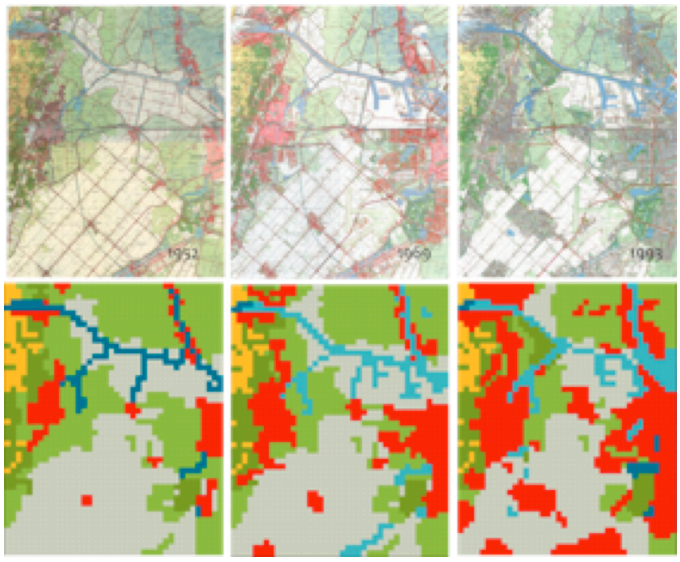
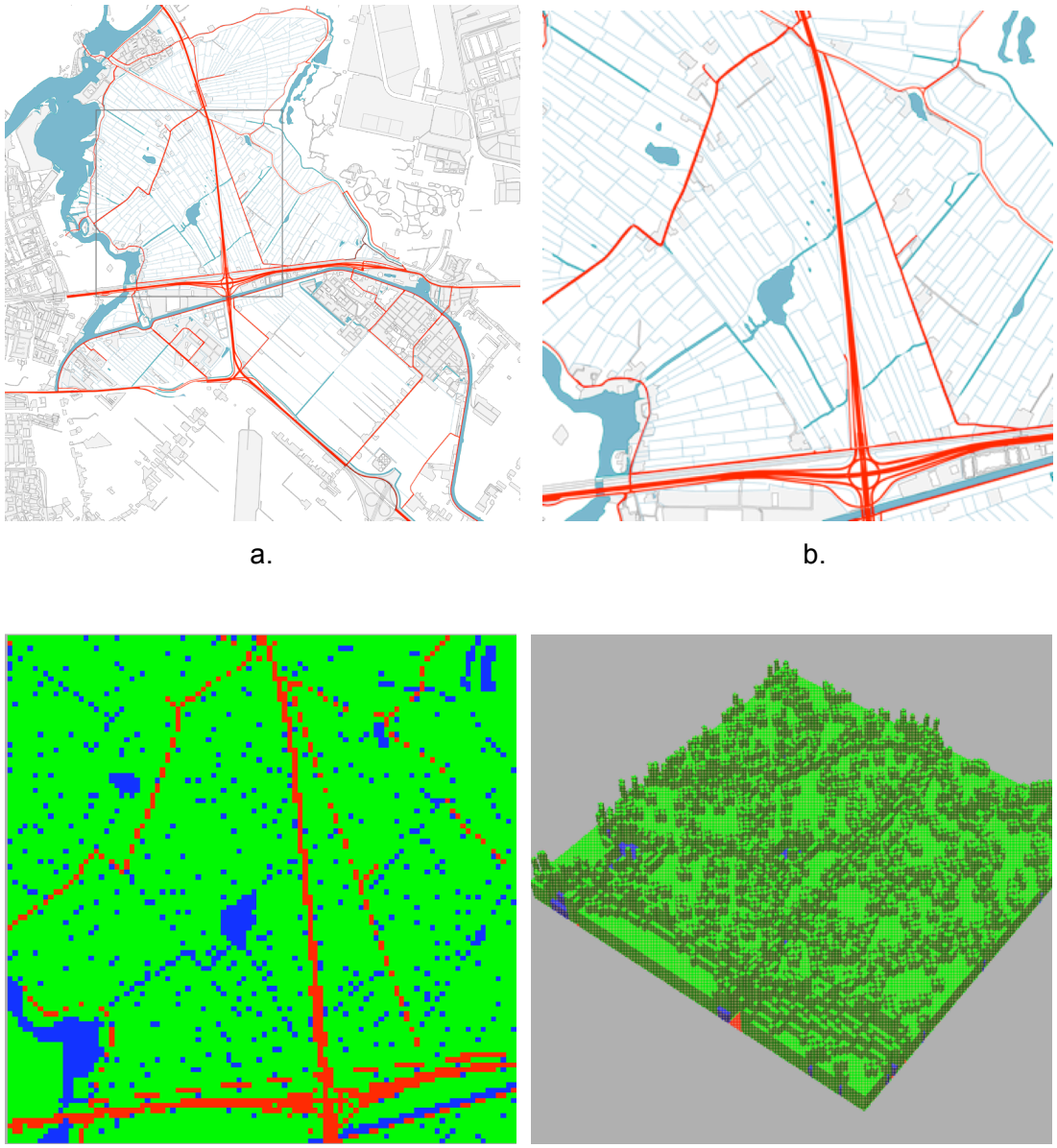


Figure 3 Transformation mappings prepared for the process of land transformations in Haarlemmermeer region. These mappings are analyzed in order to understand the transformation process of the Dutch lowlands. Land use in relation to soil conditions, and transformations of land into water and water into land are analyzed. With a similar approach, land values are introduced to the computational tool.

Figure 4 illustrates the conversion of the design input obtained from site analyses into computational data to be used in the adaptable algorithm. Figure 4a is the site plan provided for the project. A smaller region of the site is selected and the site image is converted to a computational image. This image is composed of a grid of land pieces, and each of these land pieces is assigned a value with reference to their land conditions. Red represents the road, blue represents the water, and green represents the land. The sizes of the grid pieces depend on the desired level of

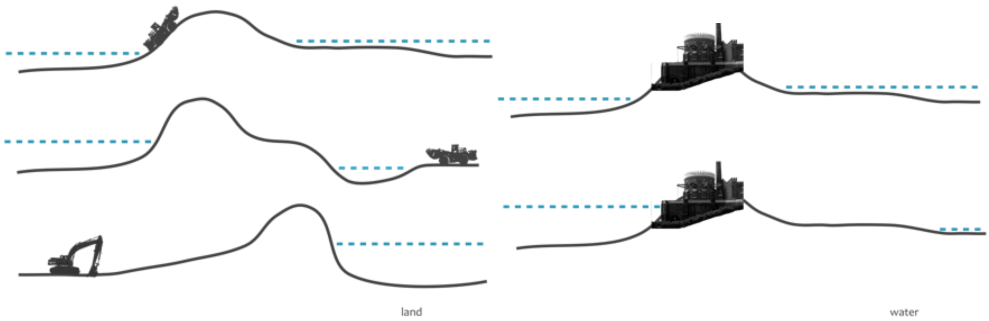
detail. After trying different grid sizes, I decided on this grid size for that the level of detail was enough to represent the relations of the land pieces.



a. b.

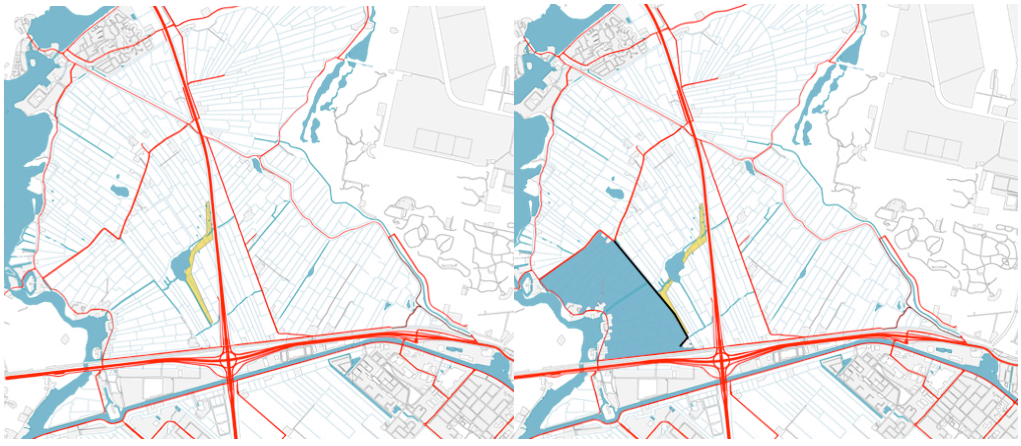
c. d.

Figure 4 Conversion of design input to computational data. a. site plan b. selected part of the site c. selected site redefined in distinct colors depicting the land pieces attributes, ready to be implemented in the generative system d. design input (site image) converted to computational data, depicting land values assigned on the converted image.

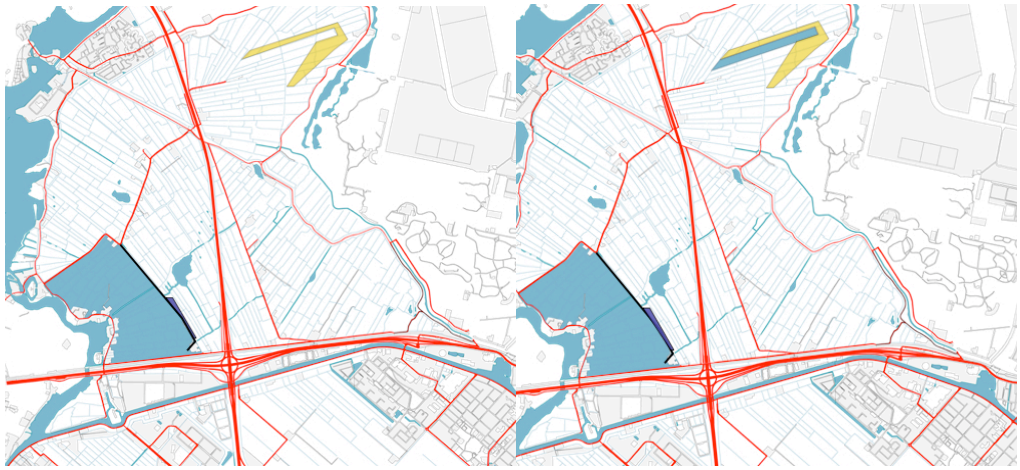


a. b.

Figure 5 a) Transformations of land b) Transformations of water



a. b.



c. d.

Figure 6 Site in Haarlemmermeer region, transformation of the landscape and transformation of the built-scape

Throughout the lifetime of the built-scape, in accordance with the experiments conducting on land-scale, land transformations are also applied periodically to the landscape of the building site. For every event, there is a specified time period. These events will change the profile of the land. The built-scape will be transformed to contain the necessary building components for that specific experiment. So, according to the various land transformations, the built-scape components will also adapt themselves. Land and water transformations are represented in Figure 5a and 5b.

According to the experiment type, the land piece where the experiment takes place is transformed into a new land piece in order to serve for the needs of that specific experiment. A diking experiment to control and prevent future floods requires the transformation of the land stacked like a hill. According to land transformations, built-scape also transforms. An example of a transformation of landscape and relatively transformation of the built-scape is represented in Figure 6. According to this figure, after selecting a suitable land piece for the built-scape as a result of converting site input to computational data, with the determination of the experiment the first baked moment of the built-scape is constructed. The first experiment is a diking experiment, where the built-scape is articulated and transformed according to the construction of the dike. Figure 6a is the first instance of the built-scape, in Figure 6b the diking experiment takes place and the built-scape is transformed according to the needs of the experiment. After the experiment is completed and if another experiment will take place according to the schedule of events, the built-scape is transported to the site where the experiment will take place and transformed into another built-scape functioning according to the experiment's requirements (Figure 6c and Figure 6d).

Transformations of the built-scape in harmony with transformations of the landscape are managed with the constantly transforming design problem definition. My discussions evolve around the idea of bringing dynamicity of the design problems and iterative processes of generative algorithms together and developing an adaptable generative algorithm to sustain the transforming definitions of design problem. In addition to the utilization of generative algorithms in architectural field as computational tools for the sole purpose of facilitating form-finding or supplementing

a great number of solutions to choose from, I explore what is more to the use of generative algorithms in terms of adaptation to support dynamic design processes.

The design process model by Archer presents the feedback loop mechanisms, which support the communication between phases of the design process. Also, Simon (1968) stated that it is crucial to provide feedback mechanism to an evolutionary system for it to adjust according to the environment's variations. Evolutionary mechanisms are brought into question for their properties concerning the adaptation to the environment to stay alive. Based on this statement, feedback and feed-forward mechanisms provide information about the system's current conditions.

In this project, feedback loops are designed as a model to provide the self-awareness of the system. To elaborate more on the graduation project design process, Figure 7 presents feedback loops, which are put to use in order to clarify, redefine and adjust the design mechanism.

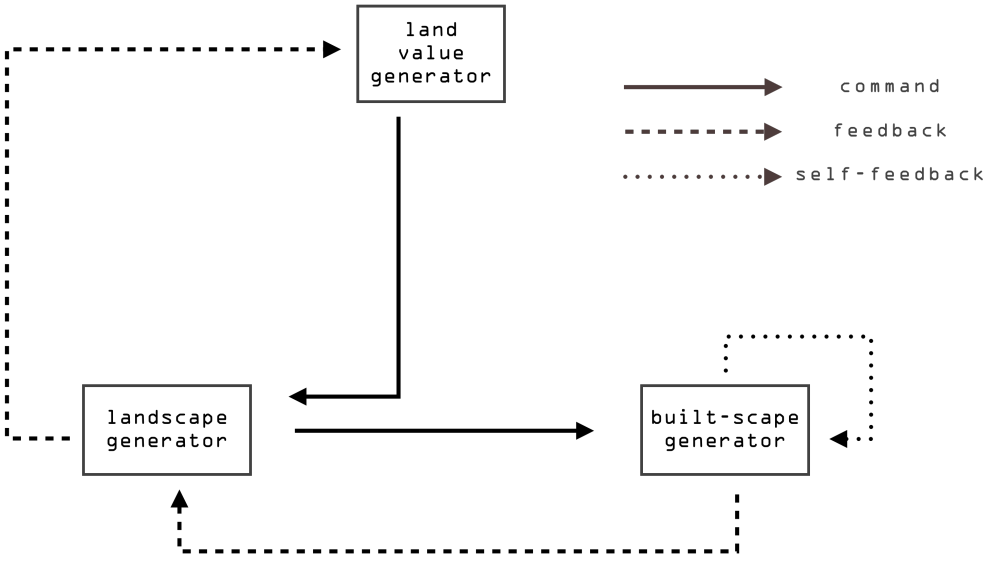


Figure 7 Main scheme for feedback loops present in the design problem

In Figure 7, land value generator analyzes the land piece's value in relation to the attributes of the neighboring pieces. Landscape generator, generates the landscape corresponding to the output of the landscape experiment. Land value generator together with landscape generator constitutes the mapping module where design data is converted to computational data. Built-scape generator generates possible configurations of the functional relations embedded in the program definitions and corresponds to the generation module in the algorithm. Land value generator initiates landscape generator, and landscape generator sends feedback about the status of the generation process. Landscape generator also with the same command mechanism communicates to built-scape generator and initiates the generation.

Figure 7 also illustrates feedback loops present in the generative system. After assigning land values on land pieces, land-value generator communicates to the landscape generator enabling it to initiate the process of the experiment. This type of feedback is mainly controlled and selected in relation to the previously determined timetables by the designer or researchers. Responding to the requirements of the designated experiment, landscape generator commands built-scape generator to calculate possible configurations for the functional relations of the program.

After termination of the event happening on the land, building components decompose into their components to be transported to the future event location and to be transformed to configure a different design setup. Even though the event is finished, some of these building components are possibly needed for that piece of land for the future, so these land pieces with the fixed building parts on, get an attribute of "occupied". To manage this communication, built-scape generator sends feedback to landscape generator and prevents landscape generator and built-scape generator include these pieces for the next iteration.

CHAPTER 3

GENERATIVE DESIGN SYSTEMS

Recently, generative computational tools are increasingly influencing the design processes. Recent studies regarding the use of generative approaches in architectural design drives the architectural community to move a little further from the sole use of generative systems to come up with a set of alternatives and employing these systems as form-finding tools, to a more aware state of how to explore these computational tools in terms of their adaptability. In addition to the increasing interest in dynamic design problem definitions, the progresses in computational tools for design problems also in relation to the artificial intelligence have opened up a way for computers to contribute to the design process and interact actively throughout the decision-making action. However, the understanding of generative design systems still does not attach enough importance to provide sufficient tools to improve the dynamic design process. In order to implement adaptability more in design processes, this thesis intends to develop an algorithm that can adapt to the dynamic nature of a design problem.

Generative design systems are explored to define an overall framework for the research specifically on generative algorithms. Chu (2006) describes generative systems as open-ended mechanisms, which respond to the arrival of new principles introduced to the system and provide competence between the existing and the emergent structure in order to generate an anticipated result. So, this statement implies that the generative system developed is not the final model since it is an open-ended system by definition. The generative algorithm can be developed so that it suffices future transformations triggered by the arrival of new tasks. To be in accordance with the emergence of a new environment the generative algorithm

therefore must be able to accept changes. It is an adaptable generative system with no immutable final structure; it is changing and adjusting itself to the new conditions dictated by the environment.

The necessities of generative design systems and their utilization for computational purposes in architectural design are investigated. Recent attempts to make use of generative design systems are mostly for generating alternatives, form finding as a result of aesthetical concerns and using these systems as performance softwares like calculating energy consumption, acoustical properties. Herr and Kvan (2007) argue about the use of generative tools as stochastic search tools. In addition to the advantages of generative design systems as stochastic search tools, when using generative methods, the computer becomes a design generator besides being a drafting tool, visualization medium, data and performance analyst in its more conventional understanding (Shea, Aish, & Gourtovaia, 2005).

