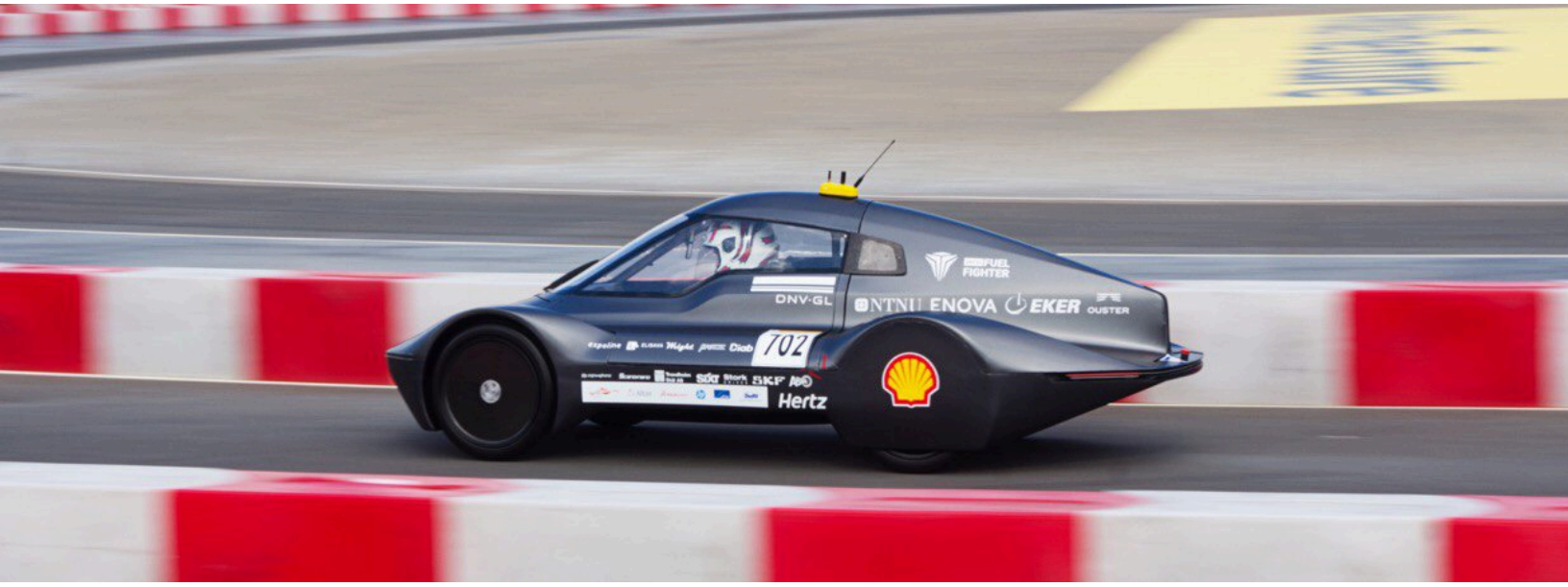
NTNU
Norwegian University of
Science and Technology

GENERATIVE DESIGN IN PRODUCT DEVELOPMENT



Creating an Urban Concept Car for Shell eco-Marathon

Short:
Investigation and shaping of a generative design methodology for product development, implemented in the design and construction of an energy efficient urban concept car participating in Shell Eco-Marathon.

Author: Eirik Evjan Furuholmen
Supervisors: Christer Elverum, Cesilia Haskins

Abstract

This thesis is an investigation and shaping of a generative design methodology for product development, implemented in the design and construction of an energy efficient UrbanConcept car for participation in Shell Eco-Marathon.
No studies was found to present general strategy for implementing generative design into product development. Thus, this thesis is presenting such a strategy base on similarities drawn between human problem-solving, design theories and methodologies and genetic algorithms – one of the major technologies for generative systems.
The aim of the implementation in the development of the vehicle is both to test the methodology on a real product development case, as well as creating a competitive vehicle for the competition. This was conducted through the development of the cars monocoque. The implementation led to very satisfying results in the development, and good results in the competition.

Sammendrag

Denne oppgaven er en undersøkelse og utforming av en generativ designmetodikk for produktutvikling, implementert i prosjektering og konstruksjon av en energieffektiv UrbanConcept bil for deltakelse i Shell Eco-Marathon.
Ingen studier ble funnet å presentere en generell strategi for implementering av generativ design i produktutvikling. Dermed presenterer denne avhandlingen en slik strategibase basert på likheter trukket mellom menneskelig problemløsning, designteorier og metodologier og genetiske algoritmer - en av de viktigste teknologiene i generative systemer.
Målet med implementeringen i utviklingen av kjøretøyet er både å teste metodikken i et ekte produktutviklings scenario, samt å skape et kjøretøy som kunne nå langt i konkurransen. Dette ble utført for utviklingen av bilen monocoque. Implementeringen førte til svært tilfredsstillende resultater i utviklingen, og gode resultater i konkurransen.

Acknowledgements

First and foremost, I want to thank my supervisors Christer Elverum and Cecilia Haskins, for their incredible positive attitude and helpful detections towards the thesis. Secondly, I want to give a huge thanks to all the member the DNV GL Fuel Fighter team over the last two years that have both contributed to this thesis, as well as made the last years at NTNU a great experience. NTNU itself ant the staff that in way have helped in the development have also been highly appreciated, and a special thanks to Knut Aasland, Børge Holen and Natalia Trotsenko.
A special thanks also goes to the other master students of the FF18/19 team; Lars Ramstad, Kristoffer Sydnes and David Swensen, which had huge contributions to the thesis and the work in DNV GL Fuel Fighter, as well as Sindre Trefall for his work with fluid dynamics in the last year. Jennifer Nugyen also deserves ha huge thanks for her effort as the deputy leader of the project, helping with administrative management and all economics of the project.

Content

1 Introduction

The student team DNV GL Fuel Fighter (FF) from NTNU has been developing and building cars participating in the international engineering competition Shell Eco-Marathon Europe (SEM) since 2008, which is the largest of several student competitions hosted by Shell all over the world. Every year FF and about 2000 other students from teams around the world are gathered in this competition to test their cars up against each other, with the ultimate goal of creating the most energy-efficient vehicle.

Figure 1: Opening ceremony of the Shell eco-Marathon 2019, with all the participants and a few of the 110 cars competing.

The competition has two types of vehicles; Prototype, where the drivers lie horizontally in tiny vehicles usually constructed with three wheels, and the larger 4 wheel UrbanConcept class that has more resemblance with road-legal vehicles for the consumer market. In the latter class, the race is constructed to simulate city driving, consisting of 15 laps to be completed within 35 minutes, each with a length of 970 meters where the car has to come to a full stop each lap (SEM18). The two vehicle types are also divided into three classes of energy source, ranging from Battery-Electric and Hydrogen to different fuels for Internal Combustion. As the classes are highly different in terms of energy consumption, winners are awarded within each energy class, while off-track awards (such as the Innovation Award and the Design Award) comprise either all, or each of the two vehicle types. In addition, a new competition called *UrbanConcept Autonomous* was introduced in 2018. Here, UrbanConcept vehicles can attend to prove their capability of autonomous driving through a range of tests, giving the competing teams points to determine the winner.

Over the years, 4 UrbanConcept cars and 1 Prototype car have been developed and built by the DNV GL Fuel Fighter team, with several master students being part of the project. Most of the cars are improved on for 2 to 3 years to bring out the best potential of the design before a new car is built. Some years have been highly successful in terms of results, with a first place in the hydrogen class in 2009 and second place in the Battery-Electric class 2018, as well as winning and pall positions for both Vehicle Design and Communications over the years. Other times however, the team has been troubled with technical issues under the competition or been unable to compete at all, not making it through the rigors technical inspection to be allowed onto the track.

Figure 2: The cars developed by DNV GL Fuel Fighter since its beginning in 2008.

The author has been part of the organization for the last two years of his master study, the first year as the leader of a R&D team researching new technologies and planning the development of the 5th UrbanConcept vehicle. Last year the author has been managing the DNV GL Fuel Fighter team, with 40 engineering students developing and builting the DNV GL Fuel Fighter 5. To the author being part of the organization has been an immensely educational and exciting experience, solving complex engineering problems and cooperating with other passionate and skillful students from different countries and study directions. It is a place where the engineering theory acquired from courses over the years can be applied to a real-world problem, putting theory into practice. It is also a place where team effort is essential to succeed, creating friendships as well as interpersonal skills of future value.

1.1 Background and motivation

When radical new technologies for aiding product development emerge, it can have a huge effect on how humans innovate. The tools engineers utilize can greatly amplify their ability to transform ideas into products, and also change how this process is conducted.
Tools that are involved in product development range from mechanical tools such as milling machines and assembly lines, computer-based tools such as FEM software and programming to more organizational directed tools such as Enterprise Resource Planning and Product Lifecycle Management. These have all influenced the way engineers innovate products, and the companies that adopt the latest technology often gain an advantage in their industry.
Tools that are made to aid the actual product development process itself have had an even more direct impact on how problems are solved. CAD software was originally created to improve and automate paper-based and manual tasks, but has shown to be much more influential. The possibility of enabling quick exploration and visualization have completely changed the culture and process and of design (Brown 2009). The development of simulation software has in a similar manner enabled engineers to create increasingly advanced products. However, continuously advancing modeling and simulation systems do not only change our abilities and the manner in which we create products, but also human problem solving itself (Becker et al. 2005).

Generative Design (GD) is an emerging technology that by many is predicted to create a new paradigm for engineering design and problem-solving in a wide range of fields ((Türkmenoğlu 2015), (McCormack et al. 2004), (Janssen et al. 2002)). With the use of Generative software systems, solutions to problems can be generated, explored and optimized in a manner that resembles natural evolution. The computer becomes a design generator, gaining “creative like” abilities compared to its normal passive roles as a performance analyst, visualizer, data checker and drafts tool (Shea et al. 2005). The utility of these systems is then to automatically generate and optimize solutions to problems more rapidly than previous techniques, and in some cases creating solutions that would be impossible to come up with using former engineering tools and human ingenuity.
While there is no agreed-upon definition of the term yet, as several software companies and disciplines disagree which technologies and methods it embodies, it seems to be a general consensus that it is a *design method where generation of form is based on rules or algorithms* (Agkathidis 2015). Defining the term is also difficult as it is having a transformative impact on such a wide specter of areas and industries, from art and architecture to a range of engineering disciplines.
In essence, generative systems is just a category of optimization software, and the field of optimization has been part of engineering for decades. However, generative design brings a big shift in the common conception of optimization; form techniques for fine tuning of objects or systems in the final stages of development, to an approach for concept exploration and optimization from the very beginning and throughout the development process. This shift can be attributed to the algorithms ability to explore and determine one or several near optimal solutions in a vast solution space, rather than finding the single global optimum to a well-defined problem.

Janssen et al. (2002) describe generative and evolutionary systems as an emerging third phase of computer design tools. In their paper, the ability to enhance the capacity of visualization trough CAD is depicted as the first phase of computer tools, while simulation software enabling analysis to quantify performance of design are considered as the second phase. Although both of these technologies have changed the way problems are solved, they are considered to be passive in the manner they are altering the process. This means they substitute manual tasks and aid problem solving without changing the actual structure of development to any large degree. Hence, the design methodologies from the pre-computer area are still highly valuable for innovation with these tools implemented in the process.
This third phase of generative and evolutionary tools however, are considered to not only aid and/or replace manual tasks, but also cognitive parts of human innovation. It might therefore change the structure of the design process, requiring a different approach to design to be effectively implemented. Some, such as Janssen et al. (2002), predict it will demand a change from having the designer at the core of the problem solving process, to methodologies where the generative system is the central guiding principal . Others see the change as going from an approach where humans solve problems using tools, to that of constructing algorithms to which problems are specified (Nordin 2018), in a sense, *moving from solving problems to growing solutions.*

Although much of the technology behind generative systems have been around for some time, such as topology optimization (TO) and genetic algorithms (GA), its use in product development has not been significant.
This can be ascribed to the huge processing power needed to solve advanced problems, as well as the difficulty of representing complex problems in algorithmic terms. However, with the increasing access to computer power and more versatile software, these systems have had a boom over the last years. Another aspect is the advancements and affordability of 3D printing, which in some cases can be the only way to realize some of the designs created with generative systems. Several of the largest CAD and FEM software companies such as Dassault Systems, Siemens and Autodesk are therefore heavily invested in developing competent systems at the time of writing, and industries like aerospace, aviation and automotive are in the forefront of implementation.

The sheer power that Generative Design can have in solving engineering problems, and their transformative potential for problem solving in general spiked the author's first interest in the field while researching possible technological applications for DNV GL Fuel Fighter. In a competition like Shell eco-Marathon, adopting new technology in a clever way might create an advantage over other competitors, enabling the team to push the boundaries in areas like weight and aerodynamics that are essential for energy efficiency. Implementing a generative design approach to advance the development of the DNV GL Fuel Fighter 5 thus became the initial ambition for this thesis.

However, it is a long way from an initial product idea to plotting values and constraints into a tailored algorithm for optimization, or using generative systems on a detailed CAD model.

At the very beginning of a product development process, there is often simply nothing to optimize yet. As a result of this gap, and the notion that these systems could demand an untraditional approach, the subsequent question was clear; how can Generative Systems be implemented successfully from the very beginning of the development, and what methodology can aid the process?
Researching the literature on this field exposed that although many people predict the technology to have a huge impact, very few articles have been written on the actual change it inflicts on product development, and what methodologies are needed for successful implementation in general cases. The large part of articles written on generative design, whether about product development or other areas, are focused on its utility in specific cases or the development of specific algorithms.
This gap in the research field evolved to become the second ambition of this thesis; approaching a generative design methodology for product development in general, which facilitates the implementation of generative systems when the opportunities of utilization arises throughout the process.

With the author's limited knowledge of the subject to begin with, and at NTNU in general, the ambitions of this thesis has been difficult to realize. However, the possibility of both contributing new ideas to such a promising field as Generative Design, and advancing the development of DNV GL Fuel Figher 5 was predominant in choosing the topic. Being part of FF has also given a rare opportunity that was critical for these goals, being able to both explore technology with little risk, and investigate the implementation in a real product development case.

1.2 Project scope

1.2.1 Problem description and objectives

With the notion that GD Systems will have a huge impact on the way engineering design is conducted, this thesis attempts to find methodologies and tools to create a framework that can aid the implementation of generative design in product development, creating an optimization driven design process. As few studies was found to present a general strategy for implementing generative design into product development, the thesis is attempting to develop its own methodology.
Further, this process is applied to the real product development case of creating an urban concept car for participation in Shell Eco-Marathon. The aim of the implementation in the development of the vehicle is both to test the methodology on a real case, as well as creating a competitive vehicle for the competition. This is to understand generative systems hands on, and utilize this to understand the possibilities and limitations for implementing the technology. This is conducted through the implementation of the methodology and generative systems in the development of the vehicle monocoque, witch constitutes the complete body of the car and its internal structures, in addition to doors, hood and and rear hatch. The implementation

is thus not just a case study for the methodology, as the success of the vehicle development is regarded to be of high importance to the author and the DNV GL Fuel Fighter Team. Thus, the thesis is in short a study of the interplay between strategy and development.

1.2.2 Research questions

From approaching such a methodology, and implementing the generative systems to enhance the development of DNV GL Fuel Fighter, three research questions are asked in this thesis;

- *How has generative design been implemented in product development before?*
- *What impact does Generative Design as a methodology have on the process of engineering design and problem solving?*
- *Can the implementation of generative design create a competitive advantage in the development of DNV GL Fuel Fighter 5?*

The relation to the work are of these are discussed in the final chapter. However, to the reader, it is important to have this questions in mind when reading the thesis, as they in large are answered throughout the work and not its conclusion.

1.2.3 Limitations

There are several limitations to the scope of this project. As FF is a self organized volunteer project, with students from different classes for the most part are part of the project for a single year (although some staying only a half, while others several years), the continuity is a huge limitation. Most of the students working with the beginning of the development of this car in the first year was shifted with new members the following year. This means that for every new year the learning curve is exceptionally large if one wants to develop a competitive vehicle. The complexity of the project also limits the implementation of generative systems, as the advanced methods needs more effort than what would be required from the average volunteer. The implementation is therefore concentrated at the monocoque development, that two other master students in the project also was focused on.
The time is also a huge limitation. Planning the development the first year with a small sub group of the team, and then constructing and building a whole car from scratch in the matter the next year is highly difficult task, especially with the problem of continuity. This means that the level of advanced methodes has to be applicable to the timeframe, and implementing generative systems on all structural parts would be unreasonable.
Another limitation is that of verification of the process, especially for implementation in product development in general. Processes are hard to verify, as a setup of two identical projects is needed, and even then, the knowledge of the people involved might interfere with results. Building two cars side by side using different approaches is obviously out of the scope,

thus, the verification of the models effectiveness had to be drawn from the results of the development.

1.3 Author's role in the development

As the author has led the first years R&D team as well as the whole team as project manager the following year, it is natural that the thesis tis written from a strategic point of view. All the decisions and work presented is therefore influenced by a several team members over the two years, and defining all contributions throughout the thesis would be difficult, especially since much of the work and decisions are team efforts. As the thesis is written in passive form, it is therefor important to state the work that was done solely by the author, and the work done solely by others. The development of the methodology which will be presented has been the independent work of the author, while the development of the monocoque was a joint effort. The authors main contribution to the monocoque has been the development of vehicles design and all CAD models throughout the project, relating to all dimensional requirements by Shell and other parts, as well as structural and aerodynamic changes. The author was also responsible for the strategy of utilizing different generative design approaches and the overall development strategy, as well as contribution to aerodynamic solutions, and interpreting the results to generate new CAD models. All the optimization using the Hyperworks tool was done by fellow master students Kristoffer Sydnes and David Swensen, and there work was essential to the generative strategy implemented. All aerodynamic simulations was also done by other members, with Sindre Trefall as major contributor in the final year of development utilizing OpenFoam. Many of the figures presented is therefor showing this joint effort, with CAD representations created by the author, and simulations done by other team members.

1.4 Method and thesis structure

This thesis is approached with the following method; first a literature review is done to give the reader an overview of the field of generative design. This is done to explain the basic operators of generative design and why they are an interesting analogy for innovation, as well as the applications limitations and future of generative design. Further, existing theories and mereologies are discussed.
Then, a comprehensive study of the similarities between genetic algorithms, human problem solving and design theories and methodologies are conducted to approach a methodology for implementing generative design into product development.
Finally, an overview the development of the monocoque is presented with the emphasis to give the reader an understanding of how the methodology was implemented and changed the overall development process.
Finally the results and the work done is discussed with the aim to answer the research questions.
Direct citations are used more often than paraphrasing in this paper, especially when comparing GA to DTM and human problem solving. This is because the author wants to ensure the reader that opinions and similarities are not skewed. When the basis for creating the theory is finding similarities, using paraphrasing would make the process less valuable. Comparing one

statement in one theory to another in another theory trough paraphrasing, one could easily make it seem that things fit well together. Direct citations are then a more honest way of comparison.

