28th annual INCOSE international symposium
Washington, DC, USA
July 7 - 12, 2018

Automated Component-Selection of Design Synthesis for Physical Architecture with Model-Based Systems Engineering using Evolutionary Trade-off

Habibi Husain Arifin
No Magic Asia
+62 – 2130052646
hharifin@bmaptech.com

Ho Kit Robert Ong
No Magic Asia
+66 - 27171117
robert_ong@nomagic.com

Nasis Chimplee
No Magic Asia
+66 – 868881782
nasis.c@nomagic.com

Jirapun Daengdej
Assumption University of Thailand
+66 – 818288523
jirapun@au.edu

Thotsapon Sortrakul
Assumption University of Thailand
+66 – 818286111
thotsapon@scitech.au.edu

Abstract. Component-Selection is an important task in design synthesis of MBSE. A trade study is commonly used to help systems engineers and stakeholders selecting the components of a systems design. A simple analysis may be sufficient when it involves only two parameters. However, when the components and their integration become more complex, the trade study also becomes harder, time- and cost-consuming, and error-prone. This paper aims to propose a method to automatically generate the solution by performing an evolutionary search. Sample components of a hybrid car which consists of an engine, an electric motor, and a battery are used in our initial prototype. The logical architecture is represented in the OMG SysML™ via CSM™. Through the experimental result, this paper shows that the proposed technique allowed the system design to be efficiently selected.

Introduction

In Model-Based Systems Engineering (MBSE), design synthesis is a fundamental engineering process that includes the generation of physical architecture specifications that satisfy the logical design and desired functional specifications (Kerzhner & Paredis 2009). One of the tasks in design synthesis is the component selection. A simple analysis with trade study generally may overcome the problem when a few parameters are involved, despite it may be insufficient to solve today's systems engineering problems as the number of components and their integration in the systems are becoming more complex, e.g., aircraft systems and passenger car systems.

In the current practice, the systems engineering process requires activities to establish the main goals of a system, specify the system requirements, synthesis a solution space with the possible alternative designs, and evaluate the alternatives to find a set of solution from the solution space (Friedenthal, Moore & Steiner 2015). Meanwhile, the systems complexity complicates the process to drag out the best alternatives from the solution space. Searching through large number of possibilities is often time- and cost-consuming (Dinger 1998), and error-prone (Branscomb et al. 2013). Thus, an optimization of the searching method is needed to overcome this issue (Dinger 1998).

Heuristic algorithms, e.g., Genetic Algorithms (GAs) (Goldberg 1989) have been applied successfully to many engineering problems, e.g., electromagnetic systems design and aircraft control/ aerodynamics (Winter et al. 1996) (Harman & Jones 2001). These potentials facts have triggered some scholars to complement GAs and design synthesis together for the success of systems engineering (Cagan et al. 2005). In the book *Engineering Design Synthesis*, (Chakrabarti 2002) presents a survey and detailed investigation of the potential applications of Genetic Programming in a design synthesis. One of them is to generate a pattern of solution which is represented in the Unified Modeling Language by Object Management Group (OMG UML™) (Chakrabarti 2002). OMG

Systems Modeling Language (SysML™ is an extension of the OMG UML™ (Kerzhner & Paredis 2009) (Vanderperren & Dehaene 2005) which focuses on the perspective of systems engineering.

This paper proposes a method to perform an automated synthesis to select the components of the physical architecture of an MBSE design. The searching and selection method is implemented by embedding a GA to the OMG SysML™. The GA represents a systems design as a chromosome where the genes of each chromosome are the features of the design. The contribution of this preliminary investigation is to show how the proposed technique can help systems engineers to automatically generate a number of possible designs, and, in addition, assist them using the best fitness solution based on the fitness value, which can be pre-defined by human experts of the domain.