The generative system –developed as a structure for a generative algorithm- transforming its structure to be able to adapt to the transformation of the environment brings another aspect of utilizing these systems in design processes. McCormack and his group (2004) argue that generative design methods are used to bring novelty and diversity to the design problem from simple units by combining the adaptability and variety in life. When compared to the argument of using generative tools in design processes to enhance the complexity level of a final result, McCormack and the others' argument is more open to introducing adaptability of generative algorithms to dynamic design processes.

3.1. Some Selected Examples and Their Biological References

To form a background study for proposing an adaptable generative algorithm structure, generative structures are investigated. The proposed adaptable quality of a generative algorithm is studied under categories of artificial generative systems, autonomous iterative systems, and biological systems.

To talk about adaptation and fully understand the essentials of adaptability, it is necessary to look at the definitions of biological systems. The notion of adaptation

originates from biological systems. It provides adjustability to new conditions and reaction to abrupt changes in the environment (Peng & Gero, 2007). Also, Chu (2006) talks about biological systems in relation to the viruses and their ability to re-configure and re-organize their host organism's metabolism and to create a new structure with different properties so that it ensures new possibilities for further transformation into a future organism. So, new structures and organizations emerge as a result of these viral interventions. The virus does not adjust to a new environment, instead it transforms the host body, and therefore it survives. It is a good example to talk about a change in the environment and how a biological mechanism functions to survive. It is also possible to look at a human body for it is the ultimate result of emergent patterns, which are determined by local interactions among a large number of cells (Jacob, 2003).

Forming a framework for the discussion of potentials of generative design systems in terms of adaptability, evolutionary mechanisms are explored for their properties concerning the adaptation to the environment to stay alive.

3.2. Evolutionary Systems

In this section, evolutionary mechanisms are explored to initiate the discussion about adaptability, which recently has been a source of inspiration in architectural practice. The reason for this is that adaptability has its roots in nature, and evolutionary systems are the primary structures to investigate about adaptability present in nature.

To start, evolutionary systems and their specific characteristics concerning their potentials about adaptation to their environment are reviewed. Evolutionary mechanisms inspired by and based upon evolution in nature (Grobman et al., 2009), are characterized by a natural selection process applied to a number of individuals of a given population; where this natural selection connotes to the survival of the fittest (Eiben & Smith, 2003). The fitness of the population increases with the evolution of the system into a system with a higher value of fitness. The reason to explore evolutionary systems is not to talk about the selection processes of these evolving systems. Acknowledging the natural selection process of evolutionary

systems, the adaptability of the evolutionary system to survive in the environment and the struggle to endure until its fitness value cannot meet the required amount is the main interest investigated in this thesis. It is the transformation –evolution- of the system into a higher-level of organism to maintain its existence that is the essence of introducing evolutionary systems.

Evolutionary systems are adopted as sufficient computational tools to support the increasing level of complexity in design problems and to tackle with the abundant quantity of information acquired form the design input. Besides utilizing these systems to achieve complexity in architecture, which in some cases cannot be achieved by conventional design methods, evolutionary references are used to come up with form-generation tools challenging the understanding of an architectural object. Evolutionary systems are addressed as computational tools to support the growing quantity of information, the level of complexity. But in this thesis, the main reason to address these systems is the capabilities of these evolutionary systems to self-manage and to restructure their systems to correspond to the needs of their changing environment.

Also algorithms making use of evolutionary search tools are introduced in this section for they have potentials in terms of generating a large number of solutions meeting the requirements of the objective of the job (Grobman et al., 2009). This thesis acknowledges the use of evolutionary search algorithms as tools for generating a large number of solutions. But the main concern is how these evolutionary algorithms can be designed so that they are capable of surviving in any constantly changing environment.

Evolutionary systems are applicable in a wide range of fields. Examples can be sorted from composing interactive music by Felice, and his group (2002), and by Harley (1995) to the modelling of a material handling system by Seidel et al. (2007); and from urban simulations by Leach (2009) to basic free-form generations by Durmazoğlu and Çağdaş (2007) and Shea and her group (2005).

Evolutionary computing is based on the principles of evolutionary biology and it imitates the origins of biological systems. Evolution in biology refers to the survival

of the fittest, where in architecture it is employed as a creative process, which generates architectural form. Evolution in this matter is used as a process to transform the generated output to achieve a solution with higher level of fitness.

The question of evolution in this thesis does not necessitate that the resulting output is more satisfying than the previous solution in terms of evolution. After the design problem definition changes, it is possible that the solution will evolve into something less evolved. If a new solution is obtained from the new problem definition, it does not require a higher level of fitness to the environment. Evolution in the sense that it is an emergence for the generative system to survive in its changing environment is related to the notion of adaptability. If the generative system successfully adapts to the new problem definition, it entails that it is evolved in that sense.

I study these evolutionary systems to find correlations between evolution and adaptability. To elaborate more on the uses of evolutionary systems; genetic algorithms relating to artificial generative systems, Cellular Automata mostly relating to autonomous iterative systems, and organic computing together with swarm intelligence in relation to biological systems in nature are explored in the next section.

3.2.1. Genetic algorithms

Noteworthy examples for evolutionary systems in computing are genetic algorithms. First introduced by John Holland in 1960's, genetic algorithms are characterized by the transmissions of the characteristics from one generation to the next by genes, which highly involve the evolution of the solution according to the fitness criteria (Rocker, 2006). These algorithms are mainly searching tools to find the optimum design solution constrained and parameterized by numerous predefined inputs (Fasoulaki, 2007). Genetic algorithms perform previously assigned operations on a population of individuals with only those that fulfill the fitness criteria and therefore are able to survive. These types of algorithms are tools of nonlinear search and optimization performance. Genetic algorithms are applicable to the design problems that consist of several different phases and have discontinuous nature.
A notable example for genetic algorithm use in architecture for the purpose of

optimization is the genetic algorithm tool by Caldas and Norford (2002). This tool explores the use of genetic algorithms as part of a generative and goal-oriented design for the purpose of developing and evaluating principles concerning the environmental performance of a building. On top of these objectives for the utilization of generative algorithms –genetic or not- in a design process, the competence of such algorithms in restructuring their configurations to respond to the dynamic definitions of a design problem is fundamental.

Above all of the discussions that genetic algorithms are employed as computational tools to go through a large number of iterations in a considerably short time and find a suitable solution fulfilling almost all of the predefined requirements; acknowledging these arguments, this thesis investigates a unique application of generative design systems reflecting its evolutionary and adaptable properties defined by constraints and goals of the problem definition. The mechanism suggested emphasizes the flexibility of such mechanisms to make alternative worlds possible at once in addition to addressing the objective of finding a well-suited solution for the design problem.

3.2.2. Cellular Automata

One of the generative systems explored, which can be categorized under all of three categories mentioned above but is more related to the autonomous iterative systems, is cellular automata. In this thesis, cellular automaton is referred to as a model where contiguous or adjacent cells (units) change their states, their attributes or characteristics by means of applying simple and repetitive rules (Batty, 1997). Batty (1997) also describes cellular automaton as a mechanism where a state of a cell changes according to what is going on in the neighborhood of the cell –where “neighborhood” is defined as the primary adjacent cells. He states that the fundamental principle of cellular automaton is an “if-then” algorithm, which sometimes is tagged along with an “else” clause. The essence of cellular automata (CA) in terms of neighboring relations between the units –individuals- is the fundamental issue for the reference to study CA. The reference is strengthened with the automaton properties of CA in correspondence to the organization of the

algorithm, the interaction between the components, and the global behavior emerging from these interactions.

Cellular automaton is investigated for the potentiality of such generative systems in terms of adapting to changing definitions with respect to defined parameters assigned to them. In cellular automaton, the cell attributes change according to their neighboring units. Through redefining the relations between the units, every time a neighboring unit changes condition, the surrounding unit attributes adjust responding to this change. Acting on dynamic configurations, gliders and fixed patterns are formed and they do not change state even the environmental conditions change. Herr and Kvan (2007) argue that in relation to the cellular automata, a variety of cell behaviors can dynamically be assigned during the execution of CA, so that the transformation of the computational system during the generation process can be altered over time. This characteristic of cellular automata is interesting in terms of their potential of transforming their structure according to the dynamic demands of the design problem and adaptation to endure its life. The modular structure of the cellular automata facilitates this adaptation and brings manageability to the structure in terms of controllability.

The use of CA can be exemplified with different applications. Batty (1997) uses CA to simulate urban growth or decline whereas Herr and Kvan (2007) describe CA to be used in specific stages of design instead of using it all through the design process only to create complex forms.

Torrens (2000) talks about the decentralized feature cellular automata provide, the flexibility and their dynamic processes, which make cellular automata as an appropriate choice for urban simulation modelling. Thus, cellular automaton is a relevant system to be mentioned referring to their properties of processes. CA models also have been utilized in the survey of urban cases, traffic simulation, regional-scale urbanization, land-use dynamics, polycentricity, historical urbanization, and urban development (Torrens & O'Sullivan, 2001).

Besides their usability in the architectural field, cellular automata lack the representation of real-world phenomena because of their abstract characteristics

and their simplicity. They are not one-tool solution for urban simulation, so it needs more than one tool to compute the urban simulation when using CA. This necessitates introducing additional components to add functionality to basic CA framework (Torrens & O'Sullivan 2001). Also cellular automata do not always incorporate dynamism. If all the circumstances are met, they have the tendency to stop or create a continuous pattern without any change in the system.

To sum up, besides all the capabilities in terms of urban simulation, the inefficiency in terms of reality can be overcome with combining several approaches. Still, cellular automaton has its inadequacies to be designed as a fully adaptable system to be used at the very first phases of design processes, so there has to be another tool to handle the dynamicity of the problem developed as a whole.

3.2.3. Swarm intelligence in relation to organic computing

In relation to the biological systems, the most related issue to the adaptability is the organic computing which studies computing systems that consist of autonomous components and show forms of collective behavior in biological systems. And closely related to the organic computing, the subject of swarm intelligence is investigated.

In this section, the decentralized feature of organic computing is investigated concerning to have a swarm behavior so that the system as a whole can be aware of its environment and adjust accordingly. Even though the notion of swarm intelligence is not directly applied in the adaptable generative system proposed in this thesis, I find it interesting to provide background knowledge about how these systems work in terms of communications between the components and managements of their structures. Therefore, the issue of swarm intelligence is not extensively investigated in this section, but the most important features of swarm intelligence such as their decentralized structures and communicative components are explored to supplement the discussion about the adaptability of generative systems.

According to Merkle et al. (2008), the basic issue of organic computing is that these

systems should have decentralized control and as a result should be adaptive to changing requirements of their user. This adaptiveness, in turn, maintains the lifetime of the system. It is a crucial driver for survival. Also Jacob (2003) talks about swarms in relation to organic computing, as highly distributed developmental processes, which does not have any centralized control agency. The importance of decentralized approach is that it has a free flow structure and an emergent behavior when compared to centralized approach (Seidel et al., 2007).

A bottom-up emergent system of a swarm is advantageous in terms of embedding a system of individual intelligent components to interact with one another. The interesting course of swarm intelligence is that it makes use of agents interacting with each other, so that the whole system is composed of individual behavior. Decentralized mechanisms in the sense that there is no control mechanism to manage the operations running in the system is fundamental for the generative system because I consider that it is more manageable when individuals are the ones which decide to adjust themselves accordingly, when compared to a case where the system tries to adapt itself to new conditions as a whole.

3.3. Self-Management of Generative Design Systems

Self-management, in relation to the swarm intelligence is considered remarkable for exploring the structure of generative design systems. Following the issue of decentralized systems, self-management refers to the feature of mechanisms, which are capable of hierarchically orchestrating their individuals' behaviors and as a result having some degree of autonomy. Self-management, associated with self-organization, is essential for an adaptable system. To be able to react to the changes in the environment, it is viable to react as parts but in a synchronous way. The reaction is decomposed into its smaller parts and every element of the mechanism responds individually to do its part in order to fulfill the newly defined objectives. Self-managing properties are assigned to the developed generative algorithm structure in the form of autonomous agents. These agents in collaboration with the generators, implemented in the algorithm as loops, groups and modules, determine the algorithm activities. Individual behaviors compose a global behavior, so that self-management is achieved in the system.

Self-management is a modification in the environment, when two individuals interact; one of the individuals modifies the environment and the other responds to this newly defined environment; therefore, individual behaviour modifies the environment, which in turn modifies the behaviour of other individuals (Eleni, Turner, & Thum, 2002). Self-management, also including self-organization, is the main driver for employing "self-x" properties in design methodologies.[2] According to Gürer and Çağdaş (2006b), design methodologies inspire from natural systems mostly for their hierarchical organizations established between the levels of their systems. This hierarchical organization is result of a self-organization between the components, where self-organization is a spontaneous action occurring in structures composed of multi-levels to provide a collective behavior of the whole system (Gürer & Çağdaş, 2006b).

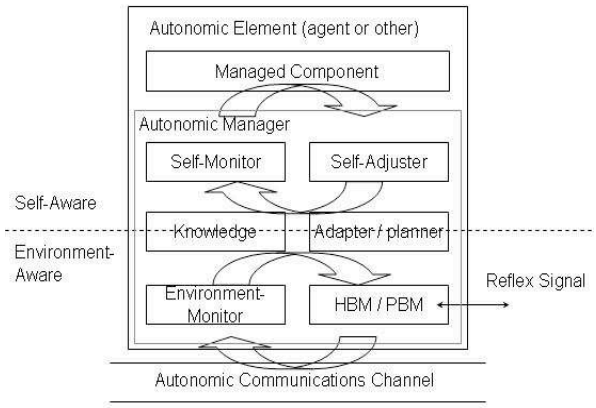


Figure 8 Autonomic Element, Managed Component and Autonomic Manager (Sterritt & Hinchey, 2005)

An example of a self-managing system in Figure 8 represents an Autonomic Element (AE), which includes a Managed Component (MC), and an Autonomic

[2] Sterritt & Hinchey (2005) talk about "self-x" properties to discuss self-managing systems. These self-x properties are self-configuring, self-optimizing, along with self-awareness, self-monitoring, and self-adjusting.

Manager (AM). Control loops – with feedback and feed forward- with sensors (self-monitoring) and effectors (self-adjusting) together with system knowledge and adapting strategies allow the autonomic element to be self-aware and to self-manage (Sterritt & Hinchey, 2005). A similar scheme makes environment awareness possible – meaning allowing self-managing if necessary, but without the immediate control to change the environment. This is realized by communicating with other autonomic managers that have the relevant influence. In this scheme, every component in a system, and every system within *systems of systems* are self-managing by managing communications between autonomic managers (Sterritt & Hinchey, 2005).

This scheme illustrating an autonomic element is exemplified for it represents how an agent can function as a mechanism. By means of implementing sub-mechanisms to interact with the environment and obtain the information essential for properly functioning, the element is instructed with tasks assigned to them where and when a change occurs in the environment. This ability to interact with the environment and being environment-aware is crucial to talk about adaptability of a generative system. In order for the agent to detect the change in its environment, the agent must have sub-mechanisms to analyze and evaluate what's happening in the environment. This environment-monitoring unit will be examined in detail in section about how agents function in chapter 5.

In addition to the environment-monitoring sub-mechanism, with a self-controlling unit –self-monitoring and self-adjusting, the element communicates with its internal elements and detects if the mechanism runs accurately. This self-controlling unit provides self-awareness and therefore self-managing characteristics in the agent mechanism that the agent knows how to achieve internal stability to be part of a higher-level global society. This self-controlling unit Sterritt and Hinchey (2005) present is applicable to serve as a mechanism to be practiced in the developed generative algorithm structure. Based on this figure, sensor agents make use of these sub-mechanisms to be aware of the environment as well as their internal operation. This scheme pertaining to an autonomic element –in this case an agent- is also interesting for the discussion of autonomicity of the algorithm structure. Agents, being self-aware and capable of self-adjusting according to the interactions

within the system and to the changing parameters of the environment, in collaboration with the generator elements constitute the essence of the adaptable generative algorithm. It is therefore crucial to talk about self-management in close relation to autonomicity.

Self-management can also be discussed through examples of a city. Leach (2009) talks about the city describing it as a self-managing device, as a dynamic, and an adaptive system, which is based on interactions with neighbors, informational feedback loops and indirect control. Also, he states that the city's form emerges from the interactions between its constituent parts and its specific context. Another aspect for the self-management examples of cities can be put in a way that the cities are bottom-up processes, which are driven by interactions between interdependent parts from which spatial and organizational qualities emerge (Schork, Burrow & Minifie, 2009).

Thus, self-management, in relation to self-organization and autonomicity, is a general but important feature to be considered in structuring of the adaptable generative model. I think self-management, when there is no external interference – an outer control mechanism such as the designer, is the fundamental feature that an autonomous adaptable system without it can collapse with conflicting objectives of individuals composing the system. I try to develop an adaptable generative system, which can adjust its component configurations to the changing problem definition by self-updating its objectives. Self-management along with autonomicity ensures that it is possible to talk about adaptability of the generative system in question. Individuals composing this generative system perform their assigned tasks and interact with each other. The resultant global behavior is the indication of a system self-managed satisfying the expected actions.

3.4. Emergent Behaviors of Generative Design Systems

Emergence in relation to the evolutionary computing is an essential issue to be discussed in this thesis in association with the changing problem definitions and the transforming environment. Design problems are no longer considered to be linear processes, because the input of the design problem definition is not static.

Therefore, the issue of emergence regarding the dynamic design processes which continuously changing and demanding constant transformations to sustain their dynamicity has become increasingly important.