2 Theory

2.1 Background on Generative Design

Although the term GD have been used in the field of architecture since the 1970s, it is rather new term in relation to product development (Nordin 2018b). Despite being used in architecture for such a long time, there is no clear definition of what GD is, and different fields have different viewpoints. In architecture, the term is often used on par with Parametric Design, while in engineering it is often used on par with topology optimization. Other, like the major CAD software developer Autodesk, see both these technologies as prerequisites of GD, blurring the lines of how to define it. The Wikipedia page on GD lists two definitions, but both are very vague and the page itself has multiple issues and contradictions in trying to describe GD accurately;

"Generative Design is a morphogenetic process using algorithms structured as non-linear systems for endless unique and unrepeatable results performed by an idea-code, as in Nature." - Celestino Soddu, 1992. (1)

"Generative Design is the transformation of computational energy into creative exploration energy empowering human designers to explore greater number of design possibilities within modifiable constrains." - Sivam Krish, 2013. (1)

Some companies that are in the forefront of developing Generative Systems (GS) such as Autodesk`s Dreamcatcher Project, are using the term as core in their marketing, while other large companies such as Dassault Systems never use the term in the description of similar software solutions like Isight. Autodesk define GD as the process of using systems that *mimics nature's evolutionary approach to design (2)*. They describe how the process works by explaining that *designers or engineers input design goals into generative design software, along with parameters such as materials, manufacturing methods, and cost constraints. Then, using cloud computing, the software explores all the possible permutations of a solution, quickly generating design alternatives (2).*
Although this description gives insight to what GD can do, it does not define what it actually is. The author will use a broad definition stated by Fernandes and Margarida (2003); they describe a GS as *a system that generates options for design problems,* and that *the basic system in all Generative Systems is Algorithmic Systems.* GD is then the practice of mediating the design process by using GSs, where *various potential design solutions can be created determined by algorithms*. In that sense, Parametric Design is a simple type of GD, as it is based on the use of

hierarchical algorithmic systems controlled by one-directional relationships (Fernandes and Margarida 2003).
While Topology Optimization has been around for about 20 years and used more advanced algorithms in the same manner as GD to optimize design problems, the focus has been, as the name states, that of optimizing topology. Although this is a major part of GD, the definition is not limited to this, but holds all qualities that could be optimized in a design. This would then mean generating a design solutions or mediating parts of the design process for any quality, like the aesthetic coloring of an object, the conductivity of an alloy or the optimization of a plastic molding process. GD is also not limited to optimizing problems, which is the focus of topology optimization, but also constitutes the ability to generate multiple design options that can be evaluated by the designer. In that way, GD is not the closed process within GSs, but the use of GSs. This means that humans can replace or participate in parts of the algorithm used in the GS, for instance choosing between multiple aesthetic propositions for a design created by the algorithm. This creates a shift in how design is conducted; instead of designing an object we want, we design or use an algorithm that designs the object we want.

Before going into details of the utilization of GD, it is necessary to explain what sort of algorithms are that are commonly used. Since algorithms are simply the use of sequential instructions for solving a problem, all programs can be considered algorithmic. It is important then to state the class of algorithms that can be considered to *generate options for design problems*. Where one draws the line of which algorithms to consider as simply problem-solving methods and which to consider solution generating methods is not straightforward. However, one could say that some algorithms are created in order to solve problems exactly, while others are created simply to search for good solutions. It is the latter that mainly is utilized when using GSs, as the problems they are applied to is simply too hard to solve with precise methods. This does not mean that the algorithm never find optimal solutions, but that the means to solve problems more resembles a child's search for solutions, rather than the procedure of a mathematician - although a very efficient child. For hard optimization problems, approaching the problem by searching for a near optimal solution can be the only way to efficiently finding a solution at all.

The algorithms often used to solve these kind of problems is a class called *metaheuristics*, that can find solutions without being constructed directly solve the problem at hand, in contrast to problem-specific heuristics. Although not completely restricted to metaheuristics, GSs is then in large user oriented software that utilize metaheuristic algorithms to optimize or generate multiple solutions for design problems. Boussaïd et al. (2013) states that *all metaheuristics share the following characteristics:*

- *They are nature-inspired (based on some principles from physics, biology or ethology.?*
- *They make use of stochastic components (involving random variables).*
- *They do not use the gradient or Hessian matrix of the objective function.*
- *They have several parameters that need to be fitted to the problem at hand.*

There exist a wide range of metaheuristics that suites different kind of optimization and solution generating problems, and going through all is out of scope for this paper. Some of the popular classes are Shape Grammar, Lindenmayer Systems, Particle Swarm, Cellular Automata and Evolutionary Algorithms. The latter are further divided into four main types: Genetic Algorithms (GA) Evolutionary Programming (EP), Evolutionary Strategies (GS) and Genetic Programming (GP). Of these, the GA (GA) has become the most popular method for solving hard optimization problems as it has the widest applicability (Gábor and Ekártab 2003). It is in a sense the allrounder of the metaheuristics, and is particularly fit for generating form or styles (Singh and Gu 2012).
In comparing GA to other types of metaheuristics, Gábor and Ekártab (2003) explains that the reason for GA being especially suited for optimization problems is *that while other methods always process single points in the search space, genetic algorithms maintain a population of potential solutions*. They also point out the flexibility of GA is also due to its ability to solve complex problems by handling multiple parameters simultaneously. GA can therefore be said to be the most flexible and practical algorithm of for GD. Although the different metaheuristics have very different ways of generating solutions, explaining how GA works will give the reader a better understanding of how GD generates solutions, and will thus be elaborated in the next section. However, the motive for describing how GA works is ultimately to clarify how the intuitions of GA may help in the implementation of GD in product development. The section will therefore only explain the basic of an ordinary GA to give the reader a ground for understanding, as more specific details will be elaborated in conjunction with the development of the generative design methodology in section 4.

2.1.1 The basics of Genetic Algorithms

Generative Algorithms was introduced by John Holland in the 1970s with the idea of mimicking the natural selection in evolution. It consists of operators that imitate genetics and Darwin's principles of how life evolve, and although the algorithmic interpretation of these principals is highly simplified compared to nature itself creating an almost rudimental algorithm, the GA often generates unexpectedly complex solutions to difficult problems (Gábor and Ekártab 2003). Even with these simple operators, the algorithm can mimic evolution like abilities, such as making a physics based 3D model of a human run completely life like without any knowledge of what running is.
The encoding of problems into GA use much of the same terminology as the DNA; decision variables are encoded as finite-length strings consisting of certain alphabets with a defined cardinality that outline the constraints of a population of potential solutions to a problem. The distinct strings within this cardinality is called chromosomes, which are the representations of each individual solutions to the problem. The shorter alphabets that the chromosome consists of are called genes, which are distinct parts that are describing certain areas of a solution. Alleles are the smallest descriptions of the problem, which describe the values of each gene.

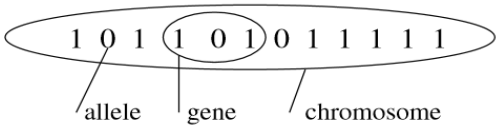


Figure 3: Representation of a problem, here as a simple bit-string

The representation of a problem and potential solutions is often described by bit-strings, but as problems grow more complex, many other representations can be applied, like parameter lists, tree structures or even complex programs (Sastry et al. 2014). For instance, the chromosome could be the representation of a microwave oven consisting of several parts; each gene could represent each part of the assembly, while the allele describes the measurements of each part. The cardinality or constraints of the problem representation creates the framework for how the product can be assembled and the region of values allowed, defining the solution space of the problem and thereby the possible microwave ovens that can exist in the population.
The GA differentiates between the genotype space; which is where all representations of solution are described, and the phenotype space; where the actual solutions are located. Different combinations of genes and alleles map onto the phenotype space to create actual solutions. Relating to the microwave oven, the genotype space would describe all the values and interactions of parts of a solution, and is thus the knowledge that describes the product. The phenotype space is the solution itself, and could for instance be represented as a CAD model. Further, to measure the fitness of a solution to apply the artificial natural selection of different solutions, the algorithm needs a means of evaluating solutions with some fitness criteria, determining a fitness space. This is often done with an objective function such mathematical models or the use of FEM analysis or other simulation tools, but could also be the subjective opinion of humans. Potential solutions in the phenotype space then maps onto the fitness space to test how good the proposed solutions are, producing information of how to evolve the next generation of the population of solutions.

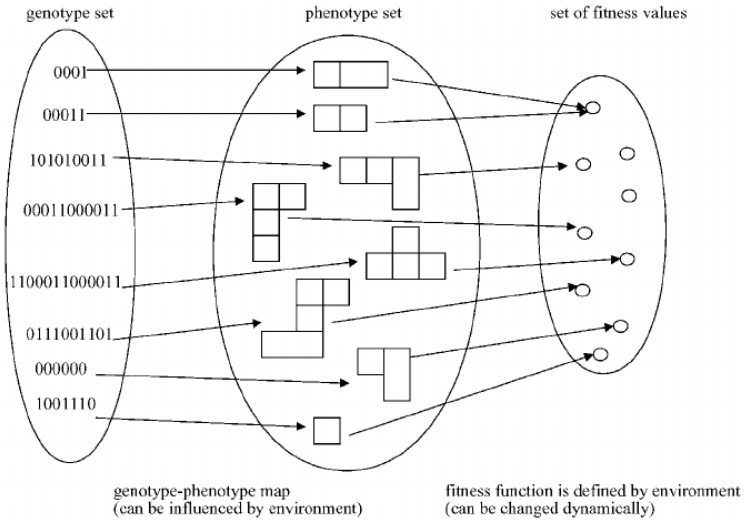


Figure 4: The GA mapping between each space. https://www.researchgate.net/figure/Genotype-phenotype-mapping-and-fitness-function-in-evolutionary-design_fig3_226912968

The GA files through an iteration process, where new generations of candidate solutions are produced continuously and added to the population, creating better solutions without any specific strategy other than measuring its own evolutionary progress towards the goal (Sastry et al. 2014):

1. Initialization: An initial population of solutions is created randomly or with some knowledge of the domain.
2. Evaluation: The initial population or offspring population is evaluated against the fitness criteria, such as objective functions, simulations or with human interaction.
3. Selection: Copies of solutions of higher fitness are reproduced to enforce a survival-of-the-fittest mechanism to the population. This can be done with several types of techniques, such as roulette-wheel selection, ranking selection, stochastic universal selection and tournament selection, giving preference to which solutions are considered good.
4. Recombination: Two or more solutions are selected as parents and combined with some type of crossover operator, to create new offspring of solutions with different traits from each parent solution.
5. Mutation: Single solutions are selected and small changes are made, adjusting the solution in its vicinity with small random changes in the solution.
6. Replacement: The offspring solutions created by step 3-5 are put back into the population, and different methods such as elitist-, steady-state- or generation-wise replacement are applied to decrease the size of the population.
7. Repeat: the steps from 2–6 are iterated until a defined termination criterion is met.

These steps explain the very basic of how the GA operates, but there exists numerous variants and advanced versions of the algorithm, like Multi objective-, Steady-State-, Distributed-, Parallel-, Messy-, Hybrid-, Structured- and injection island GA just to name a few. A search of GA in Google Scholar gives over 2 million results, which gives a clue to how huge this field is, not to mention the field of metaheuristics in general.
Further, advanced GSs often not only use a single algorithm, but can be utilizing combinations of algorithms. The term Hyper-Heuristics is used as the ability of *selecting, combining, generating or adapting several simpler heuristics* (3), making the system adaptive to different problems. In the table underneath, the basic operators of natural selection and GA is put side by side, clearly showing the inspiration in GA from nature's way of solving problems.

	Natural selection	Optimization algorithm
Goal	Survival	Minimal objective
Initialization	Wild population	Generated individuals
New solution	Variation	Mutation
	Differential reproduction	Crossover
	Heredity	Selection
Fitness	Potential to survival	Objective function
Procedure	Natural choice	Evolving
Stopping condition	None	Convergence or max iter

Figure 3: Similarity of GA an natural evolution.

2.1.2 Applications of Generative Design

There almost seem to be no boundary to the application of metaheuristics for difficult problems. A Wikipedia page on the applications of GAs alone lists almost 80 different topics (4), ranging from airlines revenue management and mechanical engineering to computational chemistry and the training of artificial neural networks. Gábor and Ekártab (2003) states that the branches of mechanical engineering where GA is most predominantly used are *conceptual design, shape optimization, data fitting, reverse engineering, mechanism design and robot path design.*

However, making GD useful is about utilizing these metaheuristics trough Generative systems that can solve a range of problems, instead of building specific algorithms for specific problems. These systems therefore often contain multiple readymade algorithms that can be applied on a higher level, making them easier to employ on a range of problems. They are increasingly used tools in architecture for applications such as floor facility layouts and beam structure optimization, with programs like Rhino and Grasshopper. In the domain of mechanical engineering, topology optimization based on load has been the widest use of GSs (also employing other methods than metaheuristics such as Solid Isotropic Material with Penalization (SIMP)), but also other types like sizing, shape, topography, and mesh structure optimization are common. There exist multiple programs for these specific applications, and larger software companies increasingly have them built into their CAD software as modules, such as Fusion 360, Tosca Structure and Siemens NX. Increasingly advanced applications have emerged recently and are under development, such as fluid optimization tools like Tosca Fluid and design optimization for manufacturability and assembly such as Frustum's software Generate. Isight, a workbench program from Dassault Systems allows users to build customized iterations between optimization algorithms and programs such as Ansys Fluid, Catia, Abacus and a range of other programs. Even graphic design tools have emerged, such as Processing, making it possible to create art, typography and advanced infographics with the aid of algorithms.

Although there has been much hype around this type of systems, there is no doubt that they will influence the future of product development. The reason that these systems have become increasingly popular in recent time (even though optimization with metaheuristics and other algorithms have been around for decades) is without doubt related to the huge advancements in additive manufacturing and cloud computing. While additive manufacturing makes it possible to create the advanced shapes that often emerge from GD, the increasing power of cloud computing makes it possible to run processor expensive algorithms with more ease. This huge advancement in technology makes GD very interesting for the process of developing the next Fuel Fighter car. Gulanova and Vereš (2014), professors from the Institute of Transport Technology and Design at the Slovak University of Technology, states that *generative design as a new method of product development and innovation has become very important in the automotive industry at present. The reason for the introduction of this method into the process of development tasks solution is simple. This approach to car innovation leads to a reduction in the time spent developing prototypes.*

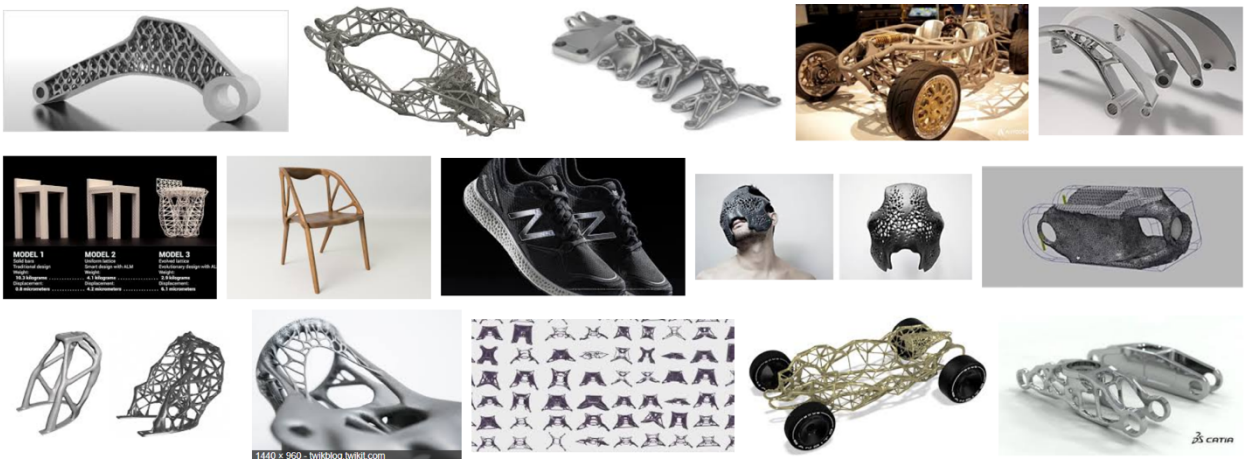


Figure 4: some of the applications presented by Autodesk.

2.1.3 Future of Generative Design

Another interesting area of computer-aided technologies to aid innovation of product development is a rather new type of tools known as CAI (computer-aided innovation). While the initial ideas in the beginnings of these programs was focused on aiding designers in early stages of product development, more integrated visions have emerged, with the goal of aiding the complete process from the fussy front end of innovation to successful products in the market (Leon 2009). There exist several tools that guide product development, but they are often separated, and used by different departments of an organization. The idea behind these CAI tools is to integrate commonly used tools throughout the process into a single platform that can better guide the innovation of a whole company. These could be organizational tools such as Product Life Cycle Management, Enterprise resource planning, Knowledge Management, CAx systems etc, but also problem-solving methods and product development strategies such as decision-making tools, system-engineering tools, TRIZ and Set-Based Concurrent Engineering

Systems. *Hence, it is not only a question of IT solutions, but also one of the in-depth focalization on the development of methodologies and concepts for supporting innovation teams more effectively and efficiently, thus improving the advantages of adopting new integrated CAI systems* (Leon 2009). Dassault System's 3Dexperience platform is an example of a step in the direction of these systems, with a single platform for social and collaborative apps (3DSWYM and Enovia), 3D modelling apps (Solidworks, Catia and Geovia), information intelligence apps (EXALEAD, NETWIBES) and content, simulation and GS apps (3DVIR, DELMIA and Simula).

With the wide applications of metaheuristics such as GAs, it is clear that these techniques can becom an essential part of CAI systems in the future, making it possible to optimize the product development process as a whole as well as its parts. Metaheuristics have already been proven useful to aid or optimize all kinds of tools and development methods that CAI systems consist of.
Leon (2009) claims that *it is expected that changes in innovation paradigms will occur through the use of computer-aided innovation methods and tools, and that new information technologies, such as Semantic Web, Text and Data Mining, chaos theory and Evolutionary Algorithms, will play an important role in the future of computer-aided innovation.* If this prediction is right, one could ask the question of what impact such tools will have on the way humans solve problems in the future? As the tools in many ways shape the way we innovate, adapting our product development processes to the tools would be necessary to utilize the its full potential.
Since GAs has shown to be the most flexible and applicable algorithm of the metaheuristics in several fields of product development as well as other areas, it might become central in optimization and generation of solutions in future CAI systems, which would be synonymous with advanced GSs. These systems would aid innovation and solve problems in a manner more like natural evolution. Hence, to utilize its potential, it is sensible that humans should adapt a problem-solving strategy that more resembles the tool, and consequentially natural evolution.

2.1.4 Why genetic algorithms is an interesting analogy for innovation

Humans have always been in pursuit of finding better ways of generating ideas and translating them into the real world to solve problems. The generation and translation that in novel ways creates value could be regarded as the innovative part of engineering. Even though engineers have become increasingly better at solving problems trough out the centuries, with a growing body of knowledge, more advanced tools and innovative methods - there is still an actor that in many ways is a far superior problem solver and innovator, namely nature itself. The question then is obvious; can we utilize nature's ingenuity in problem solving to benefit our own?
GAs was developed from this intuition, and as of now it is the best approximation of nature's way of solving problems that we have translated into computational methods. As described earlier, using GAs and other optimization techniques has led to novel solutions, so the answer to the question stated above would be yes. However, the broad view is that these ways of mimicking nature can be used as tools to help us solve problems, rather than ways in which to solve problems, or stated in another way; using nature's abilities rather than adopting them.