Background

Overview of Genetic Algorithms and Its Advantage. Genetic Algorithms (GAs) is inspired by Charles Darwin's Theory of Evolution (Hermawanto 2013), which illustrates the natural biological systems evolution and its natural selection (Dinger 1998). Computer scientists have adopted this approach as a metaheuristic searching algorithm to solve optimization problems (Zou et al. 2015). The searching method of GAs is mainly based on randomization with natural selection (Meffert 2017). Natural selection means that the fittest individual will survive.

The beginning process of GAs starts with a sample set of potential solutions (initial population) (Meffert 2017). It continues when two parent chromosomes in the population mate by sharing their genetic information. This mating process (crossover or recombination) produces the offspring by having half genes from a parent and another half genes from the other. Meanwhile, gene mutations can occur when copying the parent's genes and cause the genes of the new offspring are slightly different from their parent. In the end, the survivor is measured by the fitness of each potential offspring (Chakraborty 2010). The definitions of "population", "phenotype", "genotype", "chromosomes", "gene", and "allele" used in this paper refer to (Wilhelmstotter 2017) (Meffert 2017) (Chakraborty 2010) (Frye 2017).

There are several advantages of GAs than the traditional searching and optimizing algorithms: ***(1)*** *GAs can be used in a wide range of applications and to perform searching in a complex solutions space* (Lazko 2006)*; and* ***(2)*** *GAs are less likely to be led astray by the local optima because they take the advantage of an entire set of solutions spread throughout the solution space* (Meffert 2017).

Overview of OMG Systems Modeling Language (OMG SysML™) and Its Advantage. A system can consist of components wrapped into several modules or sub-systems, which are interconnected in order to perform a specific function that is not sufficient to do by the components alone (Austin 2012). In systems engineering, a system can be defined as *"An integrated set of elements, sub-systems, or assemblies that accomplish a defined objective (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements (INCOSE)"* (Walden et al. (eds.) 2015) (International Council on Systems Engineering (INCOSE) 2015).

The entire design of the prototype is represented in the OMG SysML™. The reasons are:

- It supports the design of many different elements thus it reduces the use of many diverse tools and representations (Kerzhner & Paredis 2011) (Kerzhner & Paredis 2011).
- It allows the expression of relationships between different facets of the problem (Kerzhner & Paredis 2011) (Kerzhner & Paredis 2011).
- Designing with the OMG SysML™ can maximize the reuse of models. It allows designing the problem, reusing the existing models, and gathering final architectures without having to rebuild them (Albarello, Welcomme & Reyterou 2012).
- It has its formal semantics that can be understood and interpreted by both human and machine. Therefore, it can mitigate misunderstanding between them (Graves & Bijan 2011). With the right ontology, the machine can even understand the design context and content as well.

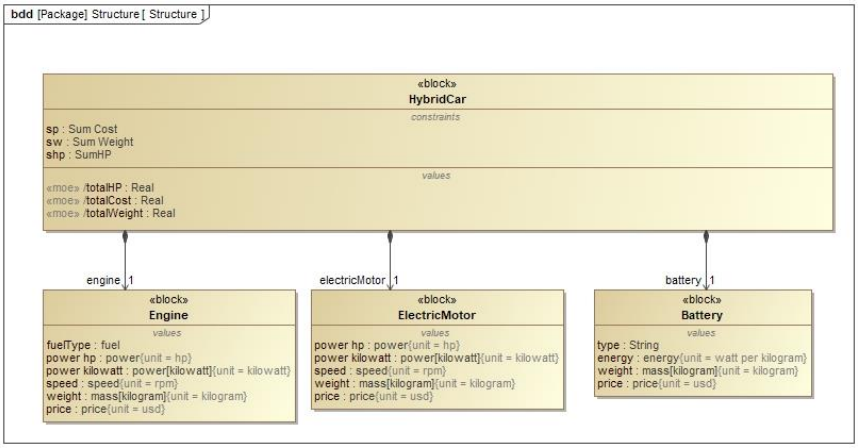

Figure 1. Example of Hybrid Car Structure

Overview of Cameo Systems Modeler. Cameo Systems Modeler (CSM™) is an OMG standard compliant modeling tool for the OMG SysML™ and UML™. CSM™ supports both the OMG SysML™ diagrams and many engineering features such as traceability, impact analysis, and trade study to enable MBSE activities. (Friedenthal, Moore & Steiner 2015). There are three main diagrams used in the prototype, i.e. Requirement Diagram, Block Definition Diagram (BDD), and Parametric Diagram (Cameo Systems Modeler: User Guide 18.1 2015).