Emergence, where it is an act of appearing, is investigated through discussing evolution of the generative systems. In case of a threat endangering the maintenance of the adaptable generative system, evolution of the system and therefore emergence of a fitter system is necessary. In this case, the feedback is the changing environment conditions depending on the changing design problem definition. In the adaptable generative algorithm designed, emergent behavior is expected as a result of the lower level interactions between individuals and with the arrival of a new design problem definition. New communications and interactions between the individuals –such as effector agent communicating to the mapping module to rerun the loop, emerge with the arrival of a new job. Local interactions between the individuals and agents result in a higher-level global behavior, therefore a hierarchical and more adaptable system emerges.

According to Johnson (2001), emergence in a behavioural aspect is connoted as the shift from low-level rules to higher-level complexity resulting from complex interactions between local components. So, emergence is what happens when a system of relatively simple elements self-organize to form more intelligent, more adaptive higher-level behavior, thus an emergent behavior results from collective behaviors of a number of individuals operating in an environment (Chien, 2007). When applied to architecture, emergence is generally used as a concept for unpredicted and unconventional forms.

Emergence is mostly utilized in architectural design to explore the borders of human conventional understanding of an architectural object since it is unpredictable. But, there also exists a potential danger that an emergent system might exhibit consequences, which are unwanted and have not been foreseen when the system was designed (Merkle, Middendorf & Scheidler, 2008). So it is important to be aware of what the designed system is capable of in terms of emerging properties. When talking about self-organized systems, it is possible that these systems can show emergent effects, which are not suitable for the algorithm to keep running. So

Merkle et al. (2008) put forward the notion of controlled emergence, where some agents detect unwanted emergent behavior in the system. In this system they propose a set of observers collecting information about the system and based on this information the controllers send control information to the components to influence their behavior. But these control mechanisms are against to the very essence of self-managing systems such as decentralized and self-organizing features. To overcome this clash of ideas, a different kind of component can be introduced preventing unwanted emergence. These components can still do normal work in the system but not like the rest of the components. They detect but do not send information to any superior unit.[3]

3.5. Examples of Generative Design Systems

In this section, I investigate examples of generative design systems. These examples are not directly related to the issue of adaptability but can be explored as computational tools being potentially able to adjust their structure to adapt to the changing conditions. The first example is the Generative Design Agents (GDA) model proposed by Gu and Maher (2005) in detail for its dynamic modelling properties. This model introduces dynamics and autonomy to the designs of 3D virtual worlds. This 3D virtual world designed with GDA model does not have a static infrastructure like the built environments; it is designed for a particular "moment", and reflects its inhabitants' interests of that "moment" (Gu & Maher 2005). The "moment" –this thesis refers to the "moment" as the "baked state" unlike the built environment- these are open to transformation over time. GDA constructs an anticipated world by setting up design goals to reduce the clashes between the world and its inhabitants. It works as a system constantly updating its structure according to the changes occurring in the system to satisfy the design goals.

Another example, Diffusion-Limited Aggregation (DLA) proposed by Narahara (2009) is surveyed. DLA is a generative system where the system can supply solid solutions for constant and endless changes. The system is designed so that it

[3] This thesis does not delve into details of how to implement these components, which are capable of preventing unwanted emergence. It is suggested as a further study.

maintains its balance as a whole, composed of individual bits.

I also investigate an emergent design software toolbox, ALife designed by Testa et al. (2001). After defining the goals of the design investigation such as how spatial elements may be configured, the user/designer specifies the initial conditions such as size, scale, and conditions of the site. The next step is to identify the elements and the interactions among them. The user/designer introduces the relationships between the individuals and how elements influence each other under which conditions. This thesis also makes use of such a process ALife tool adopts in the algorithm structure.

Another generative design system EifForm, designed by Shea et al. (2005) is a generative structural system, which is associated with Generative Components to generate structural members with an optimization process through complex design constraints and generation of multiple alternative designs from a single starting point (Shea, Aish & Gourtovaia, 2005).[4]

After investigating about generative design systems, to elaborate more on the issue of adaptability of computational tools, I also investigate the use of generative algorithms in design processes in a general context. The following chapter exploring potentials of generative algorithms in terms of adaptability covers the utilization of these systems in the architectural field. It does not only focus on dynamic design processes for the sake of exploring how a generative algorithm can gain adaptable behavior. It covers the utilization of generative algorithms in design processes in a general manner.

[4] Generative Components is a graph-based associative geometry modeling system, which combines geometric modeling and programming. It is a parametric CAD software developed by Bentley Systems.

CHAPTER 4

GENERATIVE ALGORITHMS IN DESIGN PROCESSES

The use of generative algorithms as mechanisms to search for an anticipated universe is vastly encouraged so far. The anticipated universe as Chu (2006) puts it, can be thought of as an emergent occurrence, and a possible answer for the question, which embodies a number of mutual relations at different scales. What is more to this statement can be the addition of an adaptable potential to the search for alternative worlds that can meet the requirements of the changing definition of the design problem over time.

To cover what the potentials of generative algorithms are in terms of adaptability different applications of such systems in design processes are investigated. For the reason that there are no specific applications of generative algorithms in dynamic design processes, I looked at examples from different disciplines to understand and develop my own understanding of an adaptable generative algorithm.

4.1. Potentials of Generative Algorithms

This thesis tries to answer a main research question:

- How can we make use of generative algorithms as the means to sustain dynamic design processes?

For the investigation of the answer to this question, this thesis is investigating different applications of generative algorithms in design processes. The answer to this question lies at the very heart of the issue of adaptability of generative

algorithms. Fundamentally, the concept of adaptability is closely linked to the struggle to survive under the changing conditions of the environment. Here, the environment refers to the overall structure, which the definitions of dynamic design problems govern. As discussed before, design process is a nonlinear sequence of steps and feedback loops. In this framework, analysis of the problem determines the problem definition. In dynamic design processes, since the environment is continually changing, data obtained from the environment are also changing. This brings the necessity of updating the analysis stage, therefore the problem definition. The adaptability and the struggle to survive in nature correspond to the ability to update the design system and as a result the design system maintains its functions to generate solutions.

Dynamic design problem in its definition has the notion of dynamicity since the architectural object in question is constantly changing with transformations of the design task. Therefore, the architectural object has to survive while satisfying transforming needs of the design problem. The generative system is designed as a mechanism of feedback communicating to the designer. This is where the adaptability comes into play. The use of generative algorithms in architecture is gaining popularity for years now. Mostly, generative algorithms are utilized to generate as many alternatives as possible to provide the designer a variety of design solutions to choose from. Consequently, the designer decides on a solution from this set of alternatives or if embedded in the algorithm, a search tool extracts the solutions from that set according to the designer's previously established constraints. These computational tools are more than the means to assist through design process and supplement visual representation techniques. What is more to this utilization of generative algorithms as computational tools to facilitate form-finding processes and produce populations of solutions is the potential of these generative algorithms in terms of supporting dynamic design processes. This potential also brings along the notion of adaptability.

In light of these explanations and through studying concepts related to adaptability, this chapter tries to inquire about the adaptability of generative algorithms to sustain dynamic design processes and investigates their potentials to answer the research question mentioned above.

According to Herr and Kvan (2007), we need generative design tools as management and expression mechanisms to increase the complexity of the components –factors- that determine design processes and the possibility of exploring and representing a series of alternatives using generative algorithms is often seen as an advantage of using generative algorithms.

Caldas (2001) states that regardless of the simple structure generative algorithms have, they are efficient in solving complex design problems, where other conventional optimization methods lack sufficiency to result expectedly. In addition to the efficiency of generative systems to solve complex problems, it is also necessary to use dynamic structures for a computational tool (Blackwell, 2008) coupled with the application of dynamic programming method to a computational tool (Caldas, 2001). So, the dynamic characteristics of generative algorithms provide a strong base for the affordance of using such algorithms in design processes. In complex design problems, the use of generative algorithms as computational tools can meet the demand for the search of a global and a local optimum within reasonable time and computational costs (Renner & Ekárt, 2003). Barczik and Kurth (2007) also talk about using algorithms in architectural design as systems, which can provide easier adjustment of a design to a given situation's specifics, especially once those start to change dynamically (Barczik & Kurth, 2007). But is it all generative algorithm can offer in design processes besides being a stochastic search mechanism? To explore the answer to this question, this thesis looks into applications of generative algorithms in design processes –whether these processes being dynamic or not.

4.2. Different Applications of Generative Algorithms in Design Processes

Besides using computational tools to generate free-form shapes, generative algorithms are widely applied to analyze and evaluate design problems.

An example outside the scope of architectural concerns is the application of generative algorithms to produce music tunes. These are mainly GenOrchestra developed by De Felice and the others (2002), which is a software producing music tunes with a fitness function and CHAOTICS developed by Harley (1995), which

algorithmically composes music. Although these appear to be irrelevant in the field of architecture, the basic goal behind employing generative algorithms to generate music can be made use of in dynamic design processes. To generate music by defining rules –parameters and constraints- is related to this thesis' secondary argument to define an algorithm to generate relationships. And these algorithms can be explored and be benefited from for the purpose of describing the algorithms.

4.3. Adaptability in Relation to Dynamicity and Autonomicity of Generative Algorithms

To sum up what I have talked about generative algorithms, this section discusses adaptability in relation to dynamicity and autonomicity of generative algorithms.

The generative algorithm, which is going to be developed and explained in detail in chapter 5, should be capable of detecting changes of the environment as well as the recent transformations of its own structure. The ability to detect these changes can be emphasized by inquiring about their adaptable properties.

Simon (1983) talks about the adaptability of a mechanism and states that pertaining to the efficacy of a CAD tool, it is important to have a mechanism capable of bringing changes to the system, and this mechanism needs to be adaptive. Gero and Peng (2007) propose that these changes in the structure enable the system to tackle the same task or tasks drawn from the same population more successfully the next time. So Gero (2003) proposes a design tool that adapts, based on its experience of its use is claimed to be effective. Gero and Peng (2007) describe an approach that enables a CAD tool to be built on an adaptive paradigm, so that a design tool can learn conceptual knowledge as it is being used, and as a result it adapts its behaviors to the changing environment.[5] Also, Akkawi et al. (2007) refers to adaptability of a system as the ability from which the programmers as well as the system can easily change and evolve the system to meet future requirements or adjust to runtime environment fluctuations.

[5] They put forward the notion "situatedness" which comprises the context, the observer's experiences and interactions between the context and the experiences. In relation to this statement, adaptation enables the design tool to cope with situatedness in a dynamic design process (Gero & Peng, 2007).

Dynamicity associated with adaptability is the most crucial subject to talk about in this section. This thesis investigates the use of adaptable generative algorithms in dynamic design processes. It acknowledges that design processes are dynamic, but is it just enough to state that a design problem definition changes? While talking about the adaptability of a generative structure, in my opinion it is important to also talk about its dynamic properties. It is because since I develop an adaptable generative algorithm, it means that the structure of the algorithm is also changing to compensate the changes taking place.

The main topic of recent architectural discussions about design problems is addressed as the dynamicity of such design problems. Dynamic design process by description indicates that the architectural object designed at the end is never the final solution to the problem. Since dynamic design problems entail ever-changing solutions and form close relationships with the notion of time and transformation, it is indisputable that these design problems necessitate constantly changing dynamic processes. Gilat (2005) states that dynamic adaptation helps to support continuous strength and productivity of an infrastructure. According to Caldas (2001) the use of dynamic constraints improves the degree of adaptiveness and diversity in new solutions. Also, an algorithm is a dynamic rational pattern more than a software program that is merely an instruction set (Chu 2006).

Also, Harley (1995) stated that structure of a generative system, depending on its adaptability features, can be constructed in such a way that the basic architecture of the generative model can be modified or adjusted by the user/the designer during the generative process to perform necessary objectives. Here, Harley's description of adaptability asserts that the adaptability of the algorithm is dependent on the user, who decides when to adjust the system. In a different point of view, the algorithm in question, is structured so that it can understand when to adapt and how to adapt according to the changing environmental stimuli and varying objective definitions and react correspondingly. This brings out the notion of autonomicity of a generative algorithm.
To understand the notion of adaptability also in relation to the biological systems, a new term "autonomicity" is introduced. Sterritt and Hinchey (2005) define autonomicity as the self-management of generative systems whereas autonomy is

defined as the self-governance. Self-management and self-governance appear to indicate same self-properties. In my opinion, self-management means that a system is capable of hierarchically orchestrate itself to maintain a stable balance between its individuals. So, if a change occurs in the environment, the algorithm is capable of sustaining its stability as a whole by redefining its individual relations. Individuals adapt to changing conditions, they constitute a higher level of hierarchy and this global behavior makes it possible to manage to stay alive.

On the other hand, self-governance is most likely related to the decentralized feature instead of a centralized approach to be used as a control mechanism. In self-governing systems, autonomous individuals have their own aims and goals according to the user defined objectives. Each agent in the system is assigned a task to accomplish and each knows what to do when an error occurs. There is no central control mechanism detecting if the system is running sound or if a new job definition arrives. Agents are responsible for specific tasks and they take action as a whole acting as if they constitute a control mechanism. Self-governance and self-management are issues that are not so different from each other. Self-governance embodies some of self-management; where self-management also includes the notion of self-governance. These two closely related subjects about the autonomicity and autonomy of generative algorithms, in relation to the adaptable and dynamic structure of the algorithm developed, prepare a ground for defining and developing the algorithm supporting all of these notions.

In the following chapter, I present an adaptable generative algorithm, which I developed in light of all the statements in previous chapters so far. The structure of the algorithm, implementation of agents in this structure and the interdependencies between the generative elements and these agents are presented in detail. Also I developed a sequential flow structure for the adaptable generative algorithm to be translated into a desired programming language with further studies.

CHAPTER 5

DEVELOPING A STRUCTURE FOR AN ADAPTABLE GENERATIVE ALGORITHM TO SUSTAIN THE DYNAMICITY OF THE DESIGN PROCESSES

Previous chapters have explored generative design systems and generative algorithms. I have discussed how to achieve a design tool capable of self-organizing its structure to fit in a new environment and explored the adaptability of generative algorithms in order to provide the capability of adjusting to changing conditions. Referencing to these examples from distinct disciplines and using them as guides to develop an algorithm, this thesis puts forward a definition of a generative algorithm and handles generative algorithms in a new dimension. This new definition emphasizes the use of generative algorithms as computational tools, which are capable of maintaining the algorithm structure's adaptability as well the dynamicity of the design problem definitions. These tools are used where design problem definitions are not static but open to serve as mechanisms to more than one "baked moment"[6] of the design solution. Here, baked moment refers to a snapshot of a continuing generation process and this snapshot is regarded as the end product valid only for that specific run. This means that the baked moment is not the final solution, but it can be one of the possible stages during the evolution of the end product.

Many of the design problems are dynamic in the sense that the most suitable design solution changes over time. An algorithm, therefore, has to both find possible solutions and afterwards track the changing conditions. Blackwell (2008) talks about

[6] "Bake" is a tool implemented in Grasshopper software in Rhino. This tool turns all selected geometry created with Grasshopper into actual Rhino objects. This way it is made possible to operate these geometries without Grasshopper interface.

the examples of new job arrivals in scheduling, changing expected profits in portfolio optimization, and fluctuating demand. He proposes that we should make use of an algorithm, which can detect the change when a new task is given and it has to be integrated in the set up of that algorithm.

To satisfy the need to sustain dynamic design processes, an algorithm that can adapt to changing external tasks, where “external” refers to the environment of the system is proposed. To make the focus of this thesis clearer, in relation to how the algorithm is developed, the process of adaptation including the dynamic design process is explained in detail.

Developing a problem definition for a specific design for that “moment” of time and need, and introducing the data analyzed as a dynamic input to the generative algorithm, the process of defining relations is initiated in the system. According to the user-defined parameters and constraints, as in all of the generative processes, “a solution space” comprising possible design solutions meeting the fitness criteria is generated. The designer predefines these fitness criteria concerning the parameters and constraints of that specific design problem definition.

Fitness criteria defined and introduced as dynamic constraints in the generative system refer to the changing specifications of the design problem definition that a proper outcome should satisfy. For one instant of a problem definition, a technology laboratory should be located within a specific distance to the offices. This distance determines a range of distances within which a generated programmatic node for technology lab can be located. Outcomes satisfying this fitness value are selected. When the definitions of the functional relations change, distances between programmatic nodes also change. Therefore, the fitness value determining the satisfactory solutions also changes. This way, dynamic constraints and fitness criteria are introduced in the generative design system.

If a search tool is embedded in the generative algorithm structure, within this tool search phase starts to find the most sufficient design solution satisfying these fitness criteria. The algorithm is terminated with the selection of an outcome. Until now, it is the same process as in a general generative design process –using a

generative algorithm and a search tool to generate as many alternatives as possible meeting the requirements of the design problem.

The point where this thesis offers a new dimension comes up after the phase of selecting and baking the fittest solution among a population of generated alternatives. After the problem is defined and a solution satisfying all the requirements of the problem including the aesthetical aspect of that particular architectural object is selected, what happens if the design problem needs to be redefined? I aim to develop an adaptable generative algorithm capable of handling the redefinition of a design problem. So, the design problem definition is changed to meet another set of requirements dictated by external and internal forces. One of these forces can mainly be designated as the changing site properties and therefore analyses used as input in the algorithm. Functional relations between components of the design in mind also constitute a part of these forces. These relations are necessary to be considered to have an architectural object at the end. The user needs, which are set to constantly satisfy the changing nature of an inhabitant, also constitute a part of these external forces.