Another question then arises; can we become better innovators by solving problems in a *manner* that is more in line with nature?
There are numerous design theories and methodologies that try to describe and create ways to innovate and solve problems, but few draws an analogy to natural evolution. This could be because a connection between human and nature's way of innovating cannot be drawn, or that if it is possible, doing so does not create a better framework for solving problems than the existing methodologies based directly on human and organizational innovation. However, if there exist a resemblance between human and natural innovation, it should also be possible to draw analogy between human innovation and GAs, as GA mimic nature's most essential building blocks for solving problems.
This analogy between human innovation and GA was made by Goldberg in 1991, one of the most cited authors in the Evolutionary Algorithm community. From having used the metaphor of human innovation in his writing about GA from the 1980s, he made this connection explicit in his book *Genetic Algorithms as a Computational Theory of Conceptual Design*. In the book, he states that *few studies have examined the eureka moment and the mental computations that get us them; yet, surely it is the shroud of mystery surrounding the processes of discovery, innovation, and invention that most urgently needs to be lifted if we are to get beyond the current witches brew. An exception to this state of affairs may be found in the literature of genetic algorithms (GAs) [1,2], although the connection of this body of work to design theory has been poorly understood, if recognized at all.* (D. E. Goldberg 1991).
Goldberg later saw this connection as the *interplay between thinking of innovation as a model of what GAs do and thinking of GAs as a model of what innovation is* (David E Goldberg 2000). He stated that the latter is the most important intuition, arguing that if this connection holds, what we really are doing when constructing better GAs is creating a computational model of the innovative processes of humans. Several authors have followed this intuition with the development of better GAs directly, but also in connection with computer aided innovation and design theory as well as many other topics.
In the area of using GA in computer aided innovation, Gábor and Ekártab (2003) states that *although the basic aim is to provide a solid basis for building computer programs to automatize—or at least assist—the design process, modeling has contributed to a better understanding of what design is.* Similar claims have been made regarding the development of design methodologies; *there is no doubt that much resemblance exists between evolutionary design [D. E. Goldberg, 2003; P. Bentley, 1999] and the design methodologies described in engineering literature [Wood, K.L. and Otto, K.N., 1999], so why not thrive on it more extensively in the modern product development processes?* (Stanković, et al. 2006). Though Goldberg's intuition exemplifies maybe the earliest and strongest connection between GA and innovation, he only makes this link to certain parts of the innovation process. However, other authors have also made the jump from human problem solving and design theories and methodologies to natural evolution and GA in additional areas of innovation. The actual similarities will be elaborated in section section 3.

Searching for a design theory that draws on the similarities of innovative processes of human thinking and GAs is therefore interesting for two reasons. First, as suggested earlier, the

implementation of new tools drastically change the ways in which human processes of innovation are conducted. Understanding similarities between the building blocks of GD and human problem solving is therefore essential to be able to utilize these tools to the fullest. This especially applies because GD seems to have such a broad usage that in the future may permeate the whole product development process, as well as its highly different nature from existing tools and systems. Secondly, as claimed in this section, there exist similarities between human innovation and GA. Understanding and adopting the mechanisms of GA that relates to human innovation - and in that sense the ingenuity of nature - into our design theories and methodologies could *in it-self* create more innovative product development processes.

2.1.5 Limitations of Genetic Algorithms and Generative Design

Although GD as presented seems to solve all problems, there is obviously limitations that hold this technology back. Already, the huge amount of data power required to run advanced optimizations is mentioned. Another huge difficulty is that of describing problems in order to create good problem representations for the algorithm, leading to good solutions. Rothlauf (2006) has done an extensive work for creating guidelines for creating good problem representations for Genetic and Evolutionary Algorithms, but still stress that since *no theory of representations exists, the current design of proper representations is not based on theory, but more a result of black art.* He goes on to explain that *the lack of existing theory not only hinders a theory-guided design of new representations, but also results in problems when deciding which of the different representations should be used for a specific optimization problem.* If constraints of a problem can be described as well-defined ranges and connections, the process of constructing the representations may be simpler, but this is often not the case in real-world problems. Further, all the operators of the GA have to be fine-tuned to create good results. The rate at which they are applied and the choice of different type of operators highly affect the performance of the algorithm for different problems (Gábor and Ekártab 2003).
The complexity and computation power requirements also grows extensively when moving form optimizing single problems to several aspects of a design simultaneously. Although multi-model optimization methods have been commercialized (such as multi model optimization in Altair Hyperworks and Dassault System Isight), it is still in a crude state when it comes to optimizing the totality of several parts of a design and all its interfaces.

2.2 Existing Generative Design methodologies for Product development

Few studies are conducted on the implementation of GD in product development in a broad sense. The most comprehensive study, although not heavily cited, is to the author's knowledge by Nordin (2018). Nordin's study of *challenges in the industrial implementation of GD Systems*, is a case study on two different firms, one developing aesthetic products, and the other technical products. The difficulty of finding articles describing the general implementation of

generative design is something Nordin also explaines; *Though many studies have been aimed at validating either the technical feasibility or the usefulness of generative design systems, there is, however, a lack of research on the practical implementation and adaptation in industry.* Nevertheless, he describes six points that seem to create issues for implementation ito problem-solving of both artistic and technical nature:

1: Moving from automation to generation
2: Moving from designing a product to designing an algorithm
3: Knowing what to automate
4: Replacing rules-of-thumb with measurable constraints and objectives
5: Avoiding loopholes in the constraint and objective formulations
6: Parameterizing and simplifying geometry

He explains that the main difficulty lies in the new systemization of parts in the design process that GD demands, and that companies have a hard time adapting. Knowledge is also seen as an important factor in Nordin's research. In cases where commercial systems are not applicable, the designer or engineer also have to understand programming to be able to optimize a problem. Going from creating a design to describing the design in algorithmic terms is a huge barrier for the design process. Nordin quotes the renowned computer scientists Donald Knuth to explain this problem: "*Meta-design is much more difficult than design; it's easier to draw something than to explain how to draw it".*

Some articles have been written on the importance of the interplay between CAD and the generative systems methods, putting this in the core of the generative design approach. Krish (2011) describes the the Generative Design Approach (GSM) as a theoretical framework for based on the interplay between parametric modeling and evolutionary systems such as GA. He states that *the genotypes are CAD models and the phenotypes are instances of it, thus the mapping between the two is direct,* developing the model to implement generative design directly onto CAD systems. Further he states 5 stages to implement the model:

1. Creating the genetic model.
2. Setting the initial envelope.
3. Generating designs.
4. Filtering phenotypes.
5. Selection & fine tuning.

Although the method has value, it is designed for a specific case; that of implementing generative systems into parametric modelling. In a way, it therefor only creates generality for this specific use.
The generative modelling technique called Knowledge based Integrated Design and evaluation System (KIDS) developed by Volvo Aero (Isaksson 2003) also puts modeling in the center of development, stating that *the product model is the carrier of all product information*, which the design process is continuously improving. The model ascribes the generative ability of the process to that of reusing past knowledge existing in the models in an in a way that can

generate new products, reducing the level of abstraction in the early stages of design. However, the methode is developed to reuse pas knowledge, it is more of a framework of storeing CAD knowledge in a generative manner, so that the company can reuse parts and combination of parts in creating similar products. The method also does not link this generating ability to generative design or any optimization method, and is purely a model framework. A simple example could be that of creating a connector between two different drain pipes; by parametric representation dimensions and number of bolts, or reuse of old models from a library, the connector can be generated rather than modelled. Similar frameworks can be found in the field Knowledge Based Engineering (KBE), however the general focus of this discipline is building knowledge based systems that can generate product options based on previous work, and does not to a large degree use optimization in these process.

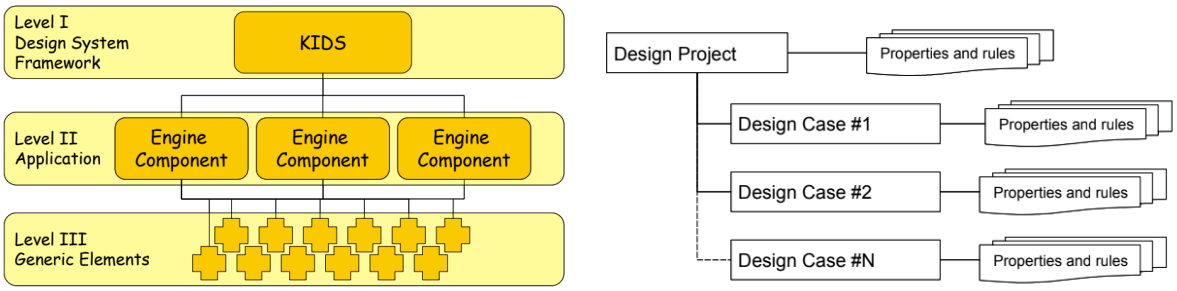


Figure 5: A graphic representation of the KBE system of Volvo Aero, utilized in the development different variants of jet engines. Left: the three levels of KIDS. Right; the first level of this framework describing the modular representation of problems.

A more general view focused on the human part of the implementation is the *constraint-based human-machine cooperative interactive product design system* proposed by Guoyan et al. (2009). The model attempts to describe the interaction between human and computer decision-making and operations, creating a human-machine cooperative system. In order to create a this interface, Guoyan et al. describes three important factors:

- *Developing visual interactive tool of optimization process.*
- *Saving variables, objective functions, constraint network and search strategy in model library.*
- *Developing an interface between interactive tool of optimization process and model library, so that the designer can modify model through the interactive interface.*

Guoyan et al. also weight the importance of creating a flexible model so that variables, constraints, objective functions and search strategies can be modified by the designer directly by modifying the model, instead of numeric interaction. This allows the designer to respond to feedback information of the system in the interface between human and computer, so that the optimization process can be guided.

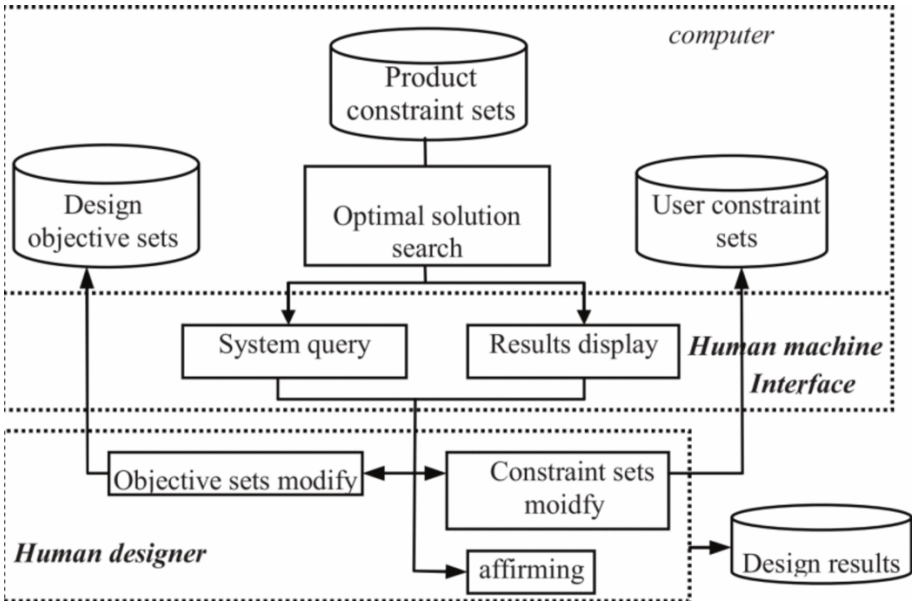


Figure 6: constraint-based human-machine cooperative interactive product design system proposed by Guoyan et al. (2009)

3 Development of a Generativ Design Methodology

3.1 Angle of approach

There are numerous design theories and methodologies that conceptualize and categorize design activities in different ways, as the activities depends on the viewpoint and level of abstraction (Sim and Duffy 2003). A ground is therefore needed to set the level of abstraction and angle in a meaningful way.
With the notion that both human problem solving and GD can both benefit and supplement each other, the strategy is to approach a methodical framework that fits both human innovation processes and the requirements and operators of GD. The starting point is therefore to compare the workings of GA to psychological theories of mental activities in human problem solving. *While using innovation for explaining working mechanisms of GAs is very useful, as a design metaphor it poses difficulty as the processes of innovation are themselves not well understood* (Sastry, Goldberg, and Kendall 2014). Starting with comparing GA to the psychology of human problem solving to create an angel of approach might therefore be a better than comparing GA directly to existing design theories or methodologies (DTM). Even though the mechanisms of human problem solving also is of somewhat mystic nature, it is more defined in simple terms in literature. One could also say that innovation simply is problem solving done in "novel ways" that creates value, and that a "good" design theory for product development is a mindset for solving problems in a way that fosters innovation. Thus, the comparison of the steps and activities in GA and human problem solving will be used as a ground for determining the level of abstraction and angle of approach.

After a general framework and level of abstraction is created in this manner, the next step will be to compare GA to acknowledged DTM in literature within the purposed framework to find concepts in product development that show some coherence for both GA and the psychology of human problem-solving. DTM can be defined as methodologies and theories created to include one or more areas to manage design, such as *design knowledge, design information, design process, resources, and design complexity* (Tomiyama et al. 2009). Many of these theories overlap in several areas, as they all try to create models and tools that can aid the design process. However, they often differ in what is regarded as the central focus, and some only describe a certain area of the process. Comparing GA to DTM through the proposed framework is consequently done to find areas and different points of focus of DTM in general that should be emphasized in a design theory for implementing GD in new product development.
Covering all parts of all acknowledged theories and methodologies to find coherence in this way is out of scope of this paper, but parts of some theories and methodologies have been selected as they seem to bear resemblance with the workings of GA. It is important to state that the comparison of DTM to GA in this manner is *not* done to reinforce the notion of GA as a theory of human innovation, as one could always find similarities when comparing two such broad fields. The reason for finding coherence is simply done to find ideas from DTM that can aid the implementation of GD in new product development, *based* on the two notions explained; that GA as a theory of human innovation might be useful, and that the implementation new tools in product development will impact the human problem-slowing process in a manner that resembles the workings of the tool itself.
Finally, in formulating a methodology, it is important to bear in mind that there is no point in creating a theory without the intention to use it. Tomiyama et al. (2009) highlights two important problems that was in focus when Hans Grabowski established the Universal Design Theory in 1990s; *the problem of universality and the problem of applicability in industrial practices.* These aspects need to be considered carefully throughout the development for of the methodology to be helpful in any way.

3.2 Creating a framework; comparing GA and human problem-solving

As stated in the previous section, the effort of finding common ground between GA and human innovation starts with the comparison of the sequential steps and activities of GA to the mental steps of and activities human problem solving. Steinberg and Davidson (2003) states that several psychologists have *described the problem-solving process in terms of a cycle [Bransford & Stein, 1993; Hayes, 1989; Sternberg, 1986],* and that humans goes through mental stages in an iterative cycle to solve problems:

1. *Recognize or identify the problem.*
2. *Define and represent the problem mentally.*
3. *Develop a solution strategy.*
4. *Organize his or her knowledge about the problem.*
5. *Allocate mental and physical resources for solving the problem.*

6. Monitor his or her progress toward the goal.
7. Evaluate the solution for accuracy.

They point out that the cycle is descriptive, and that these stages are not necessarily sequential for all problem-solving processes. The iteration are necessary because the *solution to one problem gives rise to another problem, which then again needs to be solved through the problem-solving cycle* (Steinberg and Davidson 2003). Hayes (2013), one of the authors referred to by Steinberg and Davidson describes the problem-solving process more in terms of the activities a person goes through rather his or her mental processes:

1. Finding the problem; recognizing that there is a problem to be solved,
2. Representing the problem; understanding the nature of the gap to be crossed,
3. Planning the solution; choosing a method for crossing the gap,
4. Carrying out the plan,
5. Evaluating the solution; asking "how good is the result?" once the plan is carried out, and
6. Consolidating gains; learning from the experience of solving

Even though the steps of these two explanations of human problem solving differ in some points, they share a similar understanding. How can these ways of breaking down the problem-solving process relate to the way GA works? As described in section 7.1.2, the GA goes through a series of steps that are iterated to generate a solution. To give the reader a way to compare, the steps described by Sastry, Goldberg, and Kendall (2014) are briefly summarized again:

1. Initialization: An initial population of solutions is created randomly or with some knowledge of the domain.
2. Evaluation: The initial population or offspring population is evaluated against fitness criteria.
3. Selection: Copies of solutions of higher fitness are reproduced to enforce a survival-of-the-fittest mechanism to the population.
4. Recombination: Two or more solutions are selected as parents and combined with crossover to create new offspring of solutions
5. Mutation: Single solutions are selected and small changes are made, adjusting the solution in its vicinity.
6. Replacement: The offspring solutions created by step 3-5 are put back into the population, and different methods to decrease the old generation are implemented.
7. Repeat: the steps from 2–6 are iterated until the defined termination criteria are met.

These steps are the very basic of the conventional GA. Variants of GA may exclude or have additional steps in the sequence as well as other configurations of ordering the steps.
Seeing a connection between the two ways of solving problems is difficult with these simplifications on a high level, but the author would argue that similarities can be drawn. In the following paragraphs, the basic ideas creating the ground for these similarities will be explained briefly, and will be elaborated in the following sections by relating design theories and

methodologies to GA. The similarities drawn are based on four notions with the aim to cover both the human and GA problem-solving process; that of Representation, Thinking, Evaluation and Influence:

First, in solving a problem, both humans and GA needs to understand the problem itself. Before the initialization of the first population of the GA, the problem needs to be described in algorithmic terms in such a way that makes it possible for the algorithm to represent the first population, making recombination and mutation is possible. As Gábor and Ekártab (2003) states; when designing a genetic algorithm for a given problem, choosing the representation (i.e. constructing the chromosome) is the first step. Without representations, no use of GEAs is possible (Rothlauf 2006). In one sense, this is true for human thinking as well; to solve a problem a mental representation is required to understand it, as well as a way of organizing knowledge about the problem to create clarity, such as visualizing its parts. On the basis of this view, one could say that there must exist some form of representation of the problem and solutions, as well as knowledge and its relations to be able to approach it.
Secondly, in order solve a problem there must be some form of thinking that generates new solutions or ideas for solving the problem. The "thinking" that generates solutions in GA is the process of selecting solutions and applying recombination or mutation. In the same manner humans must develop a strategy, and allocate mental and physical resources for solving the problem (Steinberg and Davidson 2003). As will be elaborated later it is the connection between these "thinking" operators of GA and human thinking that created the basis for Goldberg's intuition of GA as a model for human innovation.

Third, both human and GA problem solving needs a way of evaluating the fitness of a proposed solution to the problem. Without evaluation, there is simply no means of measuring progress or how good a solution is. GA ensures this by using a fitness function, or functions, that evaluates how well the solution solves the problem, as well as evaluating by comparing it to the population of existing solutions. A person must in the same manner find a way to monitor his or her progress toward the goal (Steinberg and Davidson 2003), by evaluating the fitness of the a solution to the problem. In addition, measuring the solution against the population of similar existing solutions is in the same way central to measure progress.