Example of Application: Hybrid Car. This paper proposes a solution for the component-selection problem in the instance level of a hybrid car model. Figure 1 shows an example of a hybrid car structure in a BDD. Since the main objective of this preliminary research is to show how the proposed technique can be applied in a design synthesis problem, only the conceptual level design and a set of simplified formulas for calculating the solution with basic parameters, i.e., total horsepower, total cost, and total weight are considered. Any complex formula can be used instead of the simplified version in the future works, depending on the domains and industries. The basic parameters are used to evaluate the fitness value.

A hybrid car must consist of an engine, an electric motor, and a battery. Each component has its own attributes where some of them refer to the specifications of Toyota Prius (Toyota Prius User-Guide: 2nd Edition for The 2010-2012 Models 2012). The details of each attribute are explained as follows:

Table 1: Detailed Attributes of Hybrid Car

Component	Attribute	Unit	Definition
Engine	fuel	-	Fuel type of an engine, i.e., gasoline and diesel
	power hp	hp	Power of an engine in horsepower (hp)
	power kilowatt	kW	Power of an engine in kilowatt (kW)
	speed	rpm	Rotational speed of an engine in revolutions per minute
	weight	kilogram	Mass of an engine
	price	US$	Price of an engine
Electric Motor	power hp	hp	Power of an electric motor in horsepower (hp)
	power kilowatt	kW	Power of an electric motor in kilowatt (kW)
	speed	rpm	Rotational speed of an electric motor in revolutions per minute
	weight	kilogram	Mass of an electric motor
	price	US$	Price of an electric motor
Battery	type	-	Cell type of a battery, i.e., NiMH and Li-Ion
	energy	W/kg	Energy of a battery pack in watt per kilogram
	weight	kilogram	Mass of a battery
	price	US$	Price of a battery

Literature Review

The Failures of Design Synthesis. The main objective to use the OMG SysML™ is to define the system specifications. Since the safety-critical systems like those of a passenger car and an airplane are very complex, they all require a well-defined specification before the design and development (Spyropoulos & Baras 2013). Meanwhile, industries still need to make a decision faster without much modeling effort (Trcka et al. 2011). Due to this time constraint and large solution space, systems engineers are forced to only focus on a limited set of design alternatives (Albarello, Welcomme & Reyterou 2012) to keep their products competing in the business market.

Component-Selection for Physical Architecture in Design Synthesis. Component-selection is a process of design synthesis to select a set of feasible components from a component library (catalog), which satisfies the requirements of the logical architecture. Today, the complexity of systems is increasing dramatically compared to a decade ago. It is accompanied by the increasing of components number and the difficulties of components dependency (Spyropoulos & Baras 2013). The two emerged problems caused by this issue (Nassar & Austin 2013): ***(1)*** *The problem to perform an extensive searching to discover the component combination(s) through many possibilities; and* ***(2)*** *The problem when there is no feasible combination exists that can satisfy the requirements.*

This paper focuses mainly to solve the first problem by designing a decision support system to assist the systems engineers and stakeholders performing the component-selection process with less efforts and resources. A random-based searching is used to find the best alternative solution among many design alternatives generated automatically by the system (Albarello, Welcomme & Reyterou 2012). Some related studies have been conducted by (Nassar & Austin 2013) (Leserf et al. 2015). As illustrated in (Nassar & Austin 2013) (Nassar 2012), the design alternatives are assembled from a components library and evaluated by using a trade study analysis. The system level architecture is assumed to be fixed. The difference to the previous studies is this study is attempting to show the usability of a GA in the OMG SysML™ to perform the trade study analysis.