To compensate the change and to be able to modify the design according to changing parameters and constraints, the generative process needs to be redefined in relation to the altered conditions. The need to utilize an adaptable generative algorithm comes into action at this point. As a result of the individual behavior and their interactions, the algorithm is made up of connections and networks. Generators used in the first baked state of the design solution are decomposed into their smaller parts and rearranged in accordance with the new job definition. For example, a functional relation concerning the proximity of a circulation shaft to the entrances of a building is redefined in terms of the building codes and places of the shaft or the entrances need to be reconfigured to satisfy the requirements. The parts responsible for the generation of the configuration between the shaft and the entrances are transformed to generate configurations according to the new objectives. This process of defining, detecting, and redefining constitute the very basics of an adaptable generative algorithm.

In the generative algorithm developed in this thesis, in relation to organic computing,

it is essential to make use of autonomous components capable of creating an emergent behavior without introducing a global plan and without control information from a higher-level authority. This capability of self-management without a central control mechanism enables the algorithm to adjust itself to the changing inputs of the design problem. According to Merkle et al. (2008) it is essential to design an autonomous system with a decentralized control mechanism.

Sönmez and Erdem (2009) also state that it is fundamental to replace a centralized mechanism with an open and decentralized network. So, to successfully have a resultant global behavior, the system has to be decomposed into its components. Each one of these components has individual behavior, which in the end result in one global behavior of the generative system. After the task definition, the objective for the use of algorithm is determined. The desired behavior of the "universe" is decomposed into component behavior schemes and individual behaviors are elaborately analyzed and decomposed into their finer pieces of interactions with the environment and the other individuals. So it is basically setting up the individual rules that lead to the desired collective behavior (Trianni, Nolfi & Dorigo, 2008). Individual –local- decisions combined with the other individual decisions lead to a larger, more global effect on the design outcome (Testa et al., 2001). The system should be composed of interdependent entities, which are in constant interaction and dialogue with each other and their environment (Schork, Burrow & Minifie, 2009). A bottom-up collective intelligence, which is more sophisticated than the behavior of its parts can be obtained (Leach, 2009). These statements constitute the very essence of the argument in this thesis.

The fundamental step towards achieving an adaptable generative algorithm is to deconstruct the desired global behaviour into simpler individual behaviours and define interactions among the components. Even a small change in an individual behavior results in a drastic change in global behavior. The non-linear and indirect relation between the components and the interaction between the components and the universe are also explicitly analyzed and described particularly.

It does not mean that the system should be centralized because the control program manages to keep the system running appropriately. With the interactions between

the individuals and the resultant global behavior can be the “control mechanism” to keep the system running properly. Self-monitoring attributes are different than self-management in terms of one being the mechanism to detect changes of its environment along with the recent transformations of its own structure and the latter being the mechanism to carry out operations to adjust the system. In addition to self-managing components, by means of self-monitoring attributes integrated in the component structure, no control mechanism is needed to detect if the system is operating without any error. All of these statements about decentralized mechanisms of adaptable generative systems relate to the adaptability of the generative algorithm in terms of manageability and realization of the adaptability concept in this thesis. To achieve an effective adaptive behavior, it is fundamental to attain a global adaptive behavior composed of interactions between local decision systems.

Exploring the interaction between the components –individuals- of a swarm is helpful to develop an understanding of the relation between the components of the algorithm and how they communicate.[7] This type of communication between the individuals of the universe allows the system to adapt to different needs. How is this adaptation going to be executed? How does the component react to the change in the system? How is the transformation managed? This type of communication between the individuals tells about the relation between the roles. But it does not tell much about the way the system adapts. The synchronization between the components and the resultant global behavior is crucial, but more importantly the transforming structure of the algorithm needs to be clearly defined.

5.1. A Proposed Structure of the Generative Algorithm

In light of the arguments in the previous section, an adaptable generative algorithm structure is developed. Referencing to the upcoming examples developed in order to solve one piece of the question of adaptability of generative systems, the algorithm structure is developed. These examples deal with a small part of the

[7] As discussed in section 3.3 about the self-management of generative design systems, this means that a change in the role of a component of the structure is delivered to the other related components and this triggers the adaptation of the other components to meet the desired function.

overall problem, which this thesis is concerned about. Therefore instead of the processes as they are employed in these examples, approaches to handle the design problem are investigated.

Building up the algorithm structure brings along defining specific design needs to a series of stages and modules. Therefore, the algorithm structure comprises a number of levels (Figure 9). First stage of the structure is “introducing the input” obtained from the analysis stage of the design process. The algorithm initiates with evaluating the attributes of the currently defined environment such as the values of the land pieces according to their proximities to nearby attraction elements –water, road, and land.

In the following “generate” stage, the algorithm computes the suitable positions and configurations for the programmatic nodes according to the defined functional relations and needs by the designer/user. These constraints are determined by the user beforehand and constitute the input set of the algorithm. Every time the problem definition changes, the designer redefines and reintroduces these constraints as the design problem input. The algorithm basically finds out all the possible configurations and results in a set of alternatives. Each one of these alternatives corresponds to a different plausible solution for the algorithm’s future transformations. In order to select the most satisfactory output, the user defines the fitness criteria determining the selection criteria –concerning the graduation project, which in this case the accessibility, proximity to a water piece, to be on a suitable land piece in terms of dimensions.

In the next “search and bake” stage, the algorithm freezes the selected alternative, and until a new job arrives the algorithm is idle meaning that until the generative procedure is called back, the algorithm stops running. New job means that a new condition has arrived to the environment and the algorithm should re-initialize the run. New job definition particular to the graduation project refers to a new event and this necessitates new functional configurations between programmatic nodes to be re-defined and re-evaluated. Activation module searches and evaluates if the current environment is suitable for the coming event, for the future requirements. If yes, it stops until a new job definition arrives. If the current configuration does not

meet the new job's requirements and is obsolete, it needs to be re-defined. In such a case, the prevailing configuration is decomposed into its nodal elements and the relations between them are re-defined by the designer. If needed, additional programmatic elements are put to the definition of the input set. As a next step, the algorithm updates the entire system according to the new definition of the functional relations. It then re-runs, and re-generates the possible configurations. And when the output set is finite, it bakes and is idle until the procedure is called again with the arrival of a new job definition.

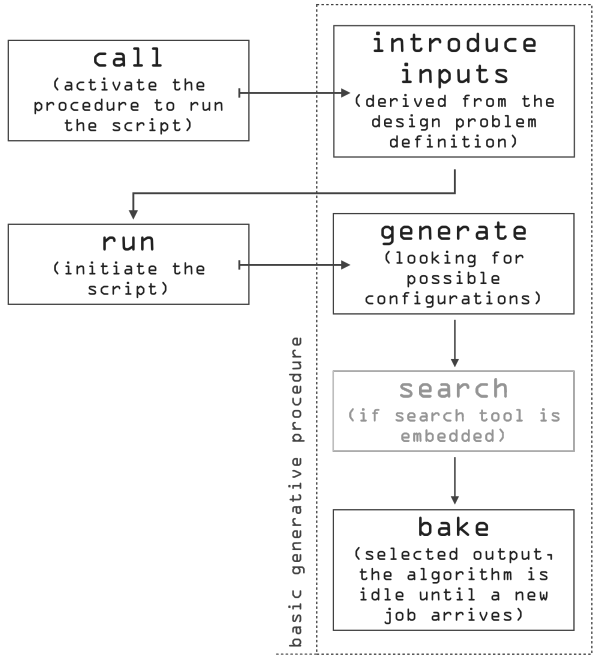


Figure 9 Basic generative procedure

Figure 9 represents the basic generative procedure of the algorithm processes and the sequence of procedures starting with the "introduce inputs" procedure. The algorithm is initiated with an outer command "run". After "run" is executed, "generate" command takes action and runs until a population of solutions are

generated.[8] To look for possible solutions and select the fittest one, “search” and “bake” commands run consecutively. After selecting the best-fit solution, the algorithm exits the loop and becomes “idle” until it is called back for a new task.

The outline of the algorithm is summarized as follows:

- Introduce inputs > Run > Generate > Search > Bake
    - New job arrives > Search > Evaluate
- Call > Introduce inputs > Run > Generate > Search > Bake

To illustrate how call mechanisms and generative codes are integrated, Figure 10 indicates the relations between these mechanisms. After the basic generative parts accomplish their tasks and the designer obtains a satisfying output, the algorithm is idle. New task activates search tool in the activation module to look through generations which the algorithm generated in t=1[9]. Then, these generations are evaluated in terms of fitness criteria. If the current algorithm structure is competent with the new problem definition, the algorithm exits the loop. If not, the generative parts are called back; the structure of these parts is adjusted with changing the series of loops in the modules and the algorithm reruns.

[8] Number of solutions composing a population is dependent on the constraints defined by the designer.
[9] As I discussed in chapter 2, dynamicity brings along the notion of time. t=1 refers to the first generation loop. It is incremented with every generative loop (t=2, t=3, t=n).

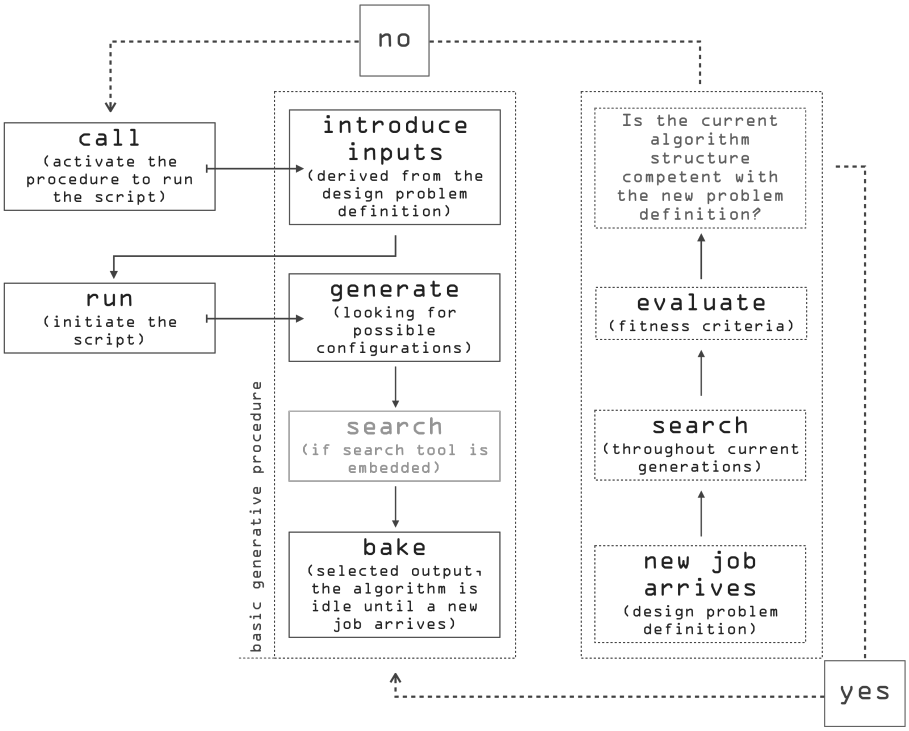


Figure 10 Algorithm process with the arrival of a new job definition

To control the generative algorithm's adaptability, it is fundamental to have a modular system composed of smaller parts. It provides manageability of smaller parts and having a global adaptable behavior as a result of individual adjustments. That being the case, the generative algorithm comprises a number of modules (Figure 11). These can be examined in two categories: generative parts and the call-procedure-back mechanisms. Generative parts are composed of a mapping module, a generation module, a search module and a bake module. Call-procedure-back mechanism is the activation module when a new job definition is introduced to the generative system.

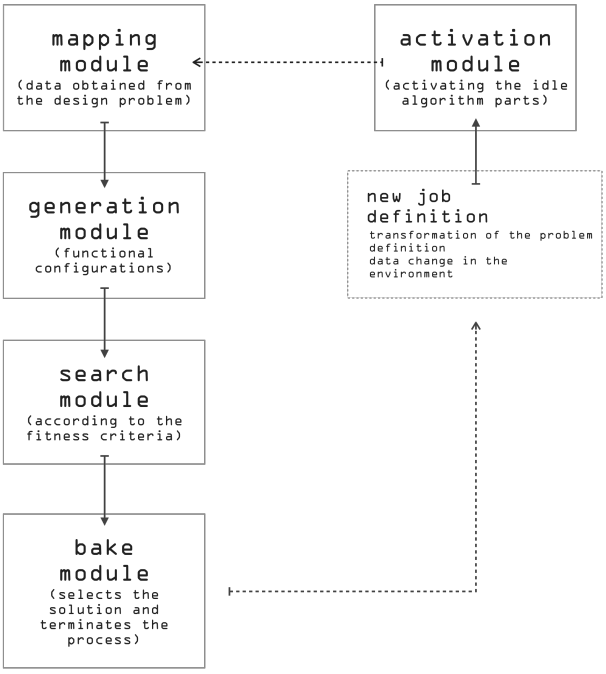


Figure 11 Modular system of the algorithm

This scheme composed of algorithm parts, constitute the main structure of the algorithm. Mapping module mainly converts data obtained from the design problem definition to parameters and constraints to be executed in the generation module. Also data about the site input and functional relations are stored in this module. After mapping module finishes its task, generation module according to the site input and functional relations between programmatic nodes, iterates a number of solutions satisfying the constraints. After a final output set is achieved, if there is a search module embedded in the system, it looks for the most satisfying outcome and selects it. If there is no mechanism to evaluate the output set, then, the designer is in charge of selecting a desired design solution. Bake module runs to freeze the selected solution and finalize the generative procedure. From mapping module to bake module, it is basically a common generative algorithm structure, where the data is processed into a design solution. When a new job is defined by the user/designer, activation module calls the generative procedure back and the

algorithm reruns starting from the mapping module and completes one generative loop.

Along with generative parts, activation module serves as a call-procedure-back mechanism and when a new task is defined, it activates the idle generative parts of the algorithm.

After an overall view of the algorithm stages, examples are presented to make the idea of levels and transformation processes clearer. The first project called BioSonic Project, which is an interactive growth system developed by Bisig (2005) is exemplified for the process of the reactions and interactions between generative system components. The scheme for the processes of transformations is mapped as follows:

Table 1 Bisig's (2005) BioSonic project, processes of reactions

chem1 → null	:	decay
chem1 ← null	:	production
chem1 → chem3	:	transformation
chem1 + chem2 → chem3	:	polymerization
chem1 + chem2 ← chem3	:	cleavage

The BioSonic Project deals with the employment of chemicals in order to achieve an evolutionary generative system. Chemicals as depicted in Table 1 go under processes of reactions to form new components to be implemented in the generative structure. In this way, there emerges a new structure of components with the introduction of mutated chemicals in the system.

Processes of reactions Bisig (2005) adopted in BioSonic, serve as a model for structuring the relations between the algorithm parts. So I adapt this table to come

up with my own processes of transformation (Table 2). To make it easier to manage the transformation processes of the generators, the aim is to define a system of reactions in a similar approach. Decomposing and composing the components of the algorithm can facilitate to achieve adaptability of the algorithm parts resulting in global adaptability of the system. It is easier to manage the tasks of the parts instead of dealing with the algorithm as a whole. In Tables 2 and 3, the transformation processes applied in the developed algorithm structure are presented.

Element	Action	Element	Action	Result	Type
idle	->	L.01			Activation
idle	<-	L.01			Deactivation
L.01	+	A.01	->	G.01	Composition type.01
L.02	+	A.02	->	G.02	Composition type.02
L.01	+	A.01	<-	G.01	Decomposition type.01
L.02	+	A.02	<-	G.02	Decomposition type.02
G.01	+	G.02	->	M.01	Combination
G.01	+	G.02	<-	M.01	Separation

Figure 12 The scheme for processes of transformations developed for the adaptable algorithm structure

The relations and transformation processes I define in Figure 12 are the types of actions and different levels of complexities in the algorithm structure. In this figure, {L.01, L.02 ... L.n} are the smallest instructional loops of the entire system. The transformation actions of the algorithm are “activation”, “deactivation”, “composition” {type.01, type.02 ... type.n}, “decomposition” {type.01, type.02 ... type.n}, “combination” and “separation”. By composition {type.01, type.02 ... type.n}, {G.01, G.02 ... G.n} are formed, which are combinations of loops and agents {A.01, A.02 ...

A.n} combined. Decomposition {type.01, type.02 ... type.n} process when needed dissolves the higher levels of groups into its constituents so that the system can evolve into another configuration of elements to serve a different function. As a result of the “combination” process, modules are {M.01, M.02 ... M.n} formed, which are the modules managing the overall generation process. When it is necessary, modules are dissolved into smaller components such as loops and agents by “separation”.

Organization of the algorithm structure emerging as a result of these reactions is presented in Figure 13. Hierarchy has, in this sense, the same definition as Simon (1968) gives: a system of interrelated subsystems, where the higher element is hierarchically superior to its subcomponent. Loops and agents form a group, and groups form the modules in this figure. The lowest level of organization in this scheme is the loop; the highest level of order is the module. These are constituents of a global whole, the algorithm. Loops are the main functions of the algorithm structure. The generative modules are composed of only loop elements whereas groups and modules are composed of both loops and agents.

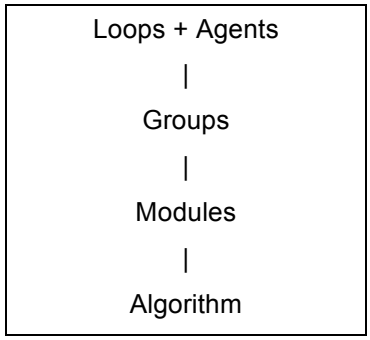


Figure 13 Organization of the algorithm structure

I investigate examples developed for generative processes. Kimura (2008) presents GLSDC (Global Local Search with Distance independent Diversity Control), where he talks about the algorithm structure and levels of generative processes. These

levels are correspondingly, initialization, local search phase, and evolution of the generation, converging phase and finally termination. Here the generative system terminates after all the criteria are satisfied. Thus, it lacks the argument about what happens with the arrival of future changes. This thesis tries to develop a structure where it is clear to predict how the system behaves when a change occurs in the environment. In addition to these levels, which most of the generative algorithms have in common, I propose a further step towards the adaptability of the generative algorithms.