Fourth, after a solution is evaluated, new knowledge is created in relation to the problem. The recognition of new knowledge, as well as the new solution itself creates a ground for determining an *influence* on the old representation. This influence is then what regulates how one should change the former representation, allowing the process of solving the problem to begin a new iteration, and create new solutions with an updated body of knowledge and potential solutions to the problem. GA determines this influence by manipulating the population with a certain rate reproduction and replacement, creating more of the fit solutions and less of the weak. The next generation will then develop with a better ground for creating good solutions. Similarly, a human must *consolidate the gains* by *learning from the experience of solving* (Hayes 2013)*,* and use this to update the body of knowledge and solutions creating a ground for a new.

The basic ground for the methodology is then the iteration of these four steps;

1. Create a *representation* of the problem; the solutions, knowledge, objectives and its relations.
2. Utilize the representation to guide the *thinking* required to generate new solutions.
3. *Evaluate* the new solutions to determine fitness and gain new knowledge.
4. Recognize and determine what *influence* the new solutions and knowledge should have on the old representation, leading to an iteration by creating a new representation in the first point.

3.3 Comparing GA to DTM within the framework

3.3.1 Representation

In creating a representation of a problem, two aspects seem to be similar for GA and human problem solving. The first is the means of decomposing the problem into smaller sub problems in order to create a representation. The second is the link between knowledge and solution relationship in human problem solving, and genotype and phenotype relationship in GA. First, the similarities of decomposition will be discussed.

3.3.1.1 Decomposition similarities

When using Genetic and Evolutionary Algorithms for optimization purposes, representations are required for encoding potential solutions (Rothlauf 2006). In a way, representing the problem is the same as representing a solution, or at least the possibility of a solution. The goal of representation is then to create a problem-solution abstraction. Decomposing a problem in into smaller sub-problems and understanding the relations is therefore crucial for the construction of a working GA, as these sub-problems are the basis for writing the different genes of the algorithm. The relationships between these sub-problems are of equal importance, as they determine the laws of how the algorithm can combine the genes to create solutions. It is then only by this problem decomposition into smaller sub-problems, solving them separately and combining them to form different solutions, that the algorithm can evolve better solutions (Rothlauf 2006).

The analogy to human intuition is apparent, as humans tend to decompose problems into smaller more manageable parts that are easier to solve. This intuition has created a whole field of study in engineering, namely Systems Engineering, as well as several design theories and methodologies for breaking down a problem-solution abstraction into sub-problems and its relations.

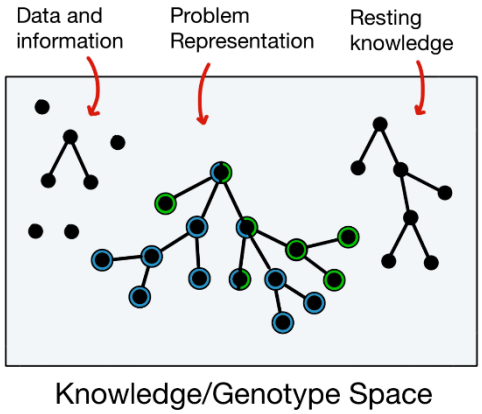


Figure 9: Representation of problem, here as problem-tree where each part or problem consist of independent sub-parts or problems. For both GA and human problem-solving, creating a problem representation and understanding its relations to knowledge and information is crucial. This creates the Genotype Space for GA which corresponds to Knowledge space in human problem-solving. For humans, this space could also contain information and resting knowledge that are not used in the current representation.

One of the widely used methodologies is the Design Structure Matrix (DSM) where *elements denote individual components of a product and off-diagonal numbers (or marks) represent interactions between the components* (Tomiyama et al. 2009). This guides the product development to decompose and understand the elements of the proposed solution to a problem and the interactions between the elements in a similar manner to that of GA. The Contact and Channel Model (C&CM) is another methodology that decompose a solution into building blocks to create an integrated model; that of *Working Surface Pairs (WPS)* and *Channel and Support Structures (CSS).* The method is created to guide the designer in representing the system coherently by understanding the functions, shapes and its relations to the environment, with an emphasis on the interfaces between parts (T. Tomiyama et al. 2009). The decomposition into smaller sub-problems is also imperative in organizational environments, as different backgrounds and competences are needed in different areas of a larger problem. Decomposition could really be seen as the cornerstone of human's ability to solve large problems.

Nevertheless, in representing a problem to be solved by a GA, simply describing the sub-problems and relations is often not sufficient. It could be enough if the problem for example is to optimize an assembly choosing from a range of fixed parts for fixed positions, but often the sub-problems are more complicated, such as finding an optimal topology. The algorithm need a defined space in witch to operate, hence the representation must be described in such a manner that a range can be defined. Understanding the constraints of the problem is then essential for this problem decomposition in GA, as the constraints set the frames in which the algorithms can work within (Gábor and Ekártab 2003). An example could be that of optimizing a bridge; one part of the representation could determine the length and number of cables used, while another part determines the thickness of the cables. Thus, the constraints of the problem is needed in order to find valuable solutions, as one probably does not want a million cables of 1mm diameter, or one cable of 10m diameter to be part of the solution space. Setting the constraints right is then important to determine the solution space to explore and the allowed combination of sub-parts; *if the representation is very general, the space could be too large, resulting in too many impossible, spiky or unusual shapes being generated, and the probability of finding valid shapes can be quite low. …On the other hand, limiting the size of the search space or access to some of its regions by the genetic representation may hinder innovation in the GA process* (Gábor and Ekártab 2003). It is important here to differentiate between the representations of the problem itself, defined by the constraints, and the representations created within the constraints as genotypic solutions, defined by points within the constraints.

In a similar manner for human problem-solving, the *design process can be viewed as a constraint satisfaction problem: given constraints on functionality, structure, and manufacturability, produce a detailed structural description of an artifact* (Sapossnek and Center 1991). Often, the constraints handling of problems is embedded only as a sub part of many design theories and methodologies, but there are also ones that put this notion in the center of the reasoning, such as Parametric Design Thinking (Bhooshan 2017) and Constraint-Based Systems Design like DOC (Design Objectives and Constraint). A constraint-based design system is defined by Sapossnek and Center (1991) as *a system capable of explicitly representing and operating upon the relationships (explicit and implicit, given, derived and assumed) between the aspects (abstract and concrete) of an artifact relating to its life-cycle concerns (including functionality, structure, manufacturability and serviceability) for the purpose of maintaining the truth values of the relationships.* They also highlight the difference between parametric design systems and constraint systems, stating that the latter separate the problem statement from the solution while the first do not. Another theory where constraints is a major part of the reasoning is Set-Based Design. This connection of this theory and GA will be described further with the notion of *influence.* Consequentially, understanding and defining the problem constraints, and utilizing tools for decomposing the problem into smaller sub-problems and their relations to create a representation is an essential part of both the human and GA problem solving process.

These, and many other similar methodologies in the field of systems engineering are developed to aid the decomposition, but often do not explain what kind of decomposition will yield the best solution. An exception of this is the theory of Axiomatic Design, where this problem is a central idea. An important part of the theory is its two axioms that states that a good design is one that have (Tomiyama et al. 2009):

- Maximum independence of the functional elements.
- Minimum information content.

In this way, it can guide the designer to decompose the problem in such a way that the elements of the design is easy to change in order to create new solutions, as well as guiding the creation of a solution that is simple but effective. Here another interesting analogy to the decomposition of GA can be drawn. John Holland, an early pioneer in the field of evolutionary computation, called effective groups of sub-solutions the building blocks of GA. He stated in the 1970s that the basic idea of GAs is that they: *(1) implicitly identify building blocks or subassemblies of good solutions and (2) recombine different subassemblies to form very high performance solutions* (David E Goldberg 2000). Holland used the term “building blocks” to describe highly fit schemata. A schema basically is a way of creating a group or subset of solutions or sub-solutions with similar genes; two sub-solutions of a problem could be described with the binary alphabet as the bit-strings (111) and (110), a schema containing these solutions would be the bit-string (11*), where * denotes a “don’t care” symbol. In the same way that GA creates good solutions by combining highly fit sub-solutions, good solution strategies are created by combining highly fit schemata of low order. The alphabet itself relates to the

information content of the representation, as a long alphabet would give more possibilities to each of the positions in the string and thus adding complexity. Goldberg used the notion of building blocks and alphabet to propose two principles for constructing good representations for GAs (Rothlauf 2006):

- *Principle of meaningful building blocks: The schemata should be short, of low order, and relatively unrelated to schemata over other fixed positions.*
- *Principle of minimal alphabets: The alphabet of the encoding should be as small as possible while still allowing a natural representation of solutions.*

To the authors knowledge, the similarities between Goldbergs principles and Axiomatic Design Theory have not been drawn before, but it is clear that there exist some sort of resemblance. One could therefore argue that these ideas from Axiomatic Design could be a guiding principle of problem decomposition that benefits both the human and algorithm.

3.3.1.2 Solution-knowledge and genotype-phenotype similarities

The second similarity between the representation in GA and human problem solving is that of dividing actual solutions from the knowledge it exists of. These could be viewed as different spaces, and for GA these spaces are called search space and solution space*. The search space is the space of coded solutions, i.e. genotypes or chromosomes consisting of genes. The solution space is the space of actual solutions, i.e. phenotypes. Any genotype must be transformed into the corresponding phenotype before its fitness is evaluated* (Gábor and Ekártab 2003). The search space of GA contains all the genetic material that builds up the solutions and possible solutions, and could therefore be viewed as a space of knowledge or information that every solution is constructed from. With direct analogy between GA and nature, the DNA of a person contains knowledge and information regarding the person, but is not the actual person. To evaluate how well the person can run, the DNA is therefore of little use. In the same manner, to evaluate a solution generated from GA, the phenotype must be used. An example of the genotype could be the numerical values of positions for all the nodes in a 3D representation of a car, while the phenotype is the 3D representation itself. It is impossible to see the aesthetics of the car by looking at the numerical positions.

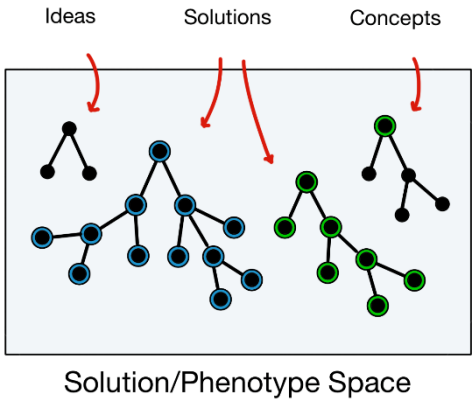


Figure 10: A distinction is made between the Knowledge/Genotype Space (Figure 1), and the Solution/Phenotype space, containing the solutions created by the representations in knowledge/Genotype Space. For humans, this space could also contain ideas and concepts.

There are several design theories and methodologies that also stress the importance of this division. General Design Theory (GDT) defines an entity as a *concrete existing object*, and entity concept as *its abstract, mental impression conceived by a human* being. (Tomiyama et al. 2009). GDT states tree axioms to define the notion of entity, entity concept and their relation:

- *Axiom 1 (Axiom of recognition): Any entity can be recognized or described by attributes and/or other abstract concepts.*
- *Axiom 2 (Axiom of correspondence): The entity set S0 and the set of entity concept S have one-to-one correspondence.*
- *Axiom 3 (Axiom of operation): The set of abstract concept is a topology of the set of entity concept.*

Although GA have the possibility to map in different ways than one-to-one (Gen and Cheng 2000 p. 6), the idea of separating the actual from the abstract is still the same as for GA. In another design theory, the Characteristics-Properties Modeling of Weber, the *distinction between characteristics and properties is put into the center of reasoning about product development/design* (Tomiyama et al. 2009). The characteristics are the things that describe the product and that can be determined or influenced directly, such as shape, materials, dimensions and surfaces. Properties on the other hand cannot be influenced directly, as they relate to the behavior of the product, qualities such as function, safety, weight, reliability, aesthetic properties, assimilability, environmental friendliness, manufacturability, cost, etc, (Tomiyama et al. 2009). Both of these theories then highlight the importance of separating knowledge and information that defines a product or solution from the product or solution itself, a distinction that is a core of how GA works; *when comparing the abilities of different individuals we must judge them on the level of the phenotype. However, when it comes to reproduction we must view individuals on the level of the genotype* (Rothlauf 2006). Based on these similarities, making a clear distinction between the solution space and knowledge space of the product development is seen as central.

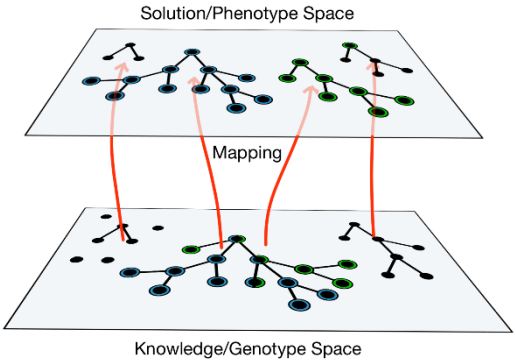


Figure 11: Different solutions in the Solution/Phenotype Space are created by using different combinations of building blocks in the Knowledge/Genotype space. Here the representation is shown as a tree structure with the possibility to map two different solutions marked green and blue. It is important to note that only actual solutions can be assembled in GA, while for humans, information or knowledge can map to ideas or concepts.

3.3.2 Thinking

Fundamental to Goldbergs view of GA as a theory for human innovation is the comparison between the "thinking" operators of GA (selection in combination with mutation and recombination), to humans mental process of continuous improvement and cross-fertilizing types innovation (David E Goldberg 2000). Goldberg describes mutation as a form of genetic hill-climbing mechanism, which does a random walk in the neighborhood of a solution by creating a similar solution where one or a few parts of the genes are changed. He argues that humans do this naturally, linking it to the continuous improvement in total quality management and the Japanese method *kaizen*, meaning "change for better". To explain this way of innovating, Goldberg quotes the British author and politician Bulwer-Lytton: *"Invention is nothing more than a fine deviation from, or enlargement on a fine model. Imitation, if noble and general, insures the best hope of originality."* (David E Goldberg 2000). Mutation can then be described as the mapping between the solution and knowledge space exploring the periphery of a solution, relating to the mental ability of *remapping primitives.*

Though mutation is a powerful tool for improving solutions, without the ability to do an intelligent jump to test another strategy, the solution will often be stuck in a local optima. This problem is solved by the recombination operator that are able to combine solutions that are far away from each other in the solution space to create originality. For humans, this is the creative ability of seeing sets of attributes or features in one context, and combining them with attributes or features from another to create a new way of solving a problem. Goldberg quotes the French mathematician J. Hadamard to explain this type off innovative thinking: *"We shall see a little later that the possibility of imputing discovery to pure chance is already excluded… Indeed, it is obvious that the invention or discovery, be it in mathematics or anywhere else, takes place by combining ideas.*"(David E Goldberg 2000). Recombination can then be described as the combination of solutions or knowledge by moving in the widths of the solution or knowledge space, relating to the mental ability of doing a *metaphorical transfer.* In classifying DTM that guides the generation of new design solutions, Tomiyama (2006) explains tree strategies to be employed, emphasizing that theories can be denoted several strategies:

- Creativity-based design: Emergent synthesis (GA, simulated annealing, ANN, and learning) and Intuitive approaches (association, analogy, stimulation methods, brainstorming, bio-inspired design)
- Combination-based design: Systematic approaches (Pahl & Beitz)
- Modification-based design: Parametric design, Case-based reasoning, shape grammar, modification rules TRIZ Emergent synthesis

Combination and modification based design here directly relates to the notions of *metaphorical transfer* and *remapping primitives* described, and are in the same way strategies for moving in the widths and between the solution and knowledge space. However, creativity-based design is seen as a strategy on its own, referred to as *a new design solution generated as a new element of the entity set.* For this to happen, new knowledge must be created, and *few theories can rationally explain it in a general framework* (Tomiyama 2006). Tomiyama relates GA to the creativity based design strategy, but even though the solutions created by GA in some cases seem brilliantly creative and far superior to what a human could develop, the algorithm is still

working within constraints determined by the knowledge and solution space, and as of yet it cannot performer magic. Seeing that GA creates novelty by using the two simple strategies of recombination and mutation, is the category of Creativity-based design then just the use of combination and/or modification in creative ways? One could also argue that GA does not belong to this category, as radical innovation requires an expansion of the knowledge and solution space by giving the algorithm completely new information to work, and this act still rest highly on human creativity taking a leap out of the current representation and body of knowledge. An example could be that of creating a tire for snowy conditions; the GA may find brilliant solutions for the grooves in the tire surface, but will newer create caterpillar tracks if this possibility does not exist within the constraints of the chromosomal representation. Still, the notion that good product development needs to employ a balance of combination and modification based strategies holds weather or not one classifies the combination of these used in novel ways as creativity-based design.

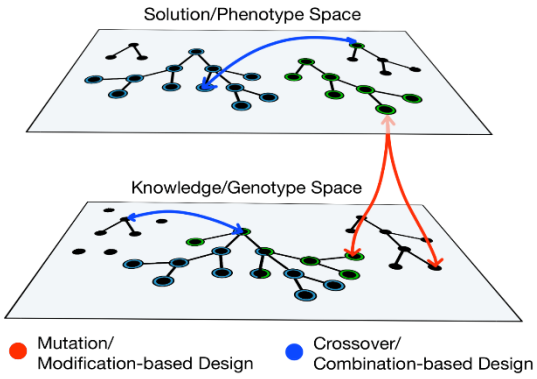


Figure 12: Mutation or Modification-based design can be seen as mapping between the knowledge/Genotype Space and the Solution/Phenotype Space, modifying parts of a solution. Crossover/Combination-based design can be seen as movement in the widths of the Knowledge/Genotype Space or the Solution/Phenotype Space, combining solutions or knowledge.

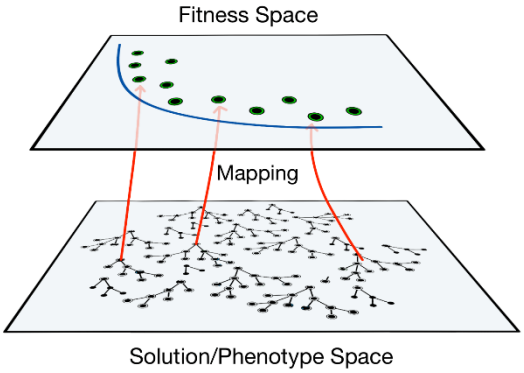
3.3.3 Evaluation

For both humans and GA, it is strictly necessary to have a means of evaluating the fitness of a proposed solution to a problem in order to know what to actually solve. In that sense, the fitness criteria are what make a problem solvable, as it points out what direction to go in the process. In some cases, the fitness is easily determined and singular, or is even the basis for the problem in the first place. A predetermined fitness criteria could for instance be given by using the goal oriented Design for X methodology, where X denotes a quality one wants to optimize, for example weight. The fitness could then easily be evaluated by measuring the weight of solutions. However, a single fitness criteria is often not sufficient for evaluating a solution in new product development, and even the Design for X strategy may lead to multiple fitness criteria originating the first (for instance Design for Aesthetics). In evaluating these types of multi-objective problems, various design decision-making methods are developed to aid in this process such as Multi-Attribute Decision Making (MADM). Other methodologies aid the designer in the creation of the fitness criteria themselves and its relations to the product, such as the initial step in Quality Function Deployment, Voice of Costumers, witch maps the costumer's requirements into the structure and components of the product. (Tomiyama et al. 2009). The variety of decision-making tools is then developed not only for evaluating the fitness of solutions and concepts for some criteria, but also aid in representing the problem-solution abstraction itself.