Traditional Trade-off vs. Evolutionary Trade-off. Trade-off is an essential part in a MBSE design synthesis (Spyropoulos & Baras 2013). It means "*the decision-making actions that select from various requirements and alternative solutions on the basis of net benefit to the stakeholders*" (International Council on Systems Engineering (INCOSE) 2015). Trade study analysis or trade-off study can help in selecting the fittest configuration to the specified criteria among many possible solutions (No Magic Documentation 2017). In this study, trade study is based upon a systematic comparison of the feasible hybrid car's designs measured with respect to performance, cost, and weight. The trade-off study is made among the different possible physical architectures relative to the requirements and the logical architecture (Ackva 2013). Compared to the traditional practice, a trade-off is commonly done by simply performing an extensive evaluation to all potential solutions. However, a complex system comprises almost infinite number of traits and a trade-off requires multiple traits in analysis (Evolution News 2014).

Genetic Algorithms in Design Synthesis

Genetic Algorithms in Design Synthesis. The detailed process of GAs in design synthesis illustrated in Figure 2 is as follows (Hermawanto 2013):

1. Determine the number of genes within a chromosome, the size of the population, the number of evolutions, and the values of mutation and the crossover rate. The size of the population can vary according to the complexity of the problem to be addressed (Zou et al. 2015).
2. Encode the library of component catalogs to the represented genes to form the chromosomes. Then generate the initial population randomly.
3. Evaluate the fitness value of the chromosomes by calculating the objective function.
4. Select the best chromosomes to become the parent for the next generation.
5. Perform the crossover and mutation operation with the specified rates to produce new offspring.
6. Repeat steps 3 to 5 until the maximum number of evolutions is reached.

7. Select the best chromosome from the last generation.
8. Decode the chromosome to the solution, in this case, is a hybrid car with an engine, an electric motor, and a battery.
9. The hybrid car can be used to support the decision making.
10. Optimize the GA, if it is needed by specifying a new configuration.