To elaborate more on the processes of the algorithm structure, Gilat's (2005) processes of self-management of generative systems are harnessed. These comprise a number of stages (Gilat, 2005).[10] Gilat (2005) presents the adaptive level in this manner; where self-managing systems can automatically decide to take action based on the input that is available to them and the knowledge of what is happening in the system. This level refers to the learning process and to resolve an upcoming solution from this experience –knowledge. If this can be done autonomously, it can be said that the system is potentially able to adapt itself to the transformation. This autonomicity leads to the autonomic level, which the users can interact with the system to adjust their objectives (Gilat, 2005). The adaptive level and the autonomic level are the main conceptions to talk about the organization of the generative algorithm in question. Also Testa and the others (2001) state that an algorithm structure mainly consist of levels, which are designating the components, defining their agency in local terms and finding out the outcome of the individual interactions.

Referencing to all of these models above, I depict the algorithm structure composed of levels once more. The first level is the initialization of the algorithm after defining the design problem and implementing design data as input to the algorithm. The algorithm initiates with evaluating the attributes of the currently defined environment

[10] Gilat (2005) in his article, talks about the levels of self-managing processes, which are consecutively basic level, managed level, predictive level, adaptive level and autonomic level. For the sake of simplicity the last three levels will be elaborated. Since it is obvious that there will be an initial stage for the algorithm set-up and a management stage where the information is accumulated and put to use, it is unnecessary to go into detail of these levels in this chapter. Yet again, these levels shed light to the set up structure of the generative algorithm used in this thesis.

mainly the site forces.

Following the “initialize”, in the “generate” stages, the algorithm computes the configurations for the programmatic nodes according to the predefined parameters and constraints by the designer/user. After resulting in a “solution space”, if a stochastic search tool is embedded in the algorithm, this tool evaluates each one of these alternatives and the most plausible solutions satisfying the fitness criteria are selected. If such a search tool is not embedded in the tool, the designer decides on which solutions to choose from the solution space. In the “bake” stage after the search process, the algorithm is idle until a new job definition arrives and the generation procedure is called back.

After the arrival of a new objective, search tool integrated in the activation module evaluates the current generations. If the current condition is not satisfying the new design problem definition, the next level of “update” begins. This means that the algorithm needs to be re-defined. In such a case, the algorithm is decomposed into its parts and according to altered conditions relations between the parts are adapted to satisfy the requirements. The infinite loop of readjusting the algorithm parts prevails.

5.2. Introducing Agents to the Generative Algorithm Structure

The structure of the generative algorithm is a composition of individual elements acting to constitute a global behavior. The most important elements of these individuals are the agents implemented in the model to provide adaptability to the structure as much as it is possible to provide communication and interaction between the algorithm modules.

According to Wooldridge & Jennings (1995), an agent can be defined as an intentional computer system while trying to fulfill its assigned goal, operates autonomously and rationally; and interacts with the environment –besides having “social ability, reactivity and pro-active and messenger” features. Agents implemented in the developed algorithm structure, serve as mobile elements to detect the change in the environment or in the algorithm, and to start or to stop the

algorithm parts running. Also Holland (1995) in a different perspective emphasizes the capabilities of an agent in terms of adaptability. He ascribes agents as adaptive individuals in a global network, where these agents comprise smaller parts. He states that an agent can achieve adaptive properties if these smaller parts are re-aggregated to form another structure in the mechanism. This definition of an adaptive agent, although it is closely related to the issue of adaptability in this thesis, does not correspond to the purpose of implementing these agents in the system.

The generative model developed in this study makes use of agents to communicate within the model structure to inform algorithm parts and allow them to be aware of their changing environment. Agents in this manner are also used as transition elements for the communication between other parts of the mechanism besides acting like design elements monitoring behaviors of pedestrians or space layout bubbles. They are not used for the sole purpose of designing. In this thesis, it is also the case. Agents are put to use as communication elements between the algorithm parts to be aware of each other and their environment.

A definition cannot be enough to understand the notion of agents; on behalf of illustrating what an agent is and does, examples and descriptions are investigated. Sterritt and Hinchey (2005) talk about mobility property assigned agents, which can adapt dynamically and execute autonomously. They also came up with an outline for agents' properties in categories like mandatory and independent (Table 2). Some of these properties such as an agent being reactive and communicative come forward in relation to the utilization of agents in this thesis. According to Sterritt and Hinchey (2005) an agent must be reactive and have autonomous properties. In this thesis, an agent must be responsive to its environment. It is important in terms of achieving an adaptable algorithm structure.

Table 2 Agents' properties (Sterritt & Hinchey, 2005)

Agents' properties (mandatory)	Agents' properties (independent)
Reactive (adapt to changes in the environment)	Communicative (with other agents)
Autonomous (self-governing – control over own actions)	Mobile (travel from one host to another)
Goal Driven (proactive)	Learning (adapt in accordance with past experience)

An agent must also be self-managing to adapt to the transforming system it is part of. An agent detects the change and as a self-manageable mechanism it responds to the changing conditions. Considering the framework Sterritt and Hinchey (2005) propose, an agent is communicative, mobile and able to learn as optional properties.

Agents are implemented as communication mechanisms in the adaptable generative algorithm structure for the primary concern of providing interaction between the algorithm parts. Agents –especially inhibitor agents, are assigned mobile properties for them to move in the structure like change places in its series. This way, an inhibitor agent is able to reach its assigned destination and inhibit the related algorithm part.

GenOrchestra developed by De Felice and his group (2002) makes use of an agent-based design system to compose music with various individual mechanisms and feedback loops. General architecture of the system can be followed in Figure 14. Composer as a generator, maestro as a control mechanism and partly as an autonomous evaluator and user as the evaluator are basic components of this software. Query functions to learn and arrange the system are put to use. And feedback provides communication between these components. In this thesis, the algorithm developed is also composed of parts and mechanisms. This way, it is manageable to detect if there is a change in the system and if there is a malfunctioning of the components. Although there is no central control mechanism, with the implementation of agents, it is possible for the parts to communicate with each other, and with global behavior as a result, the change –in some cases interruption, is handled. Also feedback loops as communication tracks are developed for the algorithm to run sound.

In the generative model, agents detecting changes in the generation process are put to use. Besides the use of agent-based systems as modelling individual behaviors and analyzing the interactions between these agents and their environment (Eleni, Turner & Thum, 2002), this model makes use of agent-based systems for the algorithm to understand changing internal and external conditions and react accordingly. So this means in the model structure, transformation of the model does

not necessitate transformation of the parts. Leach (2009) proposes a multi-agent system consisting of intelligent agents interacting with each other and efficient in terms of spatial mobility. In light of these arguments, for the sake of developing a model easy to administrate with a decentralized control mechanism composed of individuals, there are different types of agents and parts implemented in the algorithm.

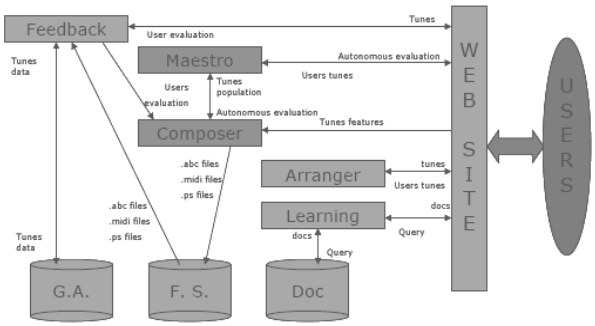


Figure 14 General architecture of GenOrchestra (De Felice, Abbattista & Scagliola, 2002)

In developing a generative algorithm structure, the key aspect of the generative design systems is the question of its adaptability. As discussed before in self-management of the generative design systems, I reasoned that it would be easier to manage a decentralized system with a global behavior composed of individual actions. Instead of tackling with the generative system as a whole, I suggested that it is plausible to work with smaller parts and then to achieve a resultant behavior out of these individuals' objectives. It is more reasonable to accomplish adaptability as a resulting global behavior orchestrated by reacting individuals. Algorithm parts or agents are not adapting to the new conditions, but with their specific configurations, the system acts as an adaptable whole. Therefore, it is a decentralized system with no central control mechanism. In this generative system, in addition to the algorithm's conventional generator structure, agents are implemented as communication and interaction mechanisms. I call them mechanisms for the reason that although they are attributed as elements, they serve for a higher level of

function in the system, and therefore they have their particular structure.

Maher and Gero (2002) talk about a multi-agent system where different types of agents are put to use (Figure 15). They speak of these agents in two main categories: “sensors” and “effectors”. In addition to these “situated agents”, in this algorithm structure other categories are included. Besides sensors and effectors, analyzers, verifiers and inhibitors are utilized.

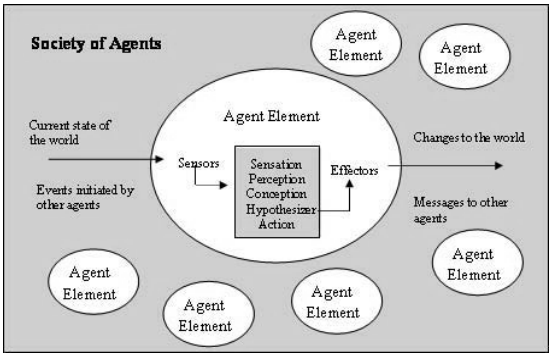


Figure 15 A 3D Virtual World as a Multi-agent System (Maher & Gero, 2002)

Agents as receivers

In this part, agents to detect if a change in the system occurs are discussed. I call these agents as “receiver agents”. I try to harness these receivers as agent mechanisms themselves, add detecting attributes and designate them as “receiver agents” from now on. This type of agents recognizes if a change occurs in the system. This can be by a new job definition or by a malfunctioning in the system. When a new problem definition is introduced to the system, sensor mechanism of the algorithm is stimulated. Receiver agent detects if something happens after the algorithm is idle following a generation phase. Receiver agent, then, combining with

the supplementary agent[11] forms the analyzer agent and therefore the analyzer agent is aware of the occurrence of a new condition in the whole system.

Agents as analyzers

In addition to the categories of agents as “sensors” and “effectors” by Peng and Maher (2007) and Maher and Gero (2002), “analyzer agents” as investigation mechanisms are integrated in the algorithm structure. These analyzer agents work as evaluation mechanisms investigating the type of the change and decide whether to take action or not. When a new definition enters the system, after the receiver agent is combined with the supplementary agent and forms the analyzer agent, analyzer agent performs accordingly. This change can be an extrinsic as well as intrinsic stimulus. External changes refer to the transformation of the environment of the system such as a new problem definition, altered site inputs or functional relations. Internal changes imply there is something in the system such as malfunctioning of the parts. Analyzer agent after evaluating what type of a change occurred in the system, decides if a reaction is necessary according to the condition of the system. If change does not require any measure to be taken, analyzer agent suspends the algorithm. If the interruption in the system necessitates an adjustment to the new conditions, analyzer agent informs the effector to take necessary measures.

Agents as effectors

Effector agents, having the ability to act on information obtained from the analyzer agents, control the rerunning of the algorithm generation parts. If the algorithm is idle after performing the assigned tasks, with the arrival of a new task definition, the adaptable generative algorithm should be able to adjust to the new definition. After analyzer agent interprets the information about the change occurring in the system and activates two effector agents in the system, one of the effector agents combines with another supplementary agent and forms the inhibitor agent. The other

[11] Supplementary agent is an idle mechanism constituted by a few lines of script, which is attached to a receiver agent in order to form an analyzer agent. So this supplementary agent is added to the end of the receiver agent script and analyzer agent is generated.

communicates with the adjusted mapping module commanding to rerun the script and call the generation procedure back. This way the idle system is activated with the call back mechanism. After inhibitor agents adjust the series structure composed of loops, generation module is activated.

Agents as verifiers

Activation module has the duty to control the generator parts of the algorithm. Verifier agent in the activation module is operated after the bake module command arrives implying that the objectives are accomplished. Following this command, verifier agent informs receiver agent that the task is performed.

Agents as inhibitors

In association with the stop module, these agents work as stop mechanisms to suspend the if-else loops with the arrival of a new problem definition. As illustrated in Figure 19, inhibitor agents interlock with the algorithm loops to deactivate them. This way, the series of if-else loops is adjusted to the requirements of the new objectives.

Structure of an agent mechanism

Referencing back to the Autonomic Element Sterritt and Hinchey (2005) present in Figure 8, which illustrates an example of a self-managing system, mechanism of an agent is constructed. Self-monitoring and self-adjusting features together with knowledge about the system are the most operational units to be implemented in the adaptable generative algorithm structure. Agent's knowledge about the environment precipitates environment-awareness possible. This scheme illustrating an autonomic element is exemplified for it represents how an agent can function as a mechanism. In Figure 16, the units and sub-mechanisms forming the agent mechanism implemented in the develop algorithm structure are represented.

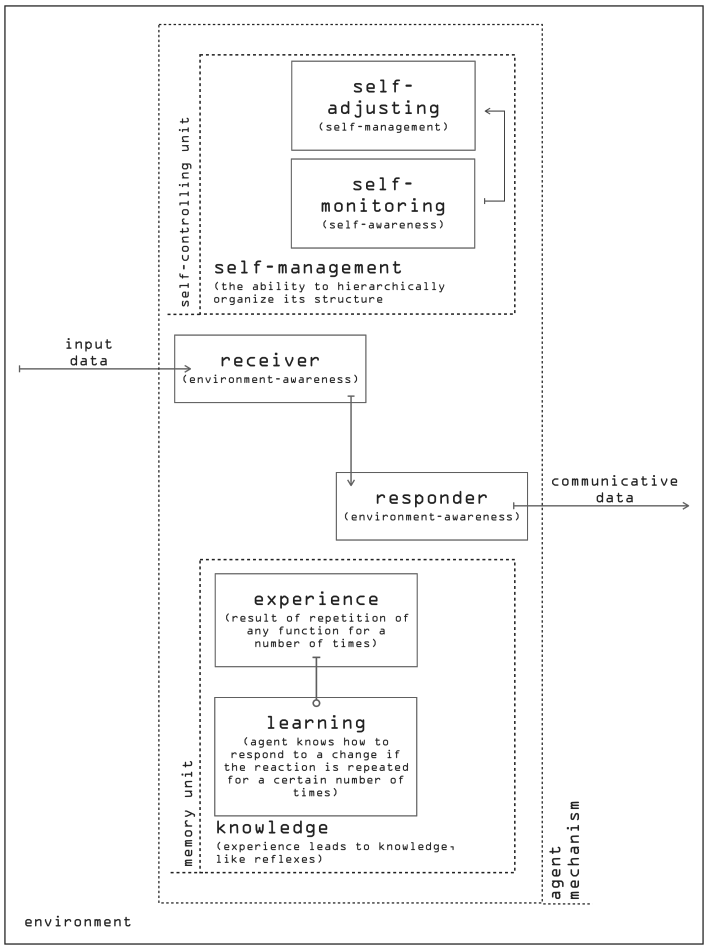


Figure 16 Agent mechanism illustrated in its environment. The mechanism is composed of a self-controlling unit and a memory unit, in association with a receiver and a responder.

By means of implementing sub-mechanisms to interact with the environment and to obtain the information essential for properly functioning, the element is instructed with tasks assigned to them where and when a change occurs in the environment. This ability to interact with the environment and being environment-aware is crucial to talk about adaptability of a generative system.

In order for the agent to detect the change in its environment, the agent must have sub-mechanisms to analyze and evaluate what's happening in the environment. These sub-mechanisms are represented in Figure 16 as receivers and responders.

The receiver interprets the input data coming from the environment. After analyzing the data, the receiver informs the responder to take proper action.

In addition to the environment-monitoring sub-mechanism, with a self-controlling unit –self-monitoring and self-adjusting, the element communicates with its internal elements and detects if the mechanism runs accurately. This self-controlling unit provides self-awareness and therefore self-managing characteristics in the agent mechanism that the agent knows how to achieve internal stability to be part of a higher-level global society. This self-controlling unit is applicable to serve as a mechanism to be practiced in the developed generative algorithm structure. Receiver agents also make use of such sub-mechanisms to be aware of the environment as well as their internal operation.

Agents, being self-aware and capable of self-adjusting according to the interactions within the system and to the changing parameters of the environment, in collaboration with the generator elements constitute the essence of the adaptable generative algorithm.

Besides the self-controlling unit and the sub-mechanisms detecting environmental change in the system, a memory unit is implemented in the agent mechanism. In relation to the discussion about the link between memory, learning ability and experience and the adaptability, this memory unit is integrated in the agent mechanism to facilitate the functioning of the agent. The agent after repeating the same function a number of times also gains experience in relation to the evolutionary computing. The agent learns how to respond to the stimulus and with this knowledge the agent reacts quicker. I do not go into details of the issue of memory but I suggest that these issues can be further studied.

Implementation of agents in the generative structure

Agents collaborate with the generative parts of the model. These agents are implemented within the modules, which control running of the generative parts.

Interactions and communications between the algorithm parts and the agent

mechanism are illustrated in Figure 17. In this figure, interactions between the components of the algorithm result in the generative process, which precipitates the model to gain generative properties. After the generative parts of the model such as mapping module, generation module, search and bake module accomplish the assigned task and result in a desired solution, and after satisfying the objectives, the generative parts become idle.[12] The system stays idle until the generative procedure is called back with the arrival of a new job definition.