Both these aspects are important for the construction of GA as well. The fitness function or functions of a GA have two parts that relate to the phenotype space and genotype space. The first part *maps the genotypic space Φg to the phenotypic space Φp, and the second maps Φp to the fitness space R* (Rothlauf 2006). In constructing a representation, one therefore simultaneously construct the fitness space in order to evaluate solutions. The fitness space is thus a part of the representation of the problem although it constitutes as an evaluation process of the solutions when the algorithm is running. The initial step of problem-solution abstraction is therefore the creation of a problem representation in genotype space that maps solutions well onto the phenotype space, and a fitness function or functions that map solutions in phenotype space well onto the fitness space.
However, knowledge about the fitness of sub-solutions in the genotype space is gained through this phenotype evaluation as well, and actually serves as a strategy for evolving better solutions by combining highly fit, low order schemata: *Instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings* (David Edward Goldberg 1989).
Evaluating both solutions and sub-solutions throughout the process, as well as utilizing the evaluation criteria to guide how the problem is represented is thus seen as an important factors in the methodology.
As mentioned, product development often involves multi objective problems, and frequently these objectives are also conflicting. One might create a terrific vacuum cleaner that cleans the floor like no other on the market, but if four people are needed to lift it up the stairs, the solution might not be the ideal household product. These kinds of problems are in the GA field handled with the special class called Multi Objective Genetic Algorithms (MOGA). The conflicting objectives are in GA often calculated and visualized as a surface called the Pareto

optimal frontier, a theory originally created for economics, but later used in other fields such as engineering and decision making. This front could for instance be explained by the acceleration of a car: the frictional grip at the wheels are proportional with the weight of the car, and thus with higher weight the car can apply more torque before the wheels slip. Having a huge engine would with this knowledge be a smart strategy, but since force required to create momentum forward is also proportional to the weight the problem is not so simple. The possible combinations of weight vs. motor power would then trace out a curve where the best accelerating cars would be the ones closest to the curve. The Pareto front could be a curve, a plane or multidimensional planes that describe the frontier of various solutions to a problem with different combination of attributes. *The goal of a Pareto is to find and maintain a representative sampling of solutions on the Pareto front. Hence, the term "optimize" is the reference for finding a solution which would give the values of all the objective functions an "acceptable trade off" to the designer* (Leon 2009). The goal of the MOGA is then to close in on the Pareto front to find the best solutions while keeping trade-off open. This is an important factor in product development, as the goal is often not completely clear since costumer requirements can depend on the outcome of solutions, as well as change throughout the process. Allowing this is then a way of holding an openness to the goals within a structured fitness space. However, some new or modified objectives requires a reconstruction of the fitness criteria of the GA, and this is more difficult to implement into the iteration, although dynamic fitness landscapes are possible (Richter 2010). Nevertheless, having a way of evaluating not only the solutions and knowledge, but also the goals and fitness criteria throughout the problem-solving process is central to both the flexibility of product development and GA.

Figure 13: are mapped onto the Fitness Space, here shown as a pareto front. Solutions that are closer to the pareto front are regarded as better, while the other side of the pareto front is infeasible. Solutions with a high x-value would be better in one objective (such as weight), while those with a high y-value would be better in another (such as motor power). The pareto front could for instance describe the theoretical limit of car acceleration .



Further, central to the way GA close in on more pareto optimal solutions is the highly iterative approach of evaluating each generation. Still, most decision-making tools are designed to evaluate a single step in the development. The tool could of course be used in an iterative way, but this is often not embedded in the methodology. The Total Design of Pugh was introduced as an iterative approach to the decision-making process of product development, where concepts are continuously evaluated through a Concept Selection Matrix or Pugh Matrix based on different criteria. This was a unique contribution to the field, as it not only supports creation of conceptual solutions, but also *concept selection of the total system architecture, subsystems*

and individual components. (Tomiyama et al. 2009). Hence, iterative decision models like this better resembles the evaluation and fitness of GA.

Altogether, three iterative notions of evaluation is then seen as significant regarding both GA and human problem solving, the means of iteratively evaluating; the evaluation criteria itself, solutions in the phenotype/solution space, and sub-solutions and knowledge in genotype/knowledge space. Finally, the interrelationship between representation and fitness criteria that allows well-organized mapping from knowledge/genotype space to solution/phenotype space and on to the fitness space is regarded as important in constructing the representation of the problem.

3.3.4 Influence

After thinking has led to new solutions and sub solutions, and the evaluation process has been applied to determine their fitness (as well as the fitness criteria themselves), new knowledge has come to the table. This brings up the question of what to do with the new knowledge and solutions gained this way, and to determine what influence this should have on the former understanding of the problem, potential solutions and the body of knowledge. A GA determines this influence by altering the population of solutions with the Selection and Replacement operators. The rate at which these operators are applied determines how the algorithms evolve. Gábor and Ekártab (2003) states that this *evolution is an emergent property of artificial evolutionary systems. The computer is only told to (1) maintain a population of solutions, (2) allow the fitter individuals to reproduce, and (3) let the less fit individuals die off.* Selection makes copies of the favorable solutions to enforce the survivor-of-the-fittest strategy of evolution. The selection pressure determines how many new copies are made, and is a highly important parameter for evolving good solutions. In a similar manner that the mutation and recombination rate adjust the convergence and exploration of finding new solutions, the selection pressure adjusts the overall base for convergence and exploration for the population evolution as a whole. There are several types of selection methods, but in general the rate at which they are applied controls the convergence. *A strong selection pressure may cause the algorithm to converge to a local optimum, while a low selection pressure may cause the GA to random results that differ from one run to another* (Jebari 2013). The replacement operator works in similar ways on the old population by determining the faith of the parent solutions. It controls the population diversity by deciding the lifetime and distribution of former generations, so in that sense it is the Grim Reaper of the algorithm. There are several types of replacement strategies, like generational; that removes the solutions in terms of their generational age, or incremental/steady state; that removes the worst solutions in general (Vavak and Fogarty 1996). Overall, replacement is then needed to keep the algorithm form overflowing of solutions, as the selection, mutation and recombination operators work with the population as basis. The selection and replacement then controls the evolution of the process, and creates the *emergent* quality of the growing population. Goldberg (2000) refers to this emergent quality by saying that the *wisdom is in the population*, and states that *the population is not only the original source of good notions, but it is also the testing ground for being sure*

that the best notions are indeed the best. Moreover, the population is where we "break some eggs to make an omelette," the place where we fail so that we may ultimately succeed. Regarding GA on the notion of *influence*, it is important here to note the difference between changing the representations of solutions i.e. the genotype population, and changing the problem representation itself i.e. the constraints and objective functions that describe the possible genotypes. The latter is usually held constant in the process of solving a problem (although there exist exceptions, like Constrain Self-Adaptive Genetic Algorithms (T. K. Singh 2016)). Nevertheless, iterating on the problem representation itself outside the actual runs of the algorithm would normally be needed if one wants to construct an effective algorithm. The idea of having the reconstruction of the problem representation and fitness criteria within the act of influence therefore makes sense.

Determining what influence new solutions and knowledge should have on the former representation of the problem to move forward in product development is also highly important for directing the human problem-solving process. The basic idea drawn from GA is then to let the problem representation, and the solutions and knowledge within to evolve as a complex system through iterations, continuously creating new ground to grow from for every new generation. The idea of thinking of product development as an emergent process of "growing solutions" rather than "solving problems" is more unconventional when comparing to existing design theories and methodologies, but some appear to regard this view as central. Complex Systems Thinking is a theory that directly refers to this emergent quality, it stresses that in the development of radical innovations, a system cannot be described deterministically, but only as a complex, evolutionary system where new structures can be created. The points of qualitative change are regarded as *instabilities*, or *bifurcations* in the solution space, created when *new aspects or elements appear and grow in the system, re-structuring it, invading new dimensions and leading to emergent properties and attributes* (Rose-Anderssen et al. 2005). Although not referring to emergence or evolution in its origins, the Set-Based Design theory is another approach that bears resemblance with how GA evolves generations. A known paradox in product development is that it is often necessary to explore the low levels of solutions to create knowledge that will result in good decisions in the preliminary phases of development, but the lower levels can only be explored when the preliminary decisions are already made. This circular dependency can be broken by allowing the preliminary decisions to be stated as sets, or constraints, so that lower levels in the development can create knowledge. Iterating this approach will give more information to the higher levels of the problem, giving a better understanding of the low-level solutions, which again create more accurate knowledge at higher levels, further defining possible solutions. This is the basis for a Set-Based approach, which shrinks the possibility for premature decisions, reducing rework as well as the possibility of project termination. The nature of exploring by holding a larger solution space throughout the process also creates a more flexible approach that better can meet changes in late process (Singer, Doerry, and Buckley 2009). In this way, Set-Based design is focused on how the solution space evolves throughout the process in a similar manner to that of selection and replacement of GA by determining the influence of new knowledge and solutions of each generation and level of detail; if the process is to divergent, one may not be able to create more detailed solutions, while if the process is to convergent, it might lead to premature solutions. It is in

many ways this way of balancing the influence of new solutions and knowledge in the existing pool of solutions and knowledge and their representations that allows the circular dependency between higher and lower levels of solutions to be broken in GA as well. Hence, the determining the influence of each generation then controls the pace of the evolution of solutions, and balancing this right is a major factor for the product development process

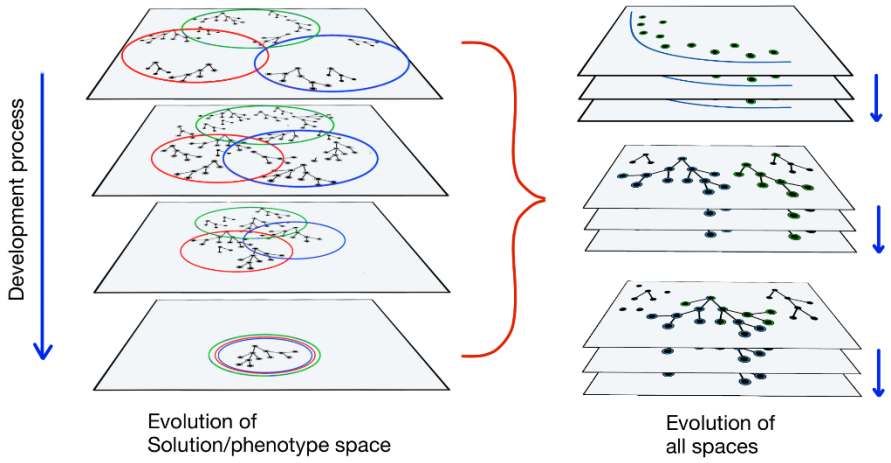


Figure 14: Left: Balancing the influence is important for the product development process. For human problem-solving, a Set-based approach can help to balance the influence of new and old solutions in the solution space. For GA, the balance of selection preassure and replacement rate is imortant. Right: In the same manner as the solution space, an influence must be determined from new knowledge and potential new objectives to evolve the former knowledge and fitness space.

Consequentially, for both humans and GA, the evaluation of new knowledge, new solutions and potential new fitness criteria and structuring of the problem, leads to the necessity of determining what influence the evaluation should have on the former representation of the problem, both its content and the constraints and relations describing the representation itself, ultimately leading to creating a new ground for the emergence of new solutions.

3.4 Proposed methodology

Throughout this paper, different theories and methodologies have been compared to GA and the steps of human problem solving. It is not an attempt to create something completely new, but to find the factors that are of importance for utilizing GA, and use this as basis for highlighting areas of known theories that might be emphasized in implementing GD. The methodology is thus more of a guiding principal for understanding the strategies and mindset that could be important for a successful GD process. An central notion behind the methodology is that of moving the human problem-solving strategy towards the more emergent quality of natural evolution, with the idea of “growing solutions” rather than “solving problems”. Although this is a somewhat vague statement, it is still creating a different angle of approach. The following figure explains the processes in relation to each other as they go through iterations, starting with the step of *representation:*

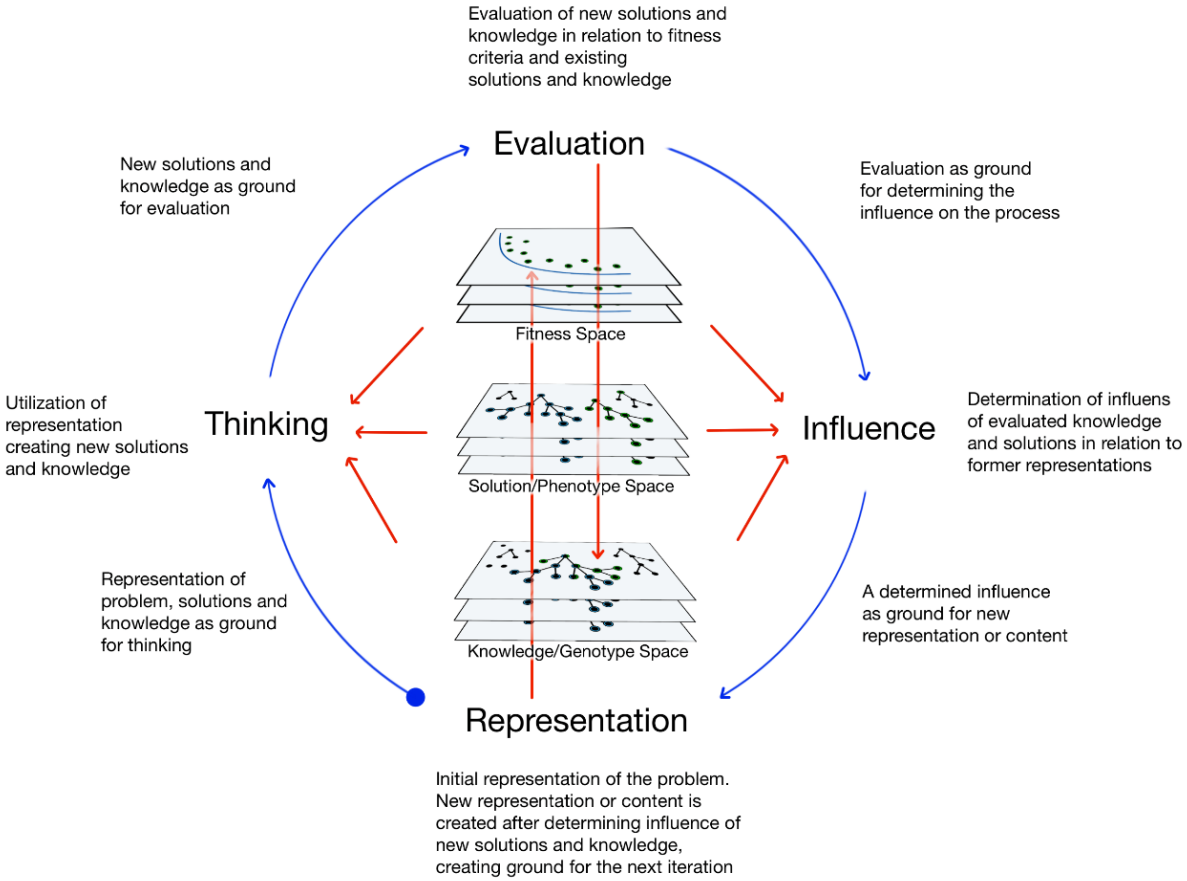


Figure 15: Starting with the initial representation, the steps of problem-solving goes through the circle in iterations, creating changes in the Knowledge-, Solution-, and Fitness Space for each iteration. The blue arrows describe knowledge that are transferred from each step to another. The red arrows represent the act in each step: Representation exerts change in the spaces, and is thus directed upward starting with the knowledge space. New knowledge creates new solutions and can result in change in objectives. Thinking utilizes information in the representation as a whole to create new solutions and knowledge, and the arrows are therefore directed outwards. Evaluation exerts change in the structure by sorting different solutions in relation to the fitness space, and is thus directed downward. Influence is determined by comparing evaluated solutions to the existing solutions and representation, and the arrows are therefore directed outwards. For each rotation, a changes are made to the representation creating a new level in each space.

3.4.1 Summary of important factors for each step:

Representation

- Utilize decomposition strategies such as systems engineering for dividing the problem into smaller more manageable sub-problems and their relations. This applies to all spaces, such as parts, objective and functional requirement structures.
- Describing problem as constraints and boundaries rather than points, creating a framework for solution generation.
- Creating a division of knowledge space, solution space and fitness space of the representation to create a clear problem decomposition.
- Have a high degree of flexibility in the solution space, allowing for rapid adaptation of new knowledge to be implemented.
- Creating a structured fitness space where objective and requirements and their relations easily can be mapped onto solutions and sub solutions for rapid evaluation
- Using Axiomatic Design as guiding principle of the quality of the representation.

Thinking

- Utilizing the representation as a base for generating solutions.
- Determine strategies of solving the problem, balancing Combination-based design and Modification-based design to generate solutions.
- Understanding where GD can substitute or aid the generation process, and when it is strategical to employ this method over normal approaches.
- Determine if a leap completely out of the current body of knowledge is necessary to move on, demanding a restructuring of the problem representation.

Evaluation

- Utilizing appropriate tools to determine fitness criteria, such as simulation or decision-making tools, and visualization such as pareto for multi-objective problems.
- Evaluating new solutions, sub-solutions and knowledge in relation to the fitness space as well as in relation to former solutions sub-solutions and knowledge.
- Evaluating the fitness criteria itself, allowing new or more defined objectives to come into the process if needed.
- Understanding the interrelationship between representation and fitness criteria that allows well-organized mapping between spaces and thus an effective representation.

Influence

- Utilize a Set-Based approach to balance the overall process.
- Continuously monitor the divergence/convergence of the overall process to determine the actions to take before each iteration.
- Considering all aspects of updating the representation before a new iteration.

4 Development of Monocoque

In this chapter, the development of the DNV GL Fuel Fighter 5 monocoque will be presented in a broad sense. Many details such as calculations and dimensions are left out, as describing every aspect of the development would both be highly comprehensive and outside the scope of this thesis, as well as overshadow the goal of creating a clear view of the process for the reader. As the purpose is to create an understanding of how the Generative Design methodology process supported the development, the chapter is constructed in a similar manner to that of the process. However, simplifications of the actual process are done to make it more reader-friendly (such as describing several iterations within each iteration).
Before the initial representation and the iterations of the process are depicted, the objectives of the monocoque are explained in relation to the overall project. A big effort was taken to implement the hierarchical approach of both fitness, solution and knowledge space into the organization, so that all parts could benefit from this process. It was also to ensure that the parts connected to the monocoque would developed with the same understanding of constraints and objectives. The system was implemented in Trello as a system design hierarchy of all parts and their connections. In addition, all members where responsible for clearly stating the objectives constraints and requirements of their parts and interfaces in relation to the overall goal of the project.
The major objectives of the monocoque was in relation to the project goal can be divided into structural, aerodynamic and design objectives, which will be further elaborated. Underneath the considerations made prior to the development in terms of tools and generative systems, as well as an overview of the process for each objective is described.