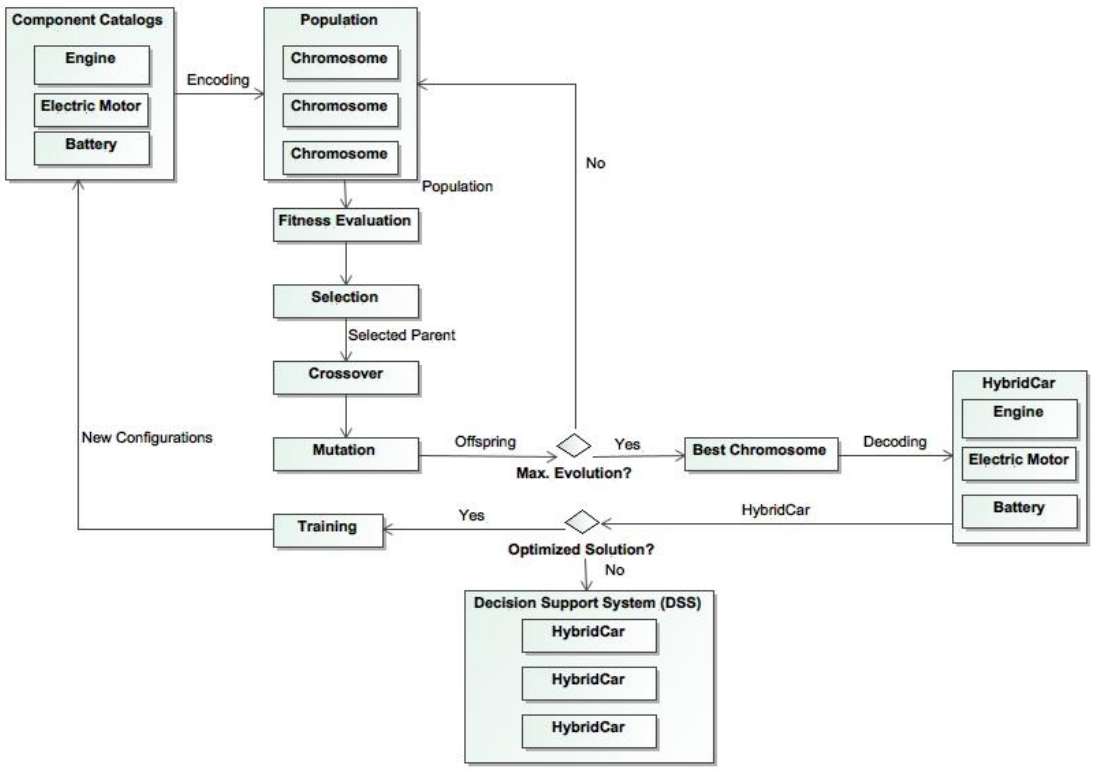

Figure 2. Genetic Algorithm in Design Synthesis

According to the design, the model requirement parameters and the MOE are used to identify the best optimized value or the Fitness Value. Figure 3 shows that the evolutionary trade-off also needs a model of fitness function to perform the natural selection of GAs. Figure 3 depicts the additional model for a HybridCar and its components. The GA parameters provide the code and the gene value for each Battery, Engine, and Electric Motor required by the GA to do the evaluation. The Trade Study Analysis block uses the Fitness Function to analyze the solution. The Fitness Function block represents the model of formula of fitness value and also includes the parameters required by the GA.

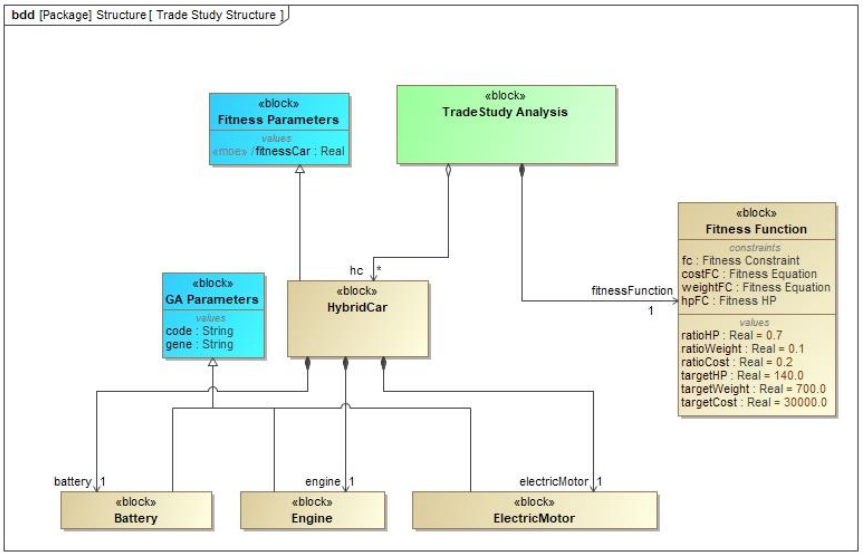

Figure 3. Trade Study Structure model with GA's parameter and Fitness Function

This paper presents seven key steps in designing GAs such as the representation and encoding technique; initial population (*how the first generation is generated*); selection method (*how to select parent individuals to be involved in reproduction*); crossover operator (*how to produce an offspring from two parent chromosomes*); mutation operator (*how to mutate an offspring*); (Zou et al. 2015) and training and guidance (*how to optimize the GA*).