With the arrival of the new job definition, agents involve in the process of adaptation to adjust the system pursuing to accomplish the new objectives dictated by the new design problem. New job definition activates the receiver agent to form the analyzer agent. Analyzer agent inspects the type of change according to where the change takes place. If the impulse is a new job definition or a change in the objectives of the problem, it is denoted as external such as site input or functional relations.[13] New definition of the design problem might indicate completely different objectives than the former task descriptions. So it is important for the generative model to be able to reconfigure its part relations in order to properly accomplish the new objectives. That's why I suggest breaking the algorithm down to its constituents from which new relations defining new objectives can be obtained.

[12] At this point, I do not go into details of the stochastic search process in the model or how this process is handled. It is more crucial to talk about the processes how this generative model achieves adaptability such as how agents are structured.

[13] Analyzer agent also detects if there is a malfunctioning in the system, and through self-manageability of the system the failure is compensated. I suggest that compensation can be either with replacement of the malfunctioning parts or activating the alternative circuit implemented on the model. By means of this compensation, self-awareness of the generative model is achieved. Designer is the decision mechanism to implement an alternative path for the probable failures occurring in the system.

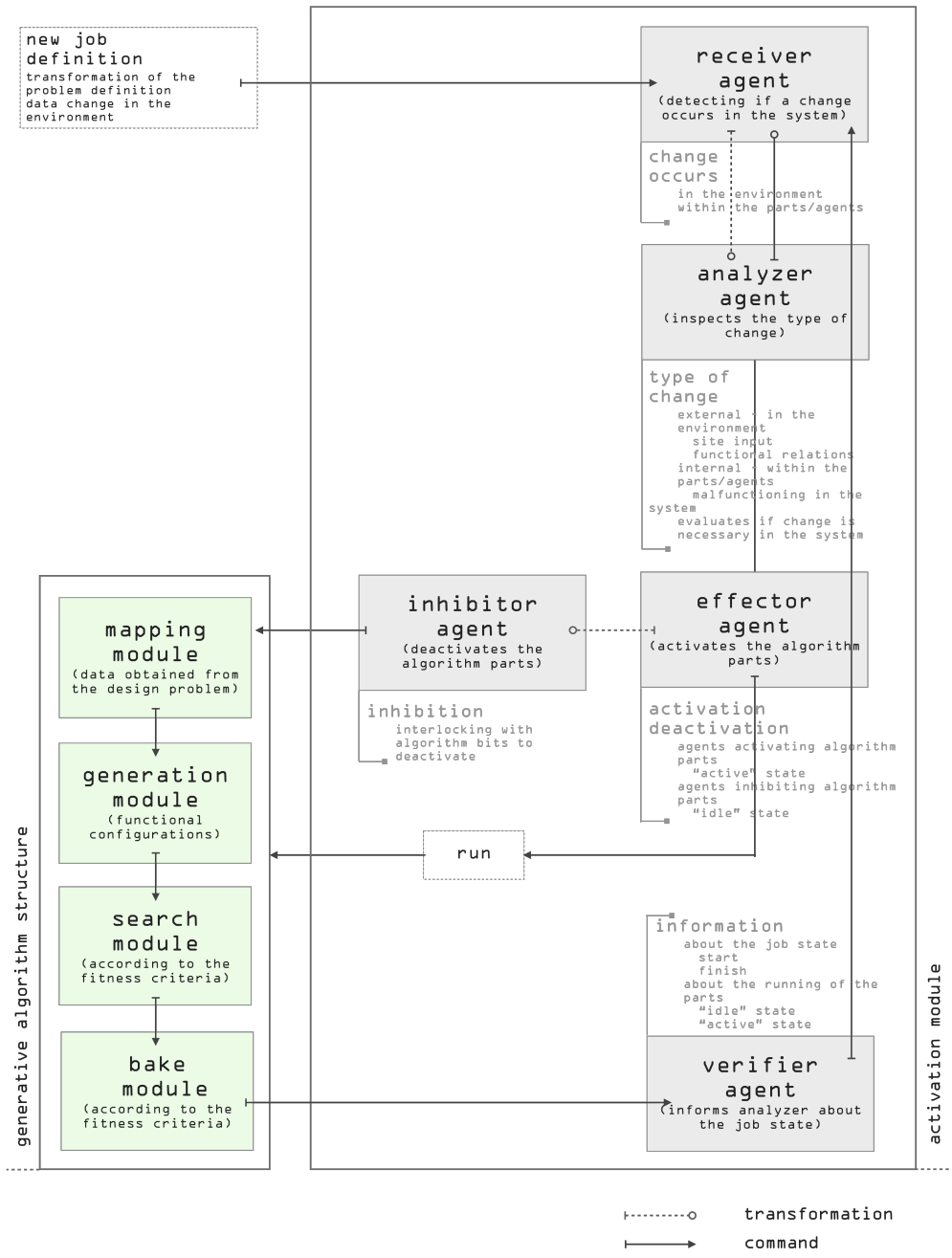


Figure 17 agents and association between agents and algorithm parts

Analyzer agent after identifying the type of change and evaluating if the structure of the system should be altered, communicates to two discrete effector agents. One of the effector agents forms inhibitor agents to adjust the series properties, which is composed of loops, while the other calls the generative procedure back. This way by sending run signal to the generation parts of the algorithm effector agent

activates the idle parts. With the introduction of the new input obtained from the problem definition, receivers in the mapping module process the information and start the generation process. Following generation of a set of design solutions and searching process, generation structure informs the verifier agent that the job is completed. Verifier agent communicates back to the receiver agent that the task is accomplished. This procedure is like switching on the circuit all the way to the receiver agent to provide data to flow through this circuit to the receiver agent. The system is idle until a new task arrives.

Also in Figure 18, different than the previous figure, modules are represented as composed of loops and the integration of agents In Figure 17, the connections between the modules and the agents differ according to the interaction type between the components. A straight line with an arrowhead represents a “command”, indicating that the instruction goes to the component to which the arrow points. A dotted line ending with a circle represent that the component transforms into another component to which the circle points.

In Figure 18, modules are represented as composed of loops and the integration of agents in the algorithm structure and their transformations are illustrated.

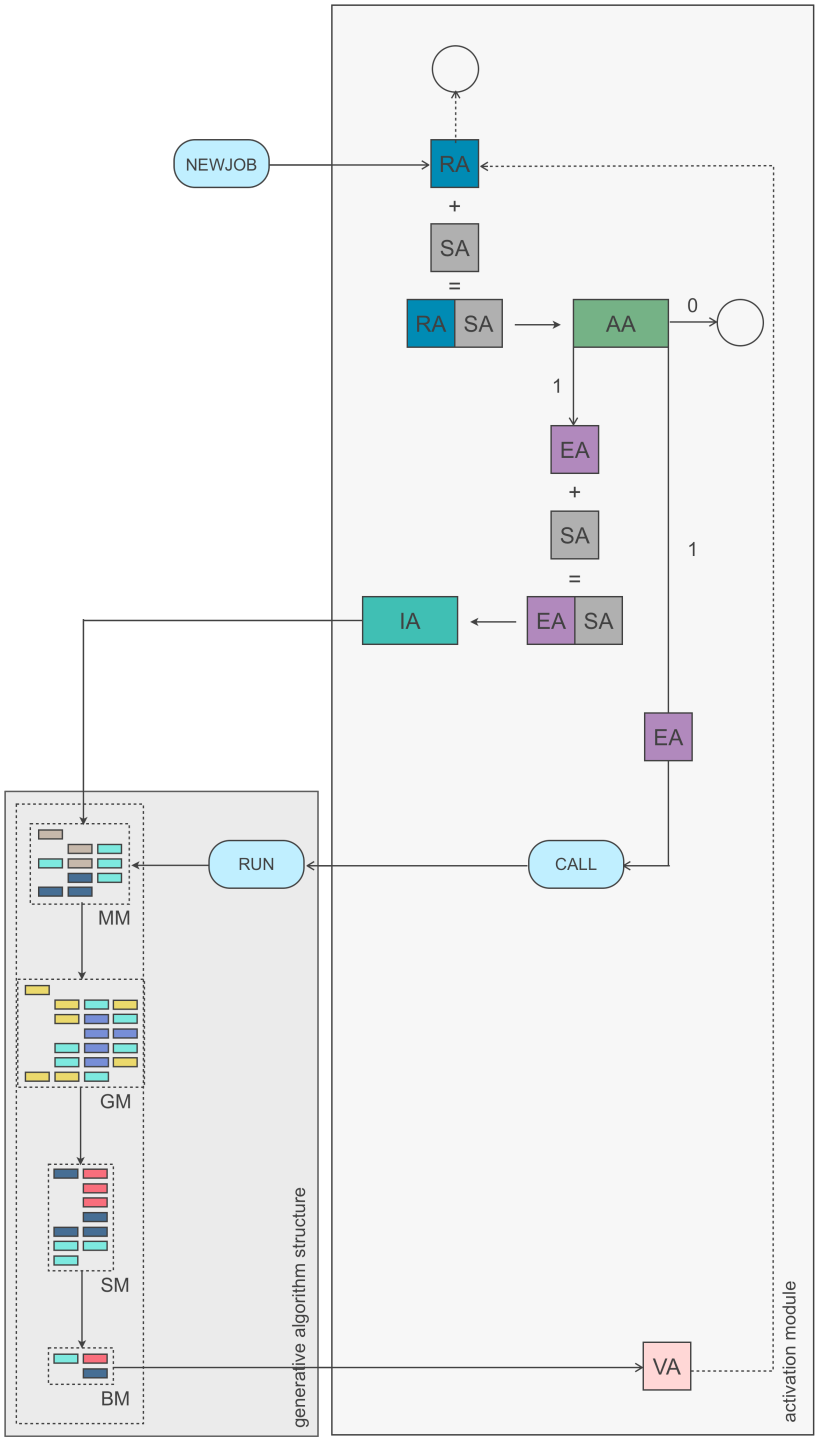


Figure 18 Modules composed of loops and integration of agents in the algorithm structure when problem definition changes. MM: Mapping module, GM: Generation module, SM: Search module, BM: Bake Module, RA: Receiver agent, AA: Analyzer agent, EA: Effector agents, IA: Inhibitor agent, VA: Verifier agent, SA: supplementary agent.

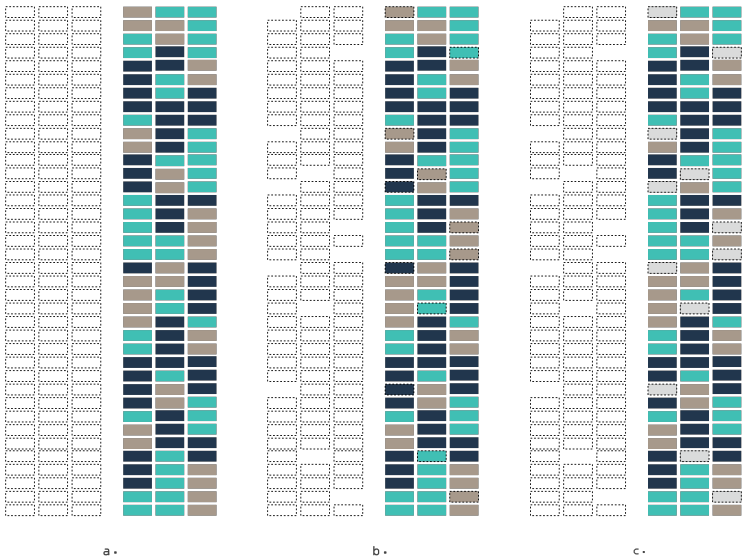


Figure 19 Representation of the agents and the loops. a) Inhibitor agents on the left side, loops on the right side. b) Agents combined with loops c) Agents deactivating loops

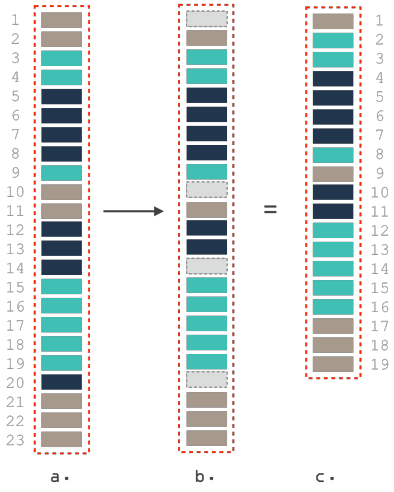


Figure 20 a) a series of if-else and repeat-until loops b) idle loops interlocked with agents c) series length and property change when inhibited loops leave the series.

Illustrating how an inhibitor agent operates and to exemplify how this activation/deactivation process changes the structure of the modules for them to

adapt to their changing environment, Figure 19 illustrates inhibitor agents and the if-else loops. Agents are on the left side of the figures while loops are on the right side. When problem definition changes and the generative algorithm is expected to alter its structure, inhibitor agents depart from their dock and snap to their assigned loop. After deactivating the loops and carrying them back to the agent dock, the series composed of loops is changed (Figure 20). The transformation of the series provides the transformation of the structure, which is redefined by the new input introduced in the mapping module. These series of actions are carried out according to the needs specified according to the new problem definition. The adaptability of the system can be achieved with the processes discussed so forth.

5.3. The Flow of the Algorithm Structure

In order to deliver the algorithm and the flow of the processes, I try to develop a sequential flow structure for the adaptable generative algorithm.

An example to investigate is a generative algorithm for an interactive agent-based system, modelling the pedestrian-environment interactions (Eleni, Turner, & Thum, 2002). This example of a recursive algorithm satisfies in each step the given conditions before proceeding to the next level. This algorithm makes use of “if-else-end if” loops to achieve a sequence of actions.

Based on the structure of the generative algorithm I designed, I try to develop a sequential flow structure for the adaptable generative algorithm. Flowchart of the algorithm with the new job definition processes is illustrated in Figure 21. Starting with introducing the input derived from the site analyses, the algorithm is initiated. After generating a population of solutions, the solution space is searched and the best-fit solution is baked. The algorithm is idle until a new task is defined. To activate the generative parts, the activation module needs to call introduce input procedure back. In order to be sure if a change is needed in the system, after the new problem is defined, generated outputs are looked over again. If the current structure of the algorithm satisfies the new problem definition, the algorithm terminates. If not, introduce input procedure is called back.

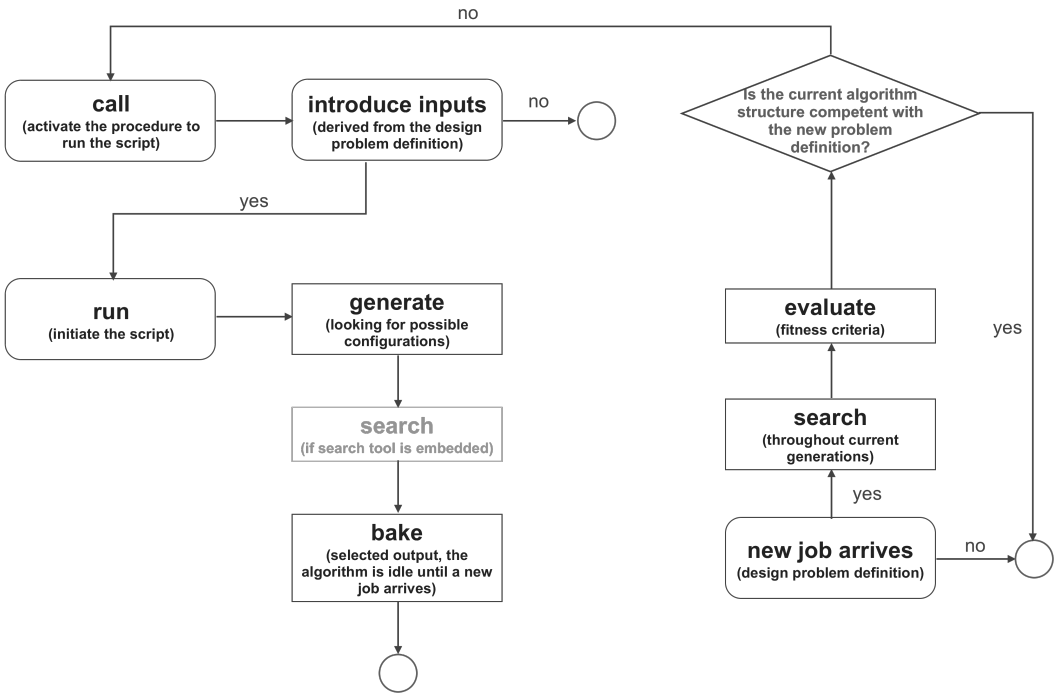


Figure 21 Representation of the algorithm flow

Making use of the flowchart in Figure 21, I tried to develop a structure for the flow of the adaptable generative algorithm. The generative process starts with defining the procedures and introducing the inner loops functioning as command modules in the system. These if-else and repeat-until loops constitute the basic communication network between the modules. Referencing back to the flowchart, with introducing design problem input to the system, the script is initiated with run command. Respectively convert input to data, generate, search and bake modules run. Interdependent relationships between these modules are hierarchically organized so that a low level module cannot take action before the higher-level module is executed and then terminated. If there is a new job defined by the designer, call mechanism process begins. Respectively, search, evaluate and if adjustment is needed according to the change call modules take action.

Figure 21 illustrates a chart developed for the agents with the arrival of new definition. This flowchart presents how the adaptable mechanisms function in the algorithm structure. Related to the flow of the adaptable algorithm, agents are integrated in the call mechanism. New job activates the receiver unit embedded in the receiver agent mechanism. Receiver agent responds to the information and combines with a supplementary agent and forms the analyzer agent. Analyzer agent evaluates what type of change is in the system with an if-else loop. If the change is external, analyzer agent searches through current generations to decide whether the algorithm is competent with the new job definition and the solution space created in t=1 satisfy the requirements or it is necessary to adapt the generative parts according to the new job. If the organization of the generative parts does not satisfy new objectives, analyzer agents call effector agents, one to form inhibitor agents, the other to call the adjusted mapping module. Inhibitor agents adjust the structure of the generative parts. After the generation process is terminated, verifier agent is informed and the circuit the feedback mechanism is switched on to the receiver agent to inform the algorithm is idle.