4.1.1 Design tools and process overview

As the design at all times "holds" the current solution constraining aerodynamic and structural developments, choosing a good platform and methods to develop the design has a huge impact on the process.
Considering the aesthetics, the use of generative software is very limited as "the feel" of an object is very hard to state in mathematical terms. Even though there exist studies where generative systems have been used in aesthetics and car design (such as the work of Tan et al. (2013) using interactive genetic algorithms to generate car silhouette styles), it is was not considered to be of use, especially since the interplay between design and aerodynamics is so important in this case.
Finding the most optimal monocoque dimensions as a result of driver position and different parts in relation to each other is also a problem that could be tackled by algorithms, as it is a version of the classical packaging problems, a whole class of optimization algorithms in itself. But also here, employing generative systems was considered to not be valuable. The reason for this is that the major driving force determining positioning is also in this case aerodynamics. Hence, finding dimensions such as the distance between front and back wheels depends on the knowledge gained from the aerodynamic simulations throughout the process. This implies that constructing an optimization process for determining dimensions in this case would be the

same as implementing aerodynamic shape optimization, as will be explained later on have some hurdles to overcome.
The key was then to use tools where new knowledge from different domains could be tested and implemented iteratively, without these tools impairing the flexibility of the process itself. Two important tools where used in this manner throughout the process; plasticine prototyping and freeform 3D modelling, which will be explained further.

In the concept stage, the use of prototyping is of huge value for exploring and testing strategies, and can also be valuable in later stages of development. This could for instance have been done by carving shapes out of hard foam, as many previous teams have done, but this approach would only hold a single solution for each prototype. Instead, by using plasticine, each prototype could be adjusted iteratively as knowledge from aerodynamic simulations or design features emerged. The prototyping could then be conducted in a set-based manner where the flexibility of each concept created a potential for an infinite number of other variants to be explored.

The same reasoning was used to find a suitable CAD approach to represent the design. The parts connected to, or in other way interfering with the monocoque demands precise modeling, which is typical for most engineering CAD software, using techniques such as solid body or hard surface modeling. By defining the dimensions and relations of a model as parameters, known as a parametric modeling workflow, changes can be made during the development without the need to completely reconstruct the model. Parametric modeling was therefore considered to be important to create flexibility of creating and positioning parts.
However, when modeling advanced surfaces like the exterior of a car, this workflow can reach its limits. Parameterization of hard surface modeling is very efficient for simple surfaces (using functions such as railed lofting for B-Rep Solids), but as the complexity of a surface increases, there comes a point where the parameterization no longer is possible without creating complex relations that are not native to most CAD software, or by severely limiting the range of the parameters i.e. reducing the flexibility.
Another class of modeling software that handles complex surfaces with more ease, is organic or freeform modeling. The direct interaction workflow of this modeling type is more like sculpting with clay by direct modification of nodes on the surface rather than the construction of pathways, and are the tool of choice for creating animated movies, CGI effects and games. A type of ths called T-spline modelling, also inhabit G2 continuity (a measure of the surface quality), which is important when dealing with smooth surface for aerodynamics. However, almost all programs created for this use lack the parameterization and precision required in engineering, and to create the most flexible workflow, the ability to utilize both parametric and direct modeling was considered to be ideal.
Autodesk Fusion 360 and its “cousin” Autodesk Inventor is to the authors knowledge the only programs that have direct T-spline modelling and parametric modelling of solids and hard surfaces. Of the two, Fusion 360 was picked as it has an inbuilt computer aided manufacturing (CAM) and simple simulation and topology optimization that team members could learn more easily than other softwares. The idea was then to utilize this modelling approach in combination with clay modelling, and translate between these two approaches. This translation

was done with the use of a 3D scanner from prototype to CAD, and 3D printing from CAD to prototype.

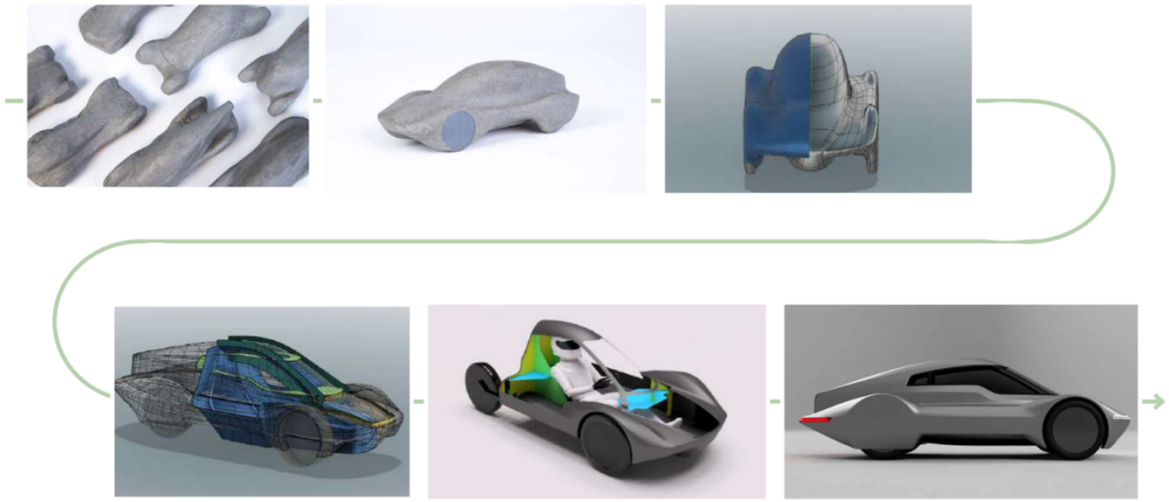


Figure 16: Several clay concepts was create innitially (1-2), and was transformed into a 3D representation (3). The constraints and body was iterated multiple times for different T-spline representations (4-5), and design features was shaped using both 3D modelling and clay modelling (6)

4.1.2 Structural tools and process overview

The use of generative design in the structural development of the monocoque was early in the research understood to be possible, as there exists a wide variety of software solutions in this discipline. The hunch was that if the right software was implemented wisely, the final product would be lighter than what would be possible to achieve trough a normal engineering approach. As learning software demands time, choosing the right one for our application was important for success, and several software was researched and tested on initial concepts before choosing. Isight, Tosca Structure, Siemens NX, and SolidThinking Inspire were considered before choosing Fusion 360 topology optimisation and Altair Hyperworks. Although not highly advanced, he Fusion 360 topology optimization tool was chosen for early concepts on the basis that the car was modeled in the same software, which allowed for quick exploration without conversion issues. Altair Hyperworks using the Optistruct solver was chosen for the major part of the development, as it is the only commercial optimization software with advanced composite laminate optimization, which also supports more common structural optimization techniques such as topology, topography, shape, buckling, size and free size optimization. The Optistruct solver also has the possibility of Multi Model Optimization, which would allow multiple different models to be optimized in the same run for multi-disciplinary studies. It is by far the most comprehensive optimization software for structural applications to date.
Before the development had begun, it was difficult to know which of these generative techniques would be useful, and exactly how the process would unfold. However, the wide range of techniques in the software setup gave the possibility to respond to implementation opportunities down the line. Creating a complete strategy beforehand would also be challenging, as there was no way of knowing how difficult and time consuming the different

techniques would be to implement. Further, such a strategy could also potentially be restrictive to the development by narrowing the solution space to early. One strategy was however used as support; the C 123 approach developed by Altair. Even though this approach came from the development of cars, it does not completely overlap the case of making a carbon fibre monocoque. However, it gave an idea of the optimization techniques that typically would be useful in the different stages of the development; topology optimization in the early stages of concept development to understand critical load paths, sizing, shape and/or detailed topology optimization as the concept gets more defined, and lastly, composite and ply optimization for the final design.
As will be elaborated in the following chapters, the implementation of structural optimization was highly successful, and yielded as much as 45% reduction in weight from the previous monocoque even though the same manufacturing techniques were used. A short overview of the developments throughout the process is laid out in the figure below.

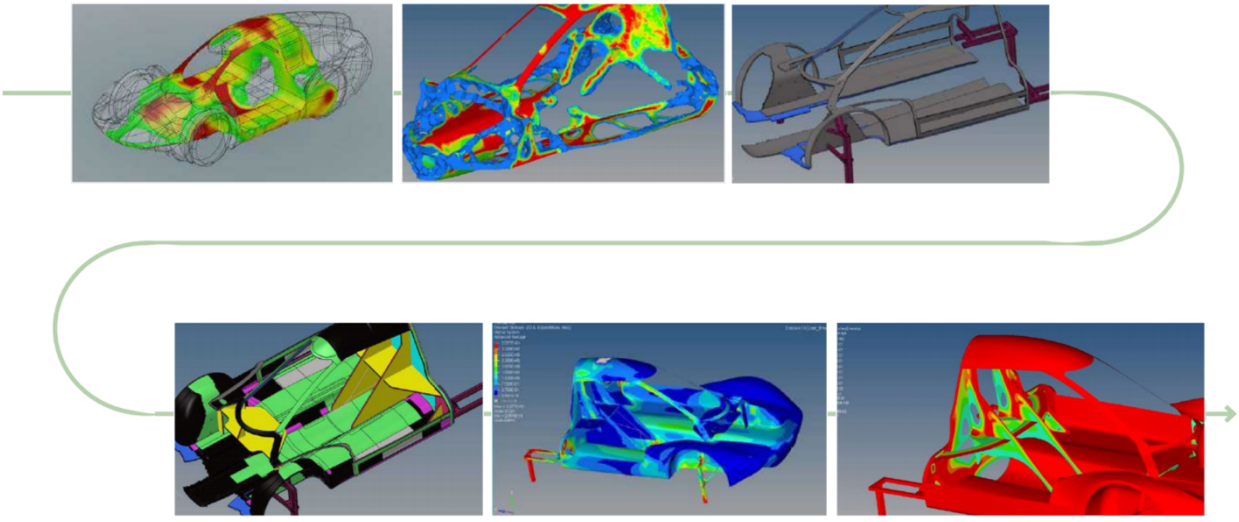


Figure 17: In the conceptual phase, several iterations of topology optimization was done to find critical load paths and structural elements in Fusion 360 (1). This was further iterated in more detailed models trough Hyperworks, creating a basis for support structures within the monocoque (2). Using the Hyperworks ply optimization module, iterations was done to find optimal core placement and carbon fibre layup taking production into account (3 and 4). Structural analysis was done to verify the optimization and decrease the number of layers to a minimum (5). Finally, 2D Topology optimization was done to remove excess material from support structures within the monocoque (6).

4.1.3 Aerodynamic tools and process overview

Aerodynamic simulation (CFD) is in general a highly data intensive task, especially for complex cases like the airflow around a car body. Even though the increase in computational power available has advanced the field extensively, there are still research being done to improve the accuracy of simple cases like the airflow around a 2D circle. When it comes to aerodynamic optimization, the problem of available computational power becomes even more prominent, as a large number intensive simulation runs are required.
Generative design in CFD is therefore in its infancy compared to structural applications, and has only recently been introduced in the car industry, mostly with the use of tailor made genetic

algorithms and evolutionary strategies for specific cases (Othmer and Grahs, 2014). The design process of developing low drag vehicles is therefore still mostly dependent on iterations done manually by engineers supported by simulation software (Karpouzas et al. 2016). Nevertheless, the three most promising aerodynamic optimization techniques for this case were researched for implementation in the development of the monocoque.
The first idea was to utilize so called black box shape optimization, where the 3D model of the exterior is parameterized in such a way that the nodes describing the surface can be modified automatically. Coupling this with aerodynamic simulation that responds to evaluations of each run by iteratively changing geometry variables (the node position and curvature), can optimize the shape for a given objective function, i.e. minimizing drag. This could for instance be implemented through Isight, coupling Ansys for simulations, Catia for 3D model parameterization and a suitable optimization algorithm from the Isight library. The drawback is that creating parameterized nodes for describing a surface can be highly complex. The cost of running the optimization is also proportional with the number of variables, and for advanced surfaces such as the car exterior consisting of hundreds or thousands of nodes, each describing several directions and curvatures, this becomes quite problematic when the limit at this point is closer to ten variables (Carsten Othmer 2014). No example of large scale implementation for car bodies is therefore found in literature, and hence the technique was not considered to be viable.

Another approach that held some promise for the early concept phase was to utilize a new type of fluid optimization based on mesh elements rather than surface modification. The technique, developed as late as 2003, works in similar ways as structural topology optimization; a meshed design space uses local criteria to iteratively remove counterproductive areas with respect to the objective function (Carsten Othmer 2014). The idea was to have the program find the optimal shape around parts that where strictly necessary; wheels and the minimum cockpit of the car and run several setups with different distances between the parts. Although not creating perfect surfaces due to the pixilated shape of meshed bodies, it does not rely on the control of nodes, and can in the concept phase find viable solutions on a single optimization run. However, the technology has mainly been used to optimize internal flows to find optimal shapes of air-ducts, and no commercial software of this type is at this point designed for outer flow on cars. A license of Dassault Systems Tosca Fluid (one of the leading commercial programs of this type) was nevertheless acquired with the hope of adjusting it to the case, but after several attempts it was concluded to not be applicable.
It was later understood by the author that aerodynamic topology optimization on external flow has not matured yet. One of the few studies found is the 2D optimization of the Ahmed body (a simplified body resembling a car) done by Othmer et al. (2017), showing promising results. Othmer also stated that only a one other paper was found on external flows, so weather or not more have been done recently, the technique is definitely in its early days. It is however the author's opinion that this would be the ideal starting point for generating low drag shapes in the early concept phase of the project when the technology is more available.

Lastly, the adjoint sensitivity type optimization method was considered. In many ways this is similar to the shape optimization, but leaves out the automatic adjustment of the surface.

Instead, it creates a sensitivity map on the 3D model that describes strategies that can be implemented for a given objective function. In the case of optimizing for minimum drag, the optimization generates colors on the 3D model describing the areas that should be moved in or out normal to the surface to lower the total drag coefficient. In a way, this creates an optimization process where the engineer substitutes one of the elements of a complete generative system, responding to each iteration by adjusting the cad model.
Of the described optimization methods, it is definitely the most applicable at this point in time, and has several commercial available softwares specifically designed to handle external aerodynamics of cars such as Engys Helyx and Ansys Fluent Adjoint Solver. The latter was tested by one of the team members, but was never fully implemented due to convergence issues that were not resolved before the semester ran out.
As car aerodynamics is a fairly advanced simulation process that requires a lot of experience, it seemed that the implementation of even more advanced aerodynamic optimization techniques might be too complicated for a student project like DNV GL Fuel Fighter, especially when none of the team members working with aerodynamic simulations where writing master thesis on the project. However, we came close to implementing adjoint based optimization starting fairly late in the process. An attempt of utilizing this technique for the next generation car could therefore be very promising.
Even though the use of generative systems was not realized in the aerodynamic development, the methodology of optimization was still employed. A rigorous process of simulations, evaluations and CAD modifications in an iterative manner yielded very successful results, with a total reduction in drag force of almost 25% compared to the previous model. In the figure below, some of the important phases of the aero dynamical process is presented.

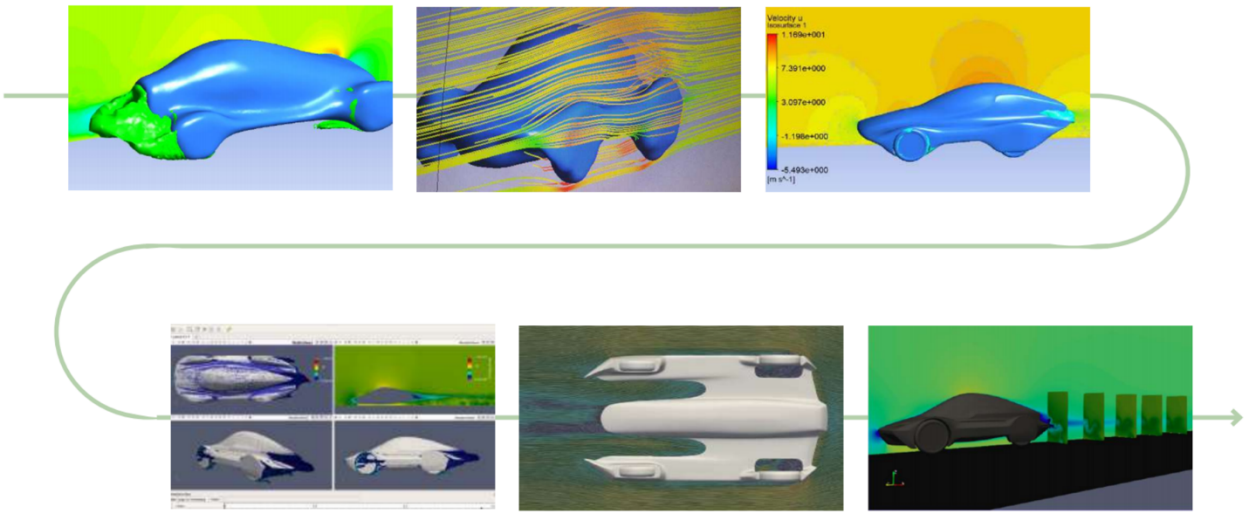


Figure 18: Initial aerodynamic simulations in Ansys Fluent was done on several 3D scanned concept clay models to test aerodynamic strategies (1). The most promising strategy was further developed in clay to reduce drag (2). An initial T-spline 3D model was developed on the basis of the 3D scanned model, and dimensions was iterated further to reduce drag and lift (3). A final T-spline model was established, and several iterations of development was done of using Open Foam (4-6)

4.2 Initial representation

4.2.1 Fitness space

Although constituting many other goals the fitness space of the monocoque can be devided into the three major objectives that can be brought back to the vision of GNV GL Fuel Fighter, to *Inspire a sustainable future - through learning and creating innovative solutions that challenge today's perception of transportation.* This vision points to two major goals; winning the Urban Concept Battery electric class, and winning the Vehicle Design Award. The Structural and Aerodynamic objectives are the major drivers for the first goal, while design is enhibits the last.

4.2.1.1 Structural Objectives

When considering the structural aspects of the monocoque, the main objective is simple; minimizing weight within the constraints and requirements of the competition and the forces acting on the car. Although this objective is simple, it requires advanced engineering to achieve good results, and it becomes increasingly complex as one moves closer to the theoretical limit. As the Shell eco-Marathon Minimizing weight is all about using materials and geometries with the largest strength to weight ratio, which - with a certain degree of safety - barely can withstand the forces applied. However, before materials completely fail, bending can become an issue and might be the limiting factor. This especially applies to flexible materials like carbon fiber in a case where displacement can negatively affect mechanical systems such as steering geometry. The second structural objective is therefore to minimize compliance (or maximize stiffness) of the monocoque to a tolerable point of displacement.

4.2.1.2 Design Objectives

The design objectives can be directed towards the goal of winning the Vehicle Design Award, which by Shell is stated to be a vehicle that is aesthetic, ergonomic, eco-friendly, technically feasible and utilize smart materials. The process of the design and overall engineering design is also weighted in the totality of the award. Although these qualities are difficult to quantify, they are seemed as highly important in the development, and is an equal part in optimizing the vehicle comparing with the other objectives.

4.2.1.3 Aerodynamic Objectives

The aerodynamic objectives can be divided into that of minimizing both the drag coefficient and frontal area, as well as approaching 0 lift. The drag coefficient is the number that describes how well a certain shape glides through a fluid, and is unaffected by the shapes volume. The frontal area of the shape (seen parallel to the flow), accounts for the size, and it is the product of these two that determines the aerodynamic losses of the car. The lift creates either downforce (negative lift making the car heavier), or lift force (such as an airfoil). Both of these scenarios are drawbacks in the energy consumption of the vehicle, and should thus be as close as zero as possible.