After defining the “introduce” procedure, the algorithm starts to run when the condition of “design problem input is introduced” is true. Fitness criteria and constraints determined by the designer are also defined in the system. The input is converted to algorithm data[14]. After all input is converted to data, "generate” command is executed. And “generated” attribute is assigned to all configurations generated by the algorithm. When the number of the “generated” configurations is equal to the number of permutation of possible nodes, all of the generated configurations are added to search queue. After the generation stage is finished, “search” command is executed and all of the generated configurations added to search queue –the solution space- are evaluated according to the constraints and fitness criteria defined by the designer. If search command finds a satisfying solution, bake command is executed and the algorithm becomes idle. If none of the generated solutions satisfy the fitness criteria, the algorithm returns back to the “introduce input” procedure and becomes idle until new design input is introduced in the algorithm.

[14] An example for this type of data can be exemplified with transformation mappings produced for the graduation project in the chapter 2.

These actions are hierarchically dependent on the previous action for the algorithm to proceed linearly. It is not relevant to talk about non-linearity in this sense. Non-linearity when the overall flow of the adaptable algorithm is in question is a matter of concern. With the arrival of a new design problem definition, the linear process of the algorithm is interrupted and a new generation loop is created. This is where the activation module creates another loop different from the generation loop for the call mechanism. With the introduction of the new job definition, the linearity of the generation process of the adaptable algorithm is interrupted and non-linearity is represented with the development of an outer loop.

With the arrival of a new design problem definition, defining “new job” procedure for the new job definition follows the basic generative process of the adaptable algorithm. Agents –receivers, analyzers, effectors, verifiers, and inhibitors- are defined and introduced in the system. New job definition brings along the statement of “change” and change is defined as being equal to “1” if it is an external influence such as the new problem definition. If there is an internal change in the system such as malfunctioning of parts, change is defined as being equal to “0”. “Backup” procedure is defined for the condition of “change” being equal to “0”. This procedure restores the algorithm to a stable state. With the new design problem definition, the designer also redefines fitness criteria and constraints and these should also be reintroduced in the system. After defining the procedures and constraints, and introducing agents to the algorithm structure, if “new job” statement is true and the algorithm recognizes the new job definition, the algorithm is initiated again to run the new job procedure.

Figure 22 represents the function chart of the agents when the environment changes, and depicts mobile properties of the agents. If the new job statement is true and search and evaluate processes are finished, “receiver agent” is called. Being informed by the new job definition, receiver agent combining with a supplementary agent transforms into the analyzer agent to analyze and evaluate the type of data. If the change is equal to “1” meaning that it is an external impact –such as an environmental change- analyzer agent assigns external to the change and searches throughout the generated configurations already existing in the solution space. This process assigns “done” attribute to the searched solutions and these

solutions are added in the evaluation queue. Search continues until all of the generations are added to the queue and “evaluate” command is executed. All of the solutions are evaluated according to the changed fitness criteria reintroduced in the system after the new job definition. If there are solutions satisfying new fitness criteria, these are stored in the memory to be added to the solution space that will be generated in the next phase. If none of the generated configurations satisfy new constraints, the algorithm directly steps up to the next phase, which is calling the generative procedures back. This also means that the generative parts of the algorithm are expected to change. In this case, the analyzer agent calls two effector agents. After one of these agents combining with the supplementary agent transforms into the inhibitor agent to activate or to deactivate the loops composing the modules, the other effector agent calls the input procedure back. After all the generations are completed and terminated, bake module communicates over verifier agent to the receiver agent to inform the receiver agent that the algorithm is idle.

If the type of the change is equal to “0”, analyzer agent assigns “internal” to the change meaning that there is malfunctioning in the generative system. Analyzer agent then calls “backup” procedure to restore the algorithm to a stable state and exits the loop.

To implement the presented sequential flow of the adaptable generative algorithm in the graduation project, in the coming section I propose a method to integrate the algorithm in a realizable project.

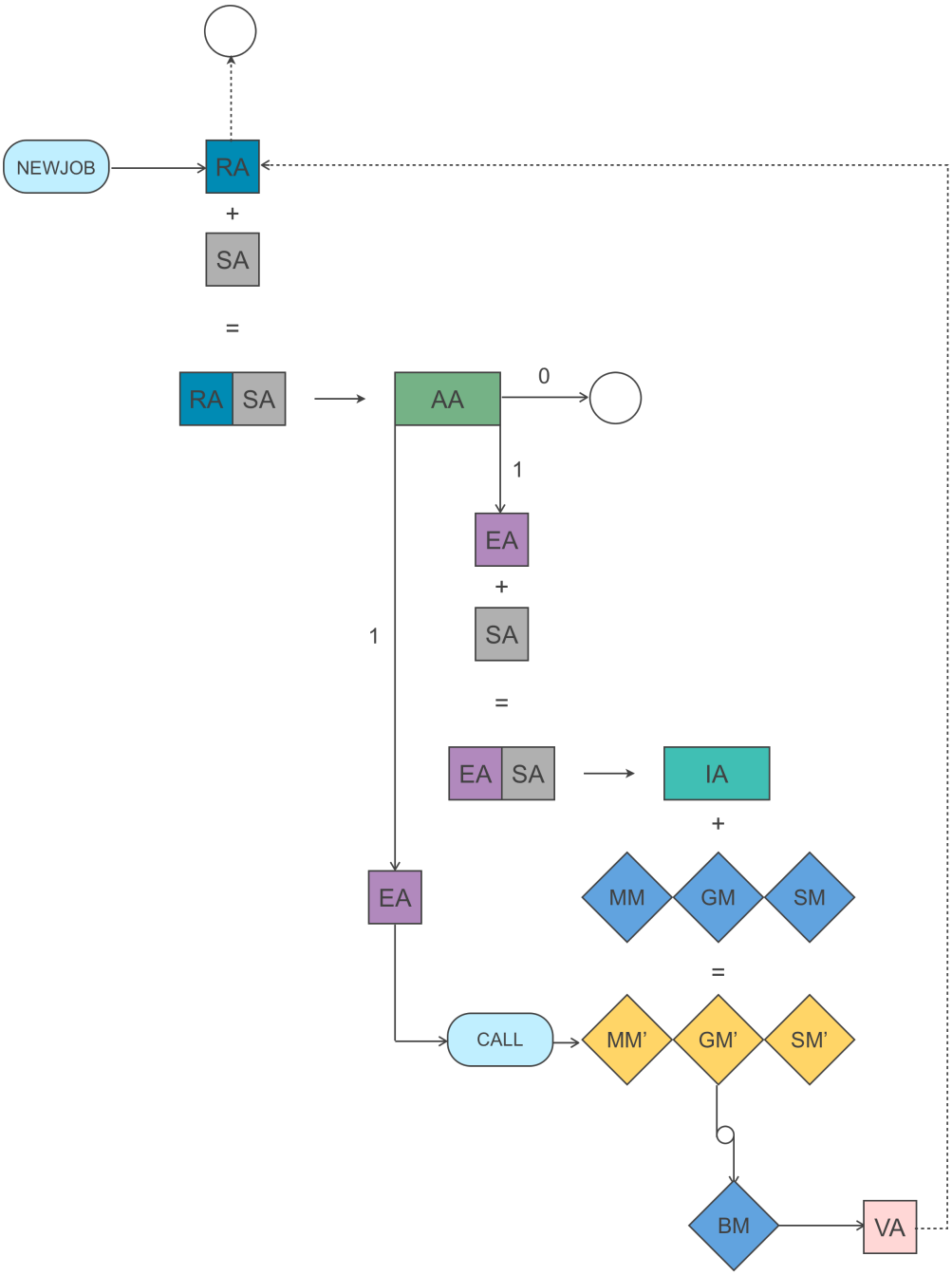


Figure 22 Function chart of the agents in case of a new problem definition. RA: receiver agent, SA: supplementary agent, AA: analyzer agent, EA: effector agent, IA: inhibitor agent, VA: verifier agent, MM: mapping module, MM': adjusted mapping module, GM: generation module, GM': adjusted generation module, SM: search module, SM': adjusted search module, and BM: bake module. In this figure, values of "0" and "1" represent the type of change where "0" means that change is internal and "1" means that change is external and the generative parts are expected to change.

5.4. Proposing How to Integrate the Adaptable Generative Algorithm in the Graduation Project

This thesis also looks for methods to implement the adaptable generative algorithm proposal in a project in order to evaluate if this developed system provides solutions to the problem definition of this thesis. This proposal is theoretically developed to implement the structure in the graduation project.

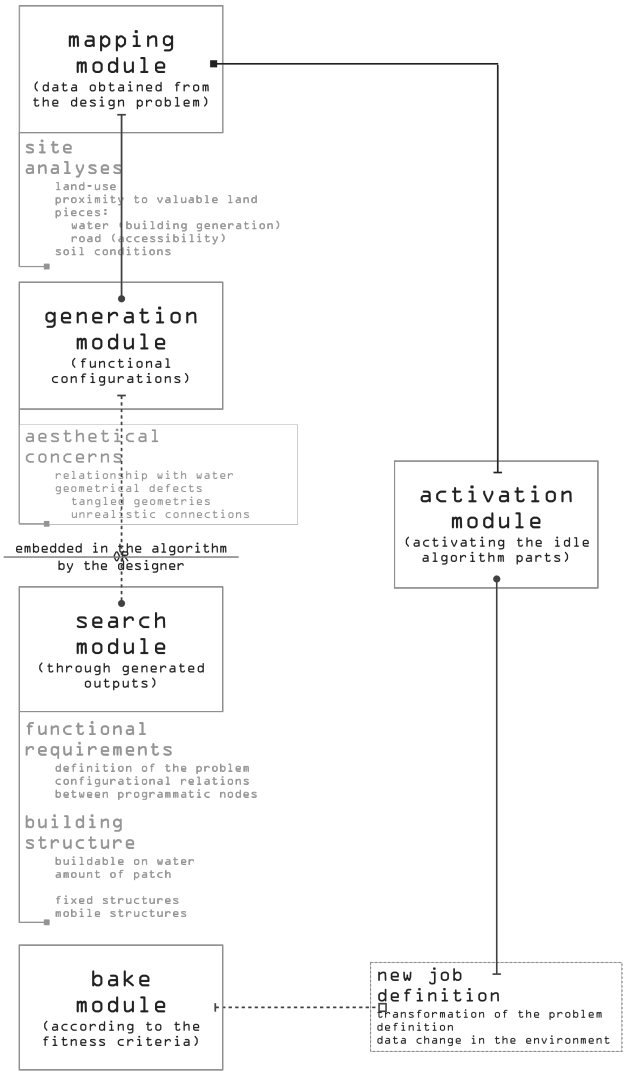


Figure 23 Algorithm structure in relation to the project implementation

To examine if it is possible to utilize this algorithm in a realizable project, which in this case is the graduation project, the algorithm structure concerning the project implementation is presented in Figure 23. This modular structure represented in Figure 23 is adopted in relation to the design stages used in this graduation project reflecting on my design process. These modules can be reorganized in order to be applicable in any design process.

MEL (Maya Embedded Language) is proposed for scripting and translating the pseudocode into a programming language in order to translate data derived from site and which is essential for the design problem definition.[15]

Mapping module is used for converting visual data obtained from all types of analyses –such as site analyses- to computational data to be processed in the algorithm. Generation module produces a population of functional configurations according to the constraints I defined considering the functional and the visual relations between the volumes. After the solution space is created, through these generated outputs search module provides me the solutions satisfying my fitness criteria like functional requirements, geometrical impossibilities and so forth. Bake module freezes the solution I select and the generative process of the algorithm is terminated. Until this stage, the algorithm is in use in the graduation project.

From this point on, when the landscape is transformed with the ongoing experiments, activation module operates and the generation sequence is reinitiated if it is needed.

5.5. Customization and Generalization of the Algorithm

The issue of customization of the scripted generative algorithm in relation to the specific design problem is studied and the designer's interaction with the algorithm is investigated.

[15] MEL is the scripting language to instruct Maya to execute a series of actions through commands typed in the command editor (Terzidis 2006).

Dynamism of these generative design systems should be handled in two meanings. One is the system transforming its structure; the other one is the possibility for the designer/programmer to adapt the algorithm according to his/her parameters and constraints. The user/designer can adjust the system to obtain a suitable generative structure for his/her own problem definition. This open feature is intentional to a certain degree to allow the user/the designer to be involved in the process of defining the structure of algorithm. The basic operators being defined allow the algorithm to be implemented to a wide range of design problems.

The adaptable generative algorithm designed in thesis, where the algorithm itself is the most pliable structure in terms of personalizing for the reason that it does not impose a specific environment to work with, is open to transformations to satisfy any designer's needs. In this thesis, I designed the algorithm to perform a determinate task, which I found intriguing to provide flexibility with integration of adaptable generative systems from the graduation project explained in chapter 2. Structure of the algorithm is molded into what the scope of this thesis dictated, therefore, the modular system of the generative structure can be reconfigured by any designer to come up with personalized working environment. Personalization of the agents is also possible for the fact that these agents are implemented in a series of modular units. When customization is in question, the structure of this modular system can also be rearranged according to specific needs of any design problem definition.

CHAPTER 6

DISCUSSIONS AND CONCLUSIONS

This chapter provides a composition of the arguments this thesis constructs and the development of the adaptable algorithm structure. It presents discussions about adaptability of the generative algorithm to sustain design processes. This chapter also talks about the potentials and limitations of the study, which suggest further studies and draws some conclusions and presents a summary of the study. It evaluates the methodology of the thesis; adaptability of the generative algorithm developed, talks about potentials, limitations and contributions of this study and suggests further studies.

6.1. Methodology

Departing from the problem of design problems that I encountered in my graduation project, this thesis investigated ways to develop a generative system to provide flexibility to maintain dynamic design problem definitions. The already existing question of constantly changing structure of the design problem definitions and the search to find ways to handle these changes in order to sustain a dynamic design process, this thesis explored utilization of the adaptability of a generative system and integrating this adaptable system in the design process.

With this intention in mind, I inquired into the adaptability of generative design systems by investigating through evolutionary systems such as genetic algorithms, cellular automata and swarm intelligence in relation to organic computing. In addition, I tried to find links between self-management of generative design systems and the notion of adaptability where it is the ability of a generative system to

hierarchically orchestrate its individuals' behaviors and as a result having some degree of autonomy. I discussed self-management, associated with self-organization and autonomicity, that these are the essential features for an adaptable system.

I investigated the issue of emergence and emergent behaviors of generative design systems in relation to the notion of evolution and I looked at examples of generative design systems and examined how adaptable these systems are and I questioned how to achieve adaptability of these systems.

To be more specific about my argument, I investigated the use of generative algorithms in design processes. Inquiring about the potentials of the utilization of generative algorithms in design processes and different applications of such algorithms in architectural field, I formed a framework to discuss adaptability of the generative algorithms. I investigated adaptability in relation to dynamicity and autonomicity to understand how the process of adaptability works. I did not come across with a wide variety of examples for the utilization of generative algorithms specifically designed to sustain dynamic design processes, so I developed my own definition of an adaptable generative algorithm to support my argument. Integrating what I examined and understood from these different applications of generative algorithms and my goals to achieve an adaptable generative algorithm, I built up generative system composed of different components. These components are the modules and the agents constituting the algorithm structure.

I developed an algorithm comprising mainly two parts, the generative structure of a common algorithm and the communicative mechanisms of agent-based generative systems. These independent systems are integrated to constitute an adaptable generative algorithm. The algorithm runs, generates and terminates as in all of the generative design systems. The point the issue of adaptability is incorporated in the design process where the design problem definition changes as a result of the changing demands of the design task, the designer or the environment. Agent mechanisms provide the analyses of the change, the evaluation about how to react and finally the response to the change occurring in the system.

According to the algorithm structure and the flowcharts I developed, I also came up with a sequential flow of the adaptable algorithm to make the process of adaptation clearer. And I proposed how to implement this adaptable generative algorithm in a realizable project discussing it over my graduation project. And I proposed and gave clues about how to customize this generative structure for the sake of adapting it to any design problem; and by means of this, how to generalize this algorithm to be implemented to any desired design problem. Modular structure of the generative system provides ease to reconfigure the relations between the modules satisfying the problem definition. These issues of the algorithm's evaluation process and customization and generalization possibilities are suggested for further studies.

I discussed the adaptability of the generative algorithm I developed over the examples and the discussions I put forward and evaluated my own process of developing the adaptable generative algorithm. And as a conclusion, in this chapter I summarized what I achieved so far.

6.2. Adaptability of the Generative Algorithm

To understand how a generative design system can be structured so that the mechanism holds an adaptable structure, so far, self-management, self-organization, self-awareness and autonomicity properties of such systems are explored. I proposed an adaptable generative algorithm structure in combination with the utilization of agents in the system. Referring back to Figure 8, the autonomic element Sterritt and Hinchey (2005) proposed, self-awareness of this generative system is crucial for the algorithm to respond properly to its environment and be well-informed about the current condition of the system as a whole.

This study employs a generative algorithm to investigate if such generative algorithms are efficient in sustaining dynamic design processes. Through discussing adaptability of generative design systems in relation to evolutionary models and generative algorithms, an understanding of and a background for how adaptability of such systems can be achieved to support the dynamicity of the design problem definitions.

Examining the examples from different principles and commenting on these to develop the adaptable generative algorithm I want to propose, I look for ways to sustain dynamic design problem definitions. After the algorithm is programmed, translated into a programming language and practiced in a realizable project, the adaptability of the algorithm and the results of implementing the algorithm in the design process can be assessed.