4.2.2 Solution space

The initial solution space was not empty to begin with, as the previous car both served as a real concept that was helpful in conceptualizing ideas and understanding the problem. In addition

4.2.3 Knowledge space

In the initial knowledge space, all the requirements from Shell that could effect the development of the monocoque had to be clearly understood. There is a large number of rules describing the dimensions of the vehicle such as total length, height and width, minimum internal dimensions of the cockpit, driver position and door sizes and visibility. These are either described as a range or a min or maximum. Further, there are several requirements for safety, stating lodes that the vehicle must be able to withstand to be allowed on the track. As the monocoque is connected to so many other parts, SEM rules of other systems might also interfere with the monocoque directly, such as turning radius requirement.
All the parts connected to the mono also had to be considered in terms of space and connections, as well as interrelationships.
When came to the knowledge of how actually build the car, the master thesis from previous FF teams was resourceful, but knowledge was also gathered from comprehensive research by the team.

4.3 Iterations

4.3.1 Iteration set 1

Representation
To create initial concept representations of what the car, it was important to first establish a real representation of the boundaries and constraints set by the rules of Shell Eco marathon. As the internal dimensions of the driver environment would be the strongest factor of the outer shape of the vehicle, a real size cockpit model was developed. With a flexible wire netting prototype of the cockpit, the driver position could be adjusted with the aim to minimize frontal area and overall volume of the driver environment, without conflicting with the driver ergonomics. The minimum volume was then 3D modelled together with other space consuming parts like wheels, drive chain and luggage space requirements. Further, these minimal required volumes was 3D printed in down scaled sizes to create a prototype platform where all volumes could be moved independently.

Figure 7: Left; the wire netting prototype, right; the minimum volumes determined by the prototype in CAD

Thinking

From the prototype representation, several clay concepts was built around the 3D printed volumes with different aerodynamic strategies. There were common strategies for these concepts such as minimal frontal area, streamlined and wing profile like shapes, and minimal area of the cars endpoint (like a teardrop). However, as there is no "ideal" car shape for low drag, strategies was drawn from other aerodynamic vehicles such as Aston Martin Valkyrie, Volkswagen XL1 (the production car with the lowest drag coefficient of 0.19) and Googles speed record-holding bicycle. Ideas was also taken from other aerodynamic competitors in Shell eco-Marathon, as the very best of them have the lowest drag of vehicles developed in general (almost half the drag coefficient of the best production cars built). This was however done with care, not to jump to conclusions and limiting the possibility of finding new strategies.

Figure 19: Left, some of the different clay concepts, right; some of the vehicles for inspiration of aerodynamic strategies.

Evaluation

To evaluate each of the concepts, the clay models was 3D scanned to create a digital mesh to create computable models. Although the surfaces of these prototypes did not have a perfect finish, the mesh was good enough to give indicative results from aerodynamic simulation trough Ansys Fluent. Computing the respective drag coefficient of the different scanned clay concepts created a rapid and flexible prototyping and evaluation system. The most promising concepts was studied more closely by looking at velocity profiles and areas of slip and

turbulence to give information for further modification of the clay model, leading to new simulations for evaluation and improvements.

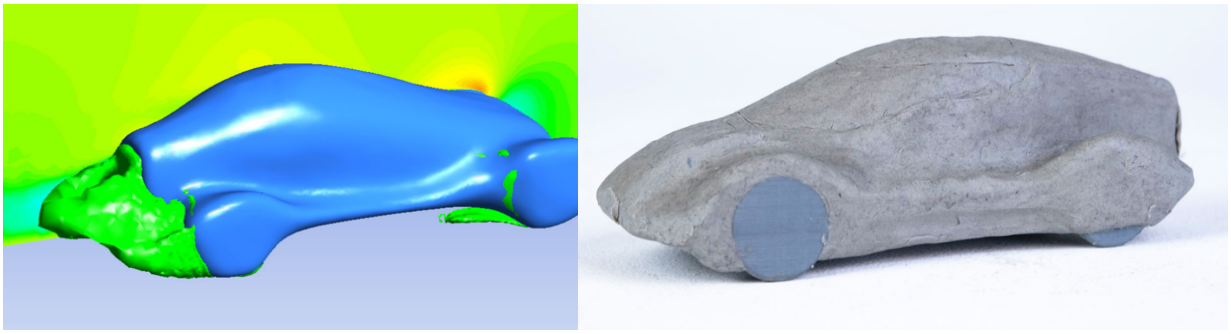


Figure 20: Left; turbulence simulation of one of the concepts. Right; the best performing caly concept.

Influence

After several iterations and simulations, the concept with the lowest product of drag coefficient and frontal area was chosen for further development, while the rest of the concepts was left behind. This decision was based on the chosen concept having a Cd of about 0.24, approximately 50% lower than any of the other concepts, as well as the lowest frontal area. This concept was by no means a well-defined shape, but served as a starting point that held promising aerodynamic abilities, as well a good potential for aesthetic development. Further, it was clear that to gain more knowledge from the simulation to adjust the model, a better surface representation was needed. A T-spline model could solve this problem and still hold the flexibility of the concept. In addition, the idea to utilize GD in form of topology optimization in the early concept stage came to the table, as this could generate knowledge that potentially could alter the outer shape of the vehicle or the driver position.

Although the concept representation at this point was singular and not a set of solutions, the concept could be said to contain an infinite set of solutions as the clay and T-spline share the same ability of rapid change. The Set-Based thinking was therefor still employed, by not defining the shape and keeping options open to gain knowledge from sets of versions.

4.3.2 Iteration set 2

Representation

The initial T-spline model of the concept was created by automatic generation, which is a rather new method, and therefor had to be done via another program. By using Autodesk Recap, the 3D scanned tri-mesh model could be transformed into quad-mesh and from this into a T-spline in Fusion 360. Surface smoothing functions ensured that the representation could create more accurate aerodynamic results, while the T-Spline setup could allow direct modifications.

The initial setup to perform the first structural optimization was made by creating a solid body representation of the scanned mesh and subtracting the minimal volumes created in the first representation. An exception was the drive chain and back suspension, as seeing how the optimization would solve the problem of connecting the rear wheels to the body could be a decision basis for choosing whether or not to keep the rear suspension from the previous vehicle. Additionally, the doors was cut out in accordance with the requirements from Shell, with the objective of minimizing the opening size. All other areas where kept filled, not constraining the optimization more than necessary so it could utilize as much area as possible. The load-cases and load constraints setup was then calculated from different scenarios, such as maximum braking, maximum turning, bump loads, the required roof and toing hook load, and passenger load when driving and stepping into the vehicle. To ensure that the optimization would not optimize for single cases, different combinations of the loads where also used, such as braking, turning and seatbelt loads simultaneously. The objective of the optimization where set to minimize mass and maximize stiffens.

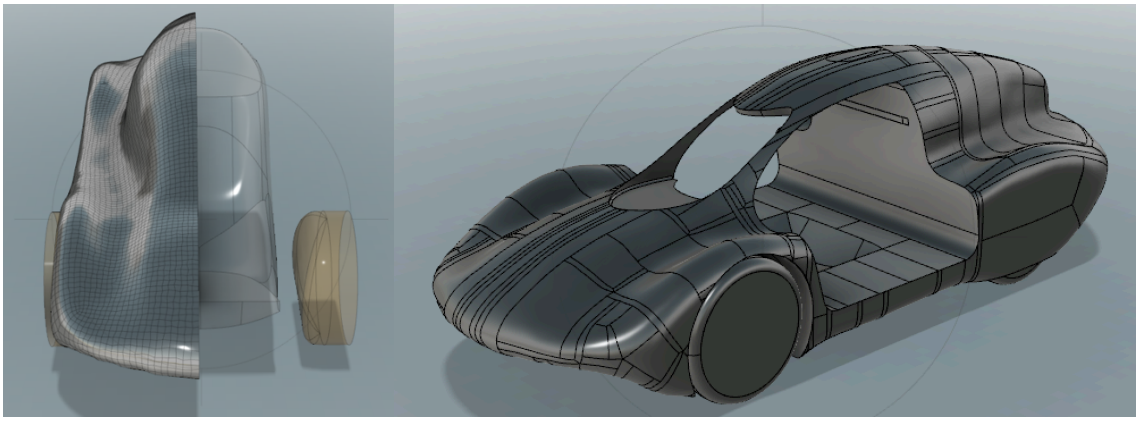


Figure 21: Left; the generated T-spline model. Right; the first solid body representation.

Thinking

The generated T-spline model was iterated to create more streamlined body and minimizing the area of the tail of the car. This was done in several steps for each CFC simulation.
As the setup for optimization was done, the “thinking” part for generating structural load paths could be performed by the computer. However, many adjustments was done to ensure that the diminutions of the CAD was right, such as reconsidering the driving position before the optimization.

Evaluation

With the improved surface finish of the T-spline, the aerodynamic simulations gave more information of the strategies what seemed to work, and what did not. Through a combination

of trial and error, as well as educated strategies from the team members, the aerodynamics was improved to a Cd of around 0.20, while lift was slightly negative creating unwanted downforce.
The results from the topology optimization was evaluated and gave an indication to the areas of the vehicle had the highest influence in creating structural stiffness.

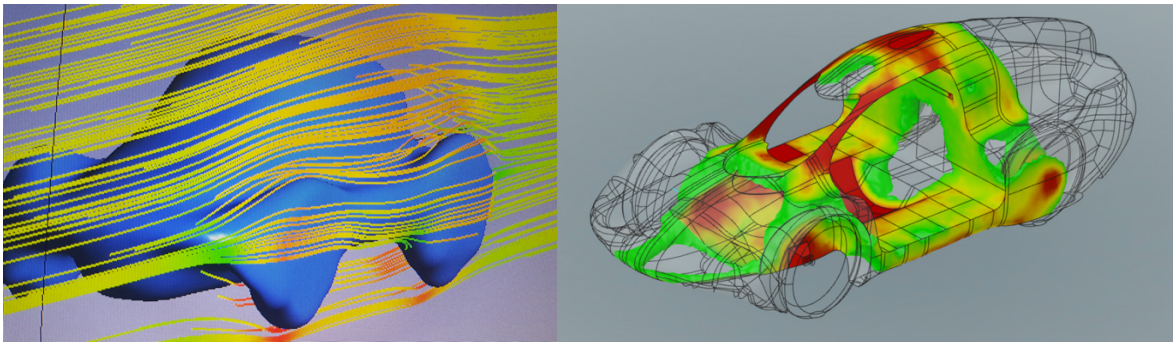


Figure 22: Left; flow simulation of the generated T-spline model. Right; Results from initial topology optimization

Influence
Although the generated T-spline model gave better results, the number of surface control points made it difficult to impose big chances in CAD (with a lower limit of 10.000 points). Adjusting areas of the car was problematic without disrupting the streamlined shapes, and a new T-spline model with less control points was therefore needed to gain the same flexibility as the clay model.
While the initial topology optimization was quite rough, not having precise internal dimensions and a rugged outer surface from being a scanned clay model, it created knowledge for future development. First, it gave an indication that the old back suspension design was a good solution, leading to the decision of keeping the design for possible reuse or rebuild.
Secondly, it gave an indication of the structural importance of the connection between a-pillars (the two narrow pilars dividing the front and the side windows) and wheel wells, in addition to support between each wheel well.
As the interplay between aerodynamic and structural changes began to emerge in the development, a better ground for deciding over the other at points of conflict was needed. Additionally, more precise load-cases was desirable to have more confidence in the values, leading to reduction in safety factors and consequently a lighter structure.

4.3.3 Iteration set 3

Representation

As the lower limit of surface control points for generating T-spline automatically was reached, a the new model had to be made from scratch. The strategy was a “just enough” principle, having enough control points to define the surface, but not more than strictly needed to make future changes as easy as possible. The new model also featured a parametric setup of the wheels and cockpit, so that other configurations could be generated rapidly. After the basic body was created, copies could be made with more control points to also explore aesthetic features. A larger base for a new clay concept was also developed for rapid prototyping od new aerodynamic strategies and aesthetic looks.

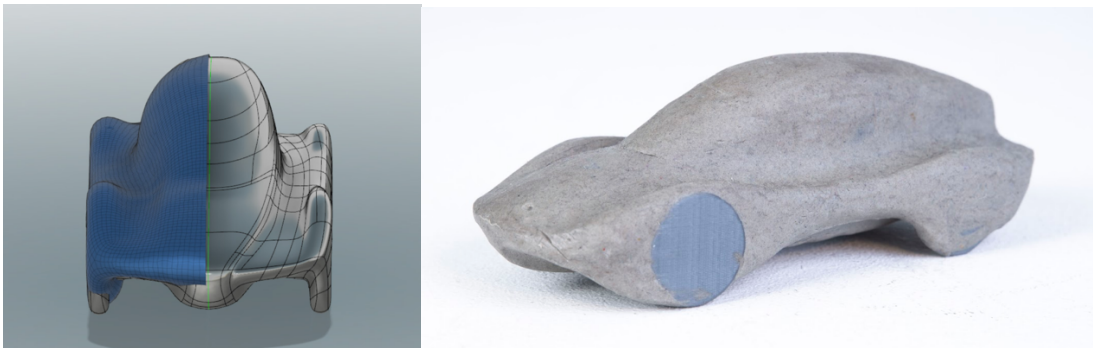


Figure 23: Left; the generated T-spline model (blue) and the initial constructed t-spline model. Right; the new clay model.

To gather more precise knowledge from the topology optimization, a new setup was made using the T-spline model as the volume and using the old back suspension.
The same load cases where used for this optimization, but an attempt was made to gather more precise data to create a better representation of the bump loads (impact from driving over uneven surfaces). The load was a big contributor to the forces acting on the car, and was highly difficult to calculate. The author therefor constructed a Design of Experiment for the old vehicle, measuring different bumps at different speeds, vehicle weights and center of gravity, so that the data gathered could transferred to the case of the new vehicle by using a the multivariate analysis and optimization tool called Unscrambler X. Another team member built and set up the experiment and gathered some limited but valuable data in the first attempt. However in the second attempt the load cell used to measure the impact forces had problems with noise, and the setup was difficult to use later because of snow conditions. Precise knowledge was therefore not gathered, but enough to give confidents of a lower limit than previously predicted, updating the knowledge space.
To create a better decision basis for aerodynamic versus structural choices, a team member working with driving strategy optimization was requested to create a model where all vehicle weights and Cd*A values where plotted against total energy consumption. Although this was a simple model of ideal conditions (flat track without corners, and even acceleration, cruise speed and deacceleration), it gave a starting point for decisions. It showed that a 10% decrease in Cd*A amounted to about 6% decrease in total energy consumption, while the same decrease in vehicle weight amounted to about 2% reduction. This meant that structural changes in most cases was inferior to aerodynamics changes in conflicting areas. The graph underneath shows the plot of the different value combinations, and could serve as a tool to determine the fitness

of solutions, updating the fitness space. Although the theoretical limit is unknown for each of the values (and especially the combination of the two), a curve plotting this limit would create the pareto front of the problem, showing where the ultimate solution for winning the competition would be.

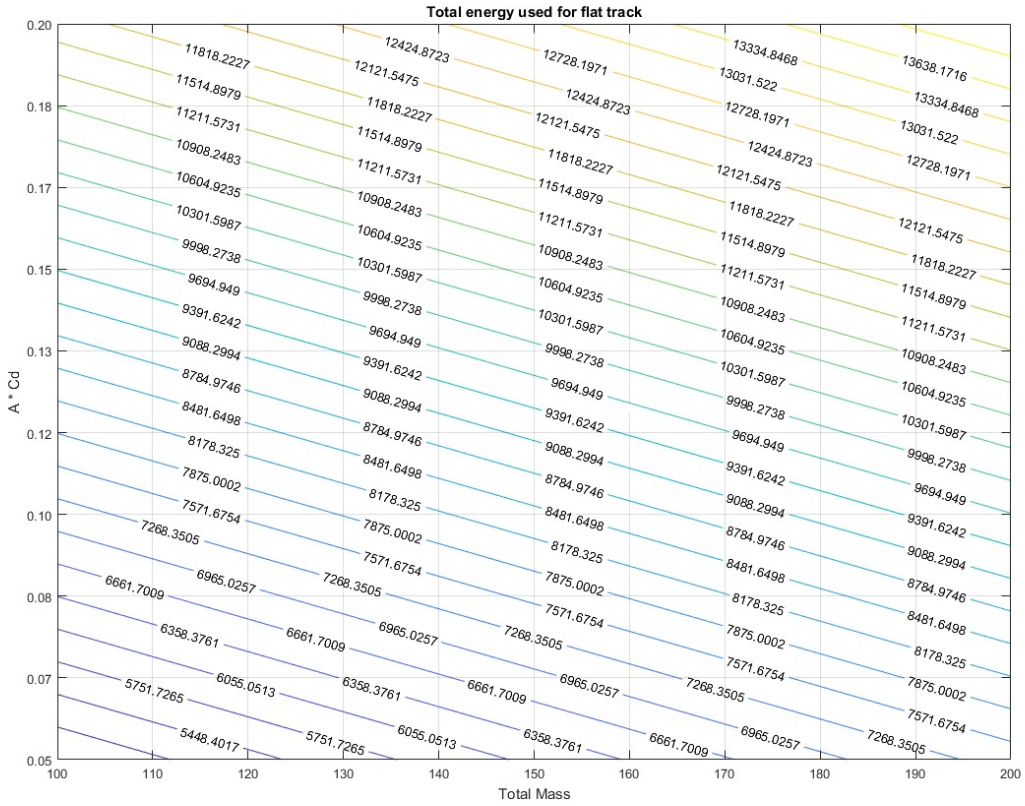


*Figure 24: The plot describes energy consumption as a function of the vehicles drag coefficient, frontal area and weight including driver. A Cd*A of 0.12 and a weight of 160kg would as an example give an energy consumption of 9395 joule*

Thinking

The new T-spline model with parametrical defined wheels and wheel wells made it a lot simpler to impose changes on the CAD model, producing a rapid interplay between CFD simulations and modelling. Together with the larger scaled clay model and several copies of the T-spline model, different aerodynamic, aesthetic and ergonomic improvements and concepts could be made in parallel. Aesthetic features could also be tested to measure potential drawbacks it might impose on aerodynamics. The topology optimization could run by itself in parallel with this development, freeing up valuable time.

Evaluation

The iterations created several improvements and knowledge of important factors for reducing drag, lift and frontal area. Some of the major changes was sharpening the rear part of the vehicle, and lowering frontal area further by modeling the surfaces as close to the minimal

volumes as possible. More air was also directed to flow underneath the car, reducing unwanted downforce.
The topology optimization reveled a more defined structure of giving more information of the structural aspects.

Influence
Utilizing the space behind the driver was seen as a possible structural advantage. Moving the driver forward, for both aero and structural advantage, and seeing that the t-spline needed more information points to create aesthetic aspects.

4.3.4 The final iterations

In the following iterations throughout the project the same approach was used when developing the vehicle. As going through the whole process is long, and the essence of how the process was implemented in this manner can be exemplified by the first iterations, a short overview is presented for the final stages. Before presenting the results of the process. As the Final T- spline was done, the final stages of the iterations revolved around the final topology optimizations and several free size, shape and ply optimizations. The employment of over 20 iteration of aerodynamic simulations and optimizations was possible with the interaction of T-spline and Open Foam, eventually creating the final product.

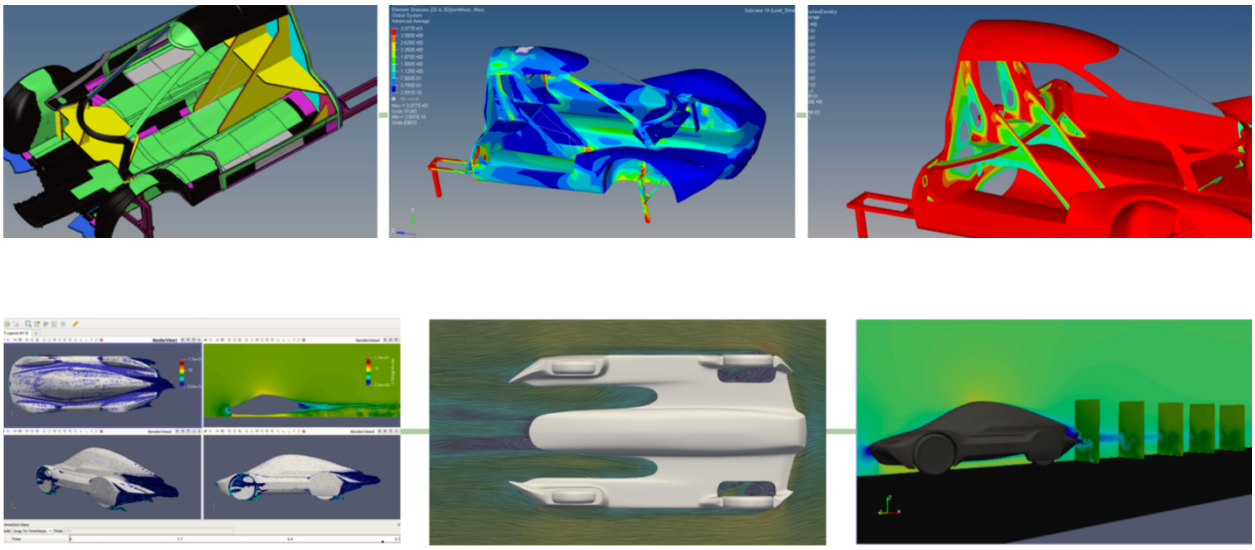


Figure 25: overwiew of the final iterations of the project

5 Discussion

5.1 Monocoque development and results

With the goal of creating a car that could excel in Shell-eco Marathon, aiming to win both the Battery-Electric award and the Vehicle Design Award, the team was overall satisfied with the results. Substantial improvements from earlier vehicles was made in all of the three major objectives of the development; that of structure, aerodynamics and design.

The weight of the monocoque was reduced by as much as 45% from the previous vehicle, with a weight of 22kg of the structural part of the monocoque, and 27kg including doors, hood and rear hatch. The total weight of the car came down to 72kg compared to the previous car with a weight of 85kg. Only one car have been lighter in the 12 years the organization have existed; the first vehicle developed with a weight of 69kg. However, with a larger budget sponsored by Petter Stordalen, the 2008 team where able to develop the vehicle using prepreg, which is much lighter as it contains less resin. Had the monocoque been developed with this method, the complete monocoque would have had a weight closer to 16kg, and changes in rules over the years that impose weight on the car (such as additional an additional door and stricter structural requirements). Overall, the structural development made the car one of the lightest in the battery-electric class, with the lightest car weighing around 68kg (TIM UPS-INSA).

In terms of aerodynamics of the previous cars built by FF, the 2018 model was by far the best of the vehicles with a Cd*A of around 0.148. This was one of the major reasons of the car taking home a 2. place in SEM 2018. With a reduction in both frontal area and drag coefficient, the new design pushed this value even further down with a Cd*A close to 0.111, leading to a 25% reduction in aerodynamic losses. Only one other car (the SZEnergy form Hungary) was considered to have a lower number in the class, with a Cd*A of somewhere between 0.09 and 0.1. But as this car was also in the heavy range of competitors weighing about 90kg, and was not considered to be the strongest opponent.

The overall best vehicle in the SEM Battery-Electric class from the previous year was considered to be TIM UPS-INSA from France. Being only 7% behind the them in the last year's competition, there was a good chance of beating them with the overall 25% reduction in aerodynamic losses and 15% reduction in total weight. This would also mean setting a world record, making it the world's most energy efficient car developed. On paper this seemed likely, as the advantage the FF2019 car had in aerodynamics was greater than the advantage TIM had in weight. Both cars also employed a similar drive chain and engines, and had similar electrical losses. However, the real world gave a different result than hoped for. Of the 4 attempts to set the best time, the team only had two valid attempts. In the first attempt, the gearing system was not working properly, and the front braking tubes was hitting the valve of the tires in all corners, leading to substantial energy loss. In the 2nd attempt, scratching noises where heard when cornering, and

it was discovered after the race that the steering arm in the front suspension was grinding against the rim. Additionally, the rear wheels was found to be 3 degrees out of alignment. Although these issues where fixed, a bad decision led the team to a failed 3rd and 4th attempt. A win or loose mentality led to the idea of employing a newly developed gearing system that would be highly efficient compared to TIMs. However, this system was untested before the race. Although it seemed to work on the test track, the car had to cancel the actual race after two rounds due to technical issues. These issues was not fixed, and the time for the last attempt was to short to be able to use the old system. Yet, with the two slightly flawed attempts, DNV GL Fuel Fighter ended up with a 5th place in the overall race, with 18 cars participating. Although the results was not as high as hoped, the development of the structural and aerodynamic objectives can be considered to be very successful. With more time for testing and fine tuning, the car should therefore have a good chance of a strong position in SEM2020.

When it comes to the design of the vehicle, holding qualities such as aesthetics and ergonomics, the successfulness of the development is difficult to quantify. The Vehicle Design Award is also not only a measure of these qualities, but the overall engineering design and process of development. The award is therefore also seen as a measure of how well the car is developed with regards to energy efficiency and environment, and consequently the materials utilized and the structural and aerodynamic development.
With around 40 cars combined from each of the energy types in the UrbanConcept class, the Sell eco-Marathon jury granted DNV GL Fuel Figher with the first place in Vehicle Design. This was a major achievement, and gave the team better confidence that the work had paid off, especially since this had been a goal from the very beginning of the project. The Vehicle Design Award was granted with the following statement from Shell about the teams effort:

Top-class chassis optimisation and reworking of the internal structure, supported by physical testing of carbon composite test specimens. Quality of both report and vehicle is exceptional. Wooden steering wheel and dashboard a clear demonstration that sustainability had been thoroughly considered with a robust design process.

In essence, these results of the development as a whole is the answer to the third research question of this thesis: *Can implementation of generative design over traditional methods create a competitive advantage in the development of DNV GL Fuel Fighter 5?*
It is of course not evidential that the implementation of generative design was the cause of the good results, or how much it can be credited the technology or the method, as a traditional approach with the same effort and team could yield similar or better results. This is impossible to know, however a good indication of its effectualness is the comparison with previous teams in FF. In terms of improvement it partook the biggest leap within all the major objectives from previous teams, and the vehicle also had the strongest position in its first competition. Although experiencing technical issues under the competition, the comparison of the vehicle specifications to other contestants is also a good indication, as none of these are known to employ generative design in this manner.

5.2 Methodology

The first research question of this thesis was exploring of how generative design has been implemented in product development before. Some of the theories closest to modeling this implementation was presented described in the theory chapter, but as also described are hard to find, especially since so many theories are focused on the technical side of implementing algorithms for specific cases. Ideas that could be helpful form related field such as parametric modelling and Knowledge Based Design was also presented, but is generally more oriented towards reuse of entities or the construction of generative systems rather than the implementation of generative design. Although many other related earlier fields could be brought into the study, such as Multi objective and Multivariate Optimization, the general view of the author is that very few studies tackles the general implementation of generative design into product development. Few was also found to draw relations to former design theories, as most are focused on the use of the tool rather its adoption and ability for changing the design process.

The second research question has been the overall goal of this thesis, trying to figure out what impact generative design as a methodology have on the process of engineering design and problem solving. Both the development of the methodology and the implementation of this methodology and generative systems in developing the monocoque has been an attempt to answer this question for the reader.
The thesis create a methodology for implementation of GD in product development with the basis that limited knowledge exist within this field. Hence, it is problematic to discuss its validity in relation to literature in other ways than have been done and discussed throughout the paper. In that sense, it is difficult to address the two major concerns described by Hans Grabowski in the establishment of the Universal Design Theory, the problem of *universality* and the problem of *applicability in industrial practices.*
What one could discuss is the approach that has been conducted. The first obvious complication is that of reducing the problem to the consideration of a single component of GD. Using the workings of the basic GA for describing GD is a huge simplification, and could therefore be a wrong step in relation to implementation of GD. However, as discussed in previous section, this simplification is done on several premises that explain the central role of GAs. Further, the notion that the ways in which humans solve problems are affected by the tools we use in a manner similar to the workings of the tool itself might be a wrong assumption for GD. Although one could see that this is the case for several other tools now used, such as CAx systems, it might not be true for GD. It could be that a completely unfamiliar approach to human problem-solving then that of GD is needed to utilize these systems to the fullest. This would be highly difficult to foresee before these systems have been applied more in industry. Nevertheless, if this notion holds, one could also question the way the similarities are drawn. First using the steps of human problem-solving and GA for creating a framework and level of abstraction, and further using this framework for seeing similarities in of GA and DTM might be a faulty approach. Completely different approaches could have been conducted, such as flipping the process altogether, beginning with a specific DTM as ground for the abstraction, or

grounding the process in the high level interaction of generative systems rather than algorithms.
The use of literature can also be questioned. Although the use of acknowledge papers have been stressed, seeing that little is written on the guiding objective of this thesis, also articles with few citations have been used to describe certain concepts. The bibliography also lacks a more nuanced view of the DTM field, as the work of Tetsuo Tomiyama has been the major source of comparison. The author also acknowledges that with limited experience of using GD tools, as well knowledge of the DTM field in general, creating a theory for implementation of GD in New Product Development might be to bite off more than one can chew.
Finally, the complete approach itself, namely using literature to create a theory as a base before implementing GD in Fuel Fighter might not be a good method. It might create a biased view that could have implications for seeing what really should be the emphasized when implementing GD. Another approach would therefore be to not research the implementation before using the technology (although that boat has gone), and go into the process blindly to see what drives the process. Consequentially, limiting this biased view is crucial to gain new knowledge from the process in the future work of utilizing these programs.

-

5.3 Further work

There are numerous theories and methodologies one could bring into this work, as well as deeper understanding of GD, to draw better analogies. Several other theories were originally intended to take part in the paper, such as C-K Theory and Emergent Synthesis, as well as other notions around GA such as Tree structure representations, and human-, and interactive-based GA. However, at a point, creating a more solide model in terms of literature becomes absolete to that of testing the model. Future work will therefore be to utilize the theory in a critical manner, so that changes can be made to create an overall better theory. This can be done by direct implementation as was conducted with Fuel Fighter, but also trough applying the theory on already known cases of product development to give more understanding of its use compared to other models.

6 References

6.1 Articles and books

Agkathidis, Asterios. 2015. “Generative Design Methods,” 9.

Becker, Markus C., Pasquale Salvatore, and Francesco Zirpoli. 2005. “The Impact of Virtual Simulation Tools on Problem-Solving and New Product Development Organization.” *Research Policy* 34 (9):

Bhooshan, Shajay. 2017. “Parametric Design Thinking: A Case-Study of Practice-Embedded Architectural Research.” *Design Studies*, Parametric Design Thinking, 52 (September): 115–43.

Boussaïd, Ilhem, Julien Lepagnot, and Patrick Siarry. 2013. “A Survey on Optimization Metaheuristics.” *Information Sciences*, Prediction, Control and Diagnosis using Advanced Neural Computations, 237 (July): 82–117.

Brown, Polly. 2009. “CAD: Do Computers Aid the Design Process After All?” *Intersect: The Stanford Journal of Science, Technology, and Society* 2 (1): 52–66.

Fernandes, Sierra, and Rita Margarida. 2003. “Generative Design a New Stage in the Design Process.” 2003.

Gábor, Rennera, and Ekártab Ekártab. 2003. “Genetic Algorithms in Computer Aided Design.” 2003.

Gen, Mitsuo, and Runwei Cheng. 2000. *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons.

Goldberg, D. E. 1991. “Genetic Algorithms as a Computational Theory of Conceptual Design.” In *Applications of Artificial Intelligence in Engineering VI*, 3–16. Springer, Dordrecht.

Goldberg, David E. 2000. “The Design of Innovation: Lessons from Genetic Algorithms, Lessons for the Real World.” *Technological Forecasting and Social Change* 64 (1): 7–12.

Goldberg, David Edward. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company.

Gulanova, Jana, and Miroslav Vereš. 2014. *COMPUTER AIDED GENERATIVE DESIGN OF AUTOMOTIVE SHAPED COMPONENTS*.

Hayes, John R. 2013. *The Complete Problem Solver*. Routledge.

Isaksson, O. 2003. “A GENERATIVE MODELING APPROACH TO ENGINEERING

DESIGN." *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*.

Janssen, Patrick, John Frazer, and Tang Ming-xi. 2002. "Evolutionary Design Systems and Generative Processes." *Applied Intelligence* 16 (2): 119–28.

Jebari, Khalid. 2013. *Selection Methods for Genetic Algorithms*. Vol. 3.

Krish, Sivam. 2011. "A Practical Generative Design Method." *Computer-Aided Design* 43 (1):

Leon, Noel. 2009. "The Future of Computer-Aided Innovation." *Computers in Industry*, Computer Aided Innovation, 60 (8): 539–50.

.

McCormack, Jon, Alan Dorin, and Troy Innocent. 2004. "Generative Design: A Paradigm for Design Research," 9.

Nordin, Axel. 2018a. "Challenges in the Industrial Implementation of Generative Design Systems: An Exploratory Study." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing : AI EDAM; Cambridge* 32 (1): 16–31.

———. 2018b. "Challenges in the Industrial Implementation of Generative Design Systems: An Exploratory Study." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing : AI EDAM; Cambridge* 32 (1): 16–31.

Othmer, C., David E. Manosalvas-Kjono, Antony Jameson, and Juan J. Alonso. 2017. "Aerodynamic Topology Optimization: Some Observations on Hysteresis in Separated Flows." In .

Othmer, Carsten. 2014. "Adjoint Methods for Car Aerodynamics." *Journal of Mathematics in Industry* 4 (1): 6.

Othmer, Carsten, and Torsten Grahs. 2014. "Approaches to Fluid Dynamic Optimization in the Car Development Process."

Richter, Hendrik. 2010. "Evolutionary Optimization and Dynamic Fitness Landscapes." In *Evolutionary Algorithms and Chaotic Systems*, 409–46. Studies in Computational Intelligence. Springer, Berlin, Heidelberg.

Rose-Anderssen, C., P. M. Allen, C. Tsinopoulos, and I. McCarthy. 2005. "Innovation in Manufacturing as an Evolutionary Complex System." *Technovation* 25 (10): 1093–1105.

Rothlauf, Franz. 2006. *Representations for Genetic and Evolutionary Algorithms*. Gesellschaft für Informatik.

Sapossnek, Mark, and Carnegie Mellon University Engineering Design Research Center. 1991. "Research on Constraint-Based Design Systems." *Department of Electrical and Computer Engineering*, January.

Sastry, Kumara, David E. Goldberg, and Graham Kendall. 2014. “Genetic Algorithms.” 2014.

Shea, Kristina, Robert Aish, and Marina Gourtovaia. 2005. “Towards Integrated Performance-Driven Generative Design Tools.” *Automation in Construction*, Education and Research in Computer Aided Architectural Design in Europe (eCAADe 2003), Digital Design, 14 (2): 253–64.

Sim, Siang Kok, and Alex H. B. Duffy. 2003. “Towards an Ontology of Generic Engineering Design Activities.” *Research in Engineering Design* 14 (4): 200–223.

Singer, David J., Norbert Doerry, and Michael E. Buckley. 2009. “What Is Set-Based Design?” *Naval Engineers Journal* 121 (4): 31–43. https://doi.org/10.1111/j.1559-3584.2009.00226.x.
Singh, Tapan Kumar. 2016. “Constrained Self-Adaptive Genetic Algorithm.” *SeMA Journal* 73 (3)

Singh, Vishal, and Ning Gu. 2012. “Towards an Integrated Generative Design Framework.” *Design Studies* 33 (2): 185–207. https://doi.org/10.1016/j.destud.2011.06.001.
Stanković, T, M Stošić, and D Marjanović. 2006. “EVOLUTIONARY ALGORITHMS IN DESIGN.” 2006.

Steinberg, Robert J., and Janet E. Davidson. 2003. “The Psychology of Problem Solving.” 2003.

Tan, Hao, Chunhui Jing, Danhua Zhao, Fangzhen Zou, and Jianghong Zhao. 2013. “Using Interactive Genetic Algorithm to Generate New Vehicle Styling Brand Elements with Feature Lines,” 12.

Tomiyama, T., P. Gu, Y. Jin, D. Lutters, Ch. Kind, and F. Kimura. 2009. “Design Methodologies: Industrial and Educational Applications.” *CIRP Annals* 58 (2): 543–65.

Tomiyama, Tetsuo. 2006. “A Classification of Design Theories and Methodologies,” January,

Türkmenoğlu, Saliha. 2015. “Paradigm Shift in Industrial Product Design: Generative Design.”

Vavak, F., and T.C. Fogarty. 1996. “A Comparative Study of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments.” 1996.

Yu guoyan, Wang Xiaozhen, and Li peng. 2009. “A Constraint Based Evolutionary Decision Support System for Product Design.” In *2009 Chinese Control and Decision Conference*, 2585–

6.2 Figures

Figure 1: https://www.flickr.com/photos/shell_eco-marathon/48173313512/in/album-72157709142796577/

Figure 2: DNV GL Fuel Fighter

Figure 3: Rothlauf, Franz. 2006. Representations for Genetic and Evolutionary Algorithms. Gesellschaft für Informatik. http://carlosreynoso.com.ar/archivos/rothlauf.pdf.

Figure 4: https://www.researchgate.net/figure/Genotype-phenotype-mapping-and-fitness-function-in-evolutionary-design_fig3_226912968

Figur 5: https://www.semanticscholar.org/paper/Active-modules-identification-in-multilayer-Li/1f1411a1e1b3f5170688fa68725c5dfd15f48546

Figur 6: https://medium.com/nyc-design/why-optimization-is-not-generative-design-bc8ac3afddef

Figur 7: Isaksson (2003)

Figur 8: Guoyan et al. (2009)

Figure 9-15: Created by author based on other theories and models of GA

Figure 16-: DNV GL Fuel Fighter

6.3 Webpages

(1) https://en.wikipedia.org/wiki/Generative_design

(2) https://www.autodesk.com/solutions/generative-design

(3) https://en.wikipedia.org/wiki/Hyper-heuristic

(4) https://en.wikipedia.org/wiki/List_of_genetic_algorithm_applications