One of the important purposes of using generative algorithms is to control the evolution of the generations of problem solutions. The process of generation initiates with finding alternative solutions for the desired configuration followed by an evaluation process to generate a new population with a higher fitness value. This evaluation process is made use of to lead to a final set of results, which then will be subjected to the external simulation software. In addition to the use of generative algorithms as computational search tools to find a set of possible solutions and additional software to evaluate these solutions, this thesis investigates to develop a generative algorithm, which can also be used as a validating mechanism to test if the system runs meeting the desired objectives. This means to propose combining the external software and the generative algorithm as a whole.

6.3. Potentials and Limitations

The aim of this thesis is not to propose a new design process definition or to dictate that the use of generative design systems is indispensible. Rather, I illustrate a framework for a generative algorithm. How to achieve generative design systems covering a wider scope is suggested in further studies.

I do not suggest that a design process can be reinvented as algorithmic operations; instead I imply that such generative systems can be articulated so that it is possible to sustain the dynamic nature of design processes. I designed this algorithm, which is an abstraction of a process to a finite number of steps and variables to accomplish a determinate task I found challenging to provide flexibility with integration of adaptable generative systems from the graduation project. The generative algorithm structure developed with a modular system to manage its adaptability and to control how this system responds to its changing environment

provided me to handle the dynamicity of the design problem. With decentralized local interactions between the components of the algorithm, it is easier to manage these local interactions than to adapt the global behavior of the algorithm as a whole. Integration of agents in this algorithm structure allowed me to handle these local communications between the components of the algorithm and the transformations of the generative parts.

This adaptable generative algorithm has a number of potentials such as the convenience of the algorithm to be practiced in design problems where changing environments prevail and the simplicity of its structure as a result of its modular system. By means of developing a modular algorithm structure, it is also beneficial in terms of personalization of this algorithm into any design problem definition where the architectural concern is to make use of an adaptable generative design system.

The proposed generative algorithm could have been designed in a parametrical way so that it can be adjusted to any design problem definition. If the parameters change values, the algorithm changes its outcome and an adjusted outcome is obtained, different than the previous generated design solution. But in this thesis, it is not the case. The reason why this algorithm is defined as an adaptable generative system but not a parametric one is that its main concern is being able to adjust its "structure" to any problem definition and sustain the dynamicity of design problems. It's not always true that predefined parameters and constraints will correspond to the new needs of the changing problem definition. In a case where problem definition changes so abruptly that any predefined parameter or constraint cannot meet the new definition's needs/requirements such as functional relations, size of the built-scape, and site input, this algorithm structure will adjust its loops, groups, modules and agents to form an improved version out of its main components.

The algorithm developed in this thesis is based on the references and examples discussed; it is an empirical study for the use of adaptable generative systems in architectural design as a means to sustain the dynamicity of the design problem definitions. This adaptable model is presented as a proposal for developing a generative algorithm out of it. Therefore, its practice and realization in the architectural field is necessary to fully grasp the potentials of the suggested

generative model. Testing the model in a realizable project, outcome of which itself is a dynamic architectural object enhances the liability of this argument and leads to further studies to discover potentials of this generative model.

The scope of this thesis does not include the discussion of how a design process is modeled, it is not concerned whether a design process is linear, or not. It only delves into potentials of utilizing computational tools as systems to provide flexibility to the process. Related to Archer's design process model, a further question can be raised. What happens in the design process if the desired output is already defined before defining the problem? What happens if the process of designing a desired architectural object is the main concern of the designer instead of the architectural object itself? This thesis acknowledges the presented design process models and builds the argument being aware of these questions. Further studies can be conducted to find answers to these questions.

The scope of this thesis does not cover the process of searching for the satisfying solution. It does not focus on how to decide on evaluating which solution generated by the model meets the fitness criteria defined by the designer. It is intentionally left optional, up to the designer's intentions to embed a search tool in the model or select his/her satisfying design solution according to the desired objectives.

This thesis does not aim to exclude the designer from the creative design process; rather it deals with the practice of adaptable generative algorithm in dynamic design processes conducted by the designer. All the data provided for the generative algorithm to start running is defined by the designer and is obtained from the design problem definition, which the designer specifies. Designer is the conductor of these dynamic design processes in question. Although the proposed generative algorithm is suggested to be autonomous and self-managing, if there is a need to alter the problem definition, designer is the main control mechanism in the process. Therefore, designer is in constant communication with the design problem and engages in an interaction with the computational tool in use. The designer redefines the problem definition to satisfy his/her own objectives and introduces this new definition to the computational tool so that this tool can also satisfy his/her own design goals.

6.4. Contributions of This Study

Examples investigated in this thesis provide a frame for developing the structure for a generative system. These examples varying from agent-based transportation systems to designing with swarm intelligence, from digital music composers to performance softwares guide to develop a structure for a generative system –a generative algorithm. I think that these systems in the sense that they deploy adaptable and agent-based structures are worth investigating but in the sense that they lack the efficiency to provide the dynamicity for design problem definitions it is more intriguing to adopt these generative systems and develop a specifically designed adaptable generative algorithm.

This study points out that it is necessary to integrate adaptability of any design process a designer pursues to sustain the dynamic design processes. It is important to provide the proper amount of flexibility to a dynamic design problem definition, which transforms in relation to changing demands of any entity belonging to the design problem, whether it is the designer or the environmental conditions. At this point, this thesis offers to develop an adaptable generative algorithm to sustain the dynamicity of the design processes and contributes to the research about adaptability of such systems for further studies.

The methodology used in this thesis is adopted as bringing together discussions about generative design systems and their potentials of adaptability and examples from different disciplines and as a result developing an adaptable generative design system. By investigating and examining examples of generative algorithms in design processes and implementing what I infer from these examples and discussions in the adaptable algorithm I developed would be the main contribution of this study in the academic field. The ability of the generative system to reconfigure its structure and adapt to changing objectives is the driving force for next generation generative systems. Since the adaptable generative algorithm proposed in this thesis is my own development of a generative design system, it brings my point of view and it contributes to the understanding of how to sustain dynamic design processes, which I believe needs to be further studied.

6.5. Suggestions for Further Studies

I would like to conclude this thesis by suggesting new research questions and further studies regarding the limitations of this study.

Little information is given about how these agents interact with each other. Besides interacting with and being aware of the environment and correspondingly the changes in the environment, this thesis suggests to further study how these agents communicate with each other.

For further studies, I suggest that an evaluation mechanism of the adaptable generative algorithm to analyse if the algorithm is running properly meeting the requirements of the objectives can be implemented to the overall algorithm structure. Also components to detect and prevent unwanted emergence can be implemented in the generative design structure.

BIBLIOGRAPHY

Akkawi, F., Bader, A., Fletcher, D., Akkawi, K., Ayyash, M., & Alzoubi, K. (2007). Software Adaptation: A Conscious Design for Oblivious Programmers. *IEEE Aerospace Conference*, (pp. 1-12). Chicago.

Barczik, G., & Kurth, W. (2007). From Designing Objects to Designing Processes: Algorithms as Creativity Enhancers. Predicting the Future 25th eCAADe Conference Proceedings. (Pp. 887-894). Frankfurt am Main. Retrieved 12.08.2010, http://cumincad.scix.net/cgi-bin/works/Show?ecaade2007_157

Batty, M. (1997). Cellular Automata and Urban Form: A Primer. *Journal of the American Planning Association*, *63* (2). (pp. 266-274).

Batty, M. (2005), *Cities and Complexity: Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*, The MIT Press.

Beekman, M., Sword, G., & Simpson, S. (2008). Biological Foundations of Swarm Intelligence. In C. Blum, & D. Merkle, *Swarm Intelligence* (pp. 3-41). Berlin: Springer.

Bisig, D. (2005). Biosonics-interactive growth system. In G. M. Buurman, *Total Interaction* (pp. 290-303). Basel: Birkhäuser.

Blackwell, T., Branke, J., & Li, X. (2008). Particle swarms for dynamic optimization problems. In C. Blum, & D. Merkle, *Swarm Intelligence* (pp. 193-217). Berlin: Springer.

Bullock, G., Denham, M., Parmee, I., & Wade, J. (1995). Developments in the use of the genetic algorithm in engineering design. *Design Studies , 16* (4), 507-524.

Caldas, L., & Norford, L. (2002). A design optimization tool based on a genetic algorithm. *Automation in Construction, 11* (2), 173-184.

Caldas, L. (2001). An evolution-based generative design system: using adaptation to shape architectural form. Doctoral Dissertation, Massachusetts Institute of Technology. Dept. of Architecture. Retrieved, 04.01.2010, http://hdl.handle.net/1721.1/8188.

Chien, S. F., (2007). Growing a City: Individuals, Interactions, and Emergent Behaviour. *Proceedings of 10th Generative Art Conference GA2007.* Milan.

Chu, K. (2006). Metaphysics of genetic architecture and computation. *Architectural Design. 76* (4), 38-45.

De Felice, F., Abbattista, F., & Scagliola, F., (2002). GenOrchestra: An Interactive Evolutionary Agent for Musical Composition. *Proceedings of Generative Art Conference 2002*. Milan.

Durmazoğlu, M. Ç., & Çağdaş, G., (2007). An Agent System Based Tool to Help Designers for Free- Form Shape Generation in Digital Media. *Proceedings of 10th Generative Art Conference GA2007*. Milan. (pp. 166-177). Retrieved 07.08.2010, http://www.generativeart.com/on/cic/papersGA2007/17.pdf

Eiben, A. E. & Smith, J. E., (2003). *Introduction to Evolutionary Computing*. Springer: New York.

Eleni, P., Turner, A., & Thum, R. (2002). Interacting unities: an agent-based system. *Proceedings of Generative Art Conference 2002*. Milan.

Fasoulaki, E. (2007). Genetic Algorithms in Architecture: a Necessity or a Trend? *Proceedings of Generative Art Conference 2002*. Milan.

Frazer, J. (1995). *An Evolutionary Architecture.* London: AA Publications.

Gero, J. S. (2003). Design tools as situated agents that adapt to their use. In W. Dokonal and U. Hirschberg (eds), *eCAADe21*, eCAADe, Graz University of Technology. (pp. 177-180).

Gero, J. S. & Peng, W. (2004). A situated agent-based design assistant. In *Conference Proceedings of CAADRIA 2004*, Yonsei University Press, Korea, (pp. 145-157).

Gilat, D. (2005). Autonomic Computing – Building Self-managing Computing Systems. In G. Flachbart, & P. Weibel, *Disappearing Architecture* (pp. 32 - 40). Basel: Birkhäuser.

Grobman, Y., Yezioro, A., & Capeluto, I. (2009). Computer-Based Form Generation in Architectural Design-a Critical Review. *International Journal of Architectural Computation , 7*, 535-553.

Gu, N., & Maher, M. (2005). Dynamic designs of 3D virtual worlds using generative design agents. *Computer Aided Architectural Design Futures 2005* , 239-248.

Gürer, E., & Çağdaş, G. (2006a). A Multi-Level Fusion of Evolutionary Design Processes. In V. Bourdakis & D. Charitos (eds) *eCAADe 2006* Communicating Space(s). (pp. 904-907). Volos: Greece

Gürer, E. & Çağdaş, G. (2006b). Nature inspired approach: An emergent form generation. In C. Soddu, *GA 2006 9th International Conference on Generative Art* (pp. 75 - 80). Milan.

Harley, J. (1995). Generative Processes in Algorithmic Composition: Chaos and Music. *Leonardo, 28* (3), 221-224.

Hemberg, M. et al., (2007). Genr8: Architects' Experience with an Emergent Design Tool. In J. Romero, & P. Machado, *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music* (pp. 167 - 188). Berlin Heidelberg: Springer.

Herr, C., & Kvan, T. (2007). Adapting cellular automata to support the architectural design process. *Automation in Construction , 16*, 61-69.

Holland, J. (1995). *Hidden Order: How Adaptation Builds Complexity.* MA: Helix Books.

Jacob, C. (2003). Dancing with Swarms: Utilizing Swarm Intelligence to Build, Investigate, and Control Complex Systems. In P. F. Hingston, L. C. Barone, & Z. Michalewicz, *Design by Evolution* (pp. 69-94). Berlin: Springer.

Johnson, S. (2001). *Emergence, the Connected Lives of Ants, Brains, Cities and Software*. UK: Penguin Press.

Kimura, S. (2008). Inference of Genetic Networks Using an Evolutionary Algorithm. In P. F. Hingston, L. C. Barone, & Z. Michalewicz, *Design by Evolution* (pp. 31-51). Berlin: Springer.

Krawczyk, R. (2002). Architectural interpretation of cellular automata. *The 5th International Conference on Generative Art*, 7.1-7.8.

Leach, N. (2009). Swarm Urbanism. *Architectural Design*, 56-63.

Lynn, G. (1999). Animate Form. New York: Princeton Architectural Press.

Maher, M. L., & Gero, J. S. (2002). Agent Models of Virtual Worlds. In *Proceedings of ACADIA 2002.* (pp.127-138). California State Polytechnic University.

Maher, M. L., Smith, G. J. & Gero, J. S. (2003). Design agents in 3D virtual worlds. In R. Sun, *IJCAI Workshop on Cognitive Modelling of Agents and Multi-Agent Interactions*, (pp. 92-100). Acapulco, Mexico.

McCormack, J., Dorin, A. & Innocent, T. (2004). Generative Design: a paradigm for design research. In Redmond, J. et al. (eds.) *Proceedings of Futureground*, Design Research Society, Melbourne.

Merkle, D., Middendorf, M., & Scheidler, A. (2008) Organic Computing and Swarm Intelligence. In C. Blum, & D. Merkle, *Swarm Intelligence* (pp. 253-281). Berlin: Springer.

Narahara, T. (2009). Bottom-up Design Inspired by Evolutionary Dynamics Adaptable Growth Model for Architecture. In G. Çağdaş & Çolakoğlu B. (eds) *eCAADe 2009 Computation: the new realm of architectural design* (pp. 391-398). Istanbul.

Oxman, R., & Oxman, R. (1992). Refinement and adaptation in design cognition. *Design Studies , 13* (2), 117-134.

Peng, W., & Gero, J. (2007). Computer-aided design tools that adapt. *Computer-Aided Architectural Design Futures (CAADFutures) 2007* (pp. 417-430). Berlin: Springer.

Renner, G., & Ekart, A. (2003). Genetic algorithms in computer aided design. *Computer-Aided Design, 35* (8), (pp. 709-726).

Rheiner, M., & Eggmann, F. (2005). Generative Design. In G. M. Buurman, *Total Interaction: Theory and Practice of a New Paradigm for. the Design Disciplines.* (pp. 251-273). Basel: Birkhäuser.

Rocker, I. (2006). When Code Matters. *Architectural Design*, 76(4). (pp.16-25).

Rowe, P. (1987). *Design Thinking.* MA: The MIT Press.

Schork T., Burrow A. & Minifie P. (2009). CloudNets: A Workbench for Emergent Urbanism and Architectural Form. . In G. Çağdaş & Çolakoğlu B. (eds) *eCAADe 2009 Computation: the new realm of architectural design* (pp. 417-423). Istanbul.

Seidel, T., Hartwig, J., Sanders, R., & Helbing, D. (2008). An Agent-Based Approach to Self-organized Production. In C. Blum, & D. Merkle, *Swarm Intelligence* (pp. 219-252). Berlin: Springer.

Shea, K., Aish, R., & Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. *Automation in Construction, 14* (2), 253-264.

Simon, H. A. (1983). Why should machine learn. In *Machine Learning: An Artificial Intelligence Approach.* Berlin: Springer. (pp. 25-37).

Simon, H. A. (1996 [1968]). *The Sciences of the Artificial. 3rd Edition.* Cambridge, MA: The MIT Press.

Sönmez, O. N. & Erdem, A. (2009). Design Games as a Framework for Design and Corresponding System of Design Games. In G. Çağdaş & Çolakoğlu B. (eds) *eCAADe 2009 Computation: the new realm of architectural design* (pp. 119-126). Istanbul.

Sterritt, R. & Hinchey, M. G. (2005). From Here to Autonomicity: Self-Managing Agents and the Biological Metaphors that Inspire Them. In Proceedings International Conference on Design and Process Technology (IDPT 2005). Beijing, China.

Terzidis, K. (2006). Algorithmic Architecture. Oxford: Architectural Press.

Testa, P., O'Reilly, U., Kangas, M., & Kilian, A. (2000). MoSS: Morphogenetic Surface Structure A Software Tool for Design Exploration. 1-11.

Testa, P., O Reilly, U., Weiser, D., & Ross, I. (2001). Emergent Design: a crosscutting research program and design curriculum integrating architecture and artificial intelligence. *Environment and Planning B, 28* (4), 481-498.

Torrens, P. (2000). *How Cellular Models of Urban Systems Work (1. Theory).* London: UCL.

Torrens, P., & O'Sullivan, D. (2001). Editorial: Cellular automata and urban simulation: where do we go from here? *Environment and Planning B: Planning and Design , 28*, 163-168.

Trianni, V., Nolfi, S., & Dorigo, M. (2008). Evolution, self-organization and swarm robotics. In C. Blum, & D. Merkle, *Swarm Intelligence* (pp. 163-191). Berlin: Springer.

Wolfram, S. (2002). *A New Kind Of Science.* Canada: Wolfram Media.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and Practice, in Knowledge Engineering Review 10(2): 115-152. Retrieved 07.08.2010, http://webcache.googleusercontent.com/search?q=cache:zxrzdwvS0PkJ:www.csc.liv.ac.uk/~mjw/pubs/ker95.pdf+%22intelligent+agents:+theory+and+practice%22&cd=1&hl=en&ct=clnk

Zellner, P. (1999). *Hybrid space: new forms in digital architecture.* London: Thames & Hudson.