1. Representation and Encoding

The representation specifies the range of candidate solutions that can be generated (Cagan et al. 2005). Before the use of a GA, encoding the potential solutions to variables called chromosomes (Dinger 1998) is necessary to let computers understand the process (Chakraborty 2010). In the initial prototype, value encoding is performed in which each chromosome is a sequence of some values, e.g., Chromosome A [ABDJEIFJDHDIERJFDLDFLFEGT] (Chakraborty 2010) (Obitko 1998). Value encoding is used because: ***(1)*** *The type of problem is difficult to solve with binary encoding* (Chakraborty 2010). *In value encoding, the values can be anything related to the specific problem, e.g., real number* (Chakraborty 2010) *or alphabetical characters* (Obitko 1998). *Thus, it is easier to encode and decode the gene of a solution, such as A → Engine A, B → Engine B;* ***(2)*** *Permutation encoding is mainly used to solve ordering problems, e.g., traveling salesman problem* (Chakraborty 2010)*; and* ***(3)*** *Tree encoding is mainly used for evolving programs or expressions* (Chakraborty 2010). Table 2, Table 3, and Table 4 below show the examples of the catalogs of a component library. Each catalog contains a unique code, an encoded genetic code, and other specifications for each engine, electric motor, and battery respectively. The component library involves 20 alternative components: 10 engines (A-J), 5 electric motors (A-E), and 5 batteries (A-E). As illustrated in the tables, a row represents a hybrid car instance. The maximum number of potential solutions from its permutation equals to 250 possibilities.

Table 2: Example of Engine Catalog

#	Name	code : String	gene : String	fuelType : fuel	power hp : power (hp)	power kilowatt : power[kilowatt] (kW)	speed : speed (rpm)	weight : mass[kilogram] (kg)	price : price (usd)
1	Engine A	E-A	A	Gas	58.0	43.0	4000.0	240.0	7000.0
2	Engine B	E-B	B	Gas	70.0	52.0	4500.0	335.0	7200.0
3	Engine C	E-C	C	Gas	76.0	57.0	5000.0	375.0	7500.0
4	Engine D	E-D	D	Gas	98.0	73.0	5200.0	500.0	8000.0
5	Engine E	E-E	E	Gas	100.0	75.0	5400.0	540.0	9500.0
6	Engine F	E-F	F	Diesel	58.0	43.0	4000.0	607.0	8500.0
7	Engine G	E-G	G	Diesel	70.0	52.0	4500.0	625.0	10500.0
8	Engine H	E-H	H	Diesel	76.0	57.0	5000.0	685.0	11000.0
9	Engine I	E-I	I	Diesel	98.0	73.0	5200.0	699.0	11500.0
10	Engine J	E-J	J	Diesel	100.0	75.0	5400.0	720.0	12500.0

Table 3: Example of Electric Motor Catalog

#	Name	code : String	gene : String	power hp : power (hp)	power kilowatt : power[kilowatt] (kW)	speed : speed (rpm)	weight : mass[kilogram] (kg)	price : price (usd)
1	Electric Motor A	EM-A	A	40.0	30.0	2000.0	205.0	3000.0
2	Electric Motor B	EM-B	B	44.0	33.0	5600.0	216.0	4000.0
3	Electric Motor C	EM-C	C	67.0	50.0	6700.0	240.0	5000.0
4	Electric Motor D	EM-D	D	80.0	60.0	13500.0	255.0	6000.0
5	Electric Motor E	EM-E	E	90.0	67.0	13500.0	275.0	7000.0

Table 4: Example of Battery Catalog

#	Name	code : String	gene : String	type : String	energy : energy (watt per kilogram)	weight : mass[kilogram] (kg)	price : price (usd)
1	Battery A	B-A	A	NiMH	600.0	125.0	2000.0
2	Battery B	B-B	B	NiMH	900.0	110.0	2200.0
3	Battery C	B-C	C	NiMH	1250.0	99.0	2400.0
4	Battery D	B-D	D	NiMH	1250.0	110.0	2500.0
5	Battery E	B-E	E	Li-Ion	4400.0	176.0	4000.0

2. Initial Population

In the initial prototype, the initial population is randomly generated by a random number generator to simulate the nature of evolutions (Cagan et al. 2005). Potential designs from previous experience might be used as the initial population. The number of chromosomes in the initial population depends on the population size which is specified by the systems engineers.

3. Fitness Evaluation

The analogy comes from Darwin's theory of evolution "The strongest species that survives" (Hermawanto 2013). To this end, a fitness value is needed to define the quality of each gene in a population to perform a selection process (Chakraborty 2010) (Dinger 1998). The purpose is to evaluate how close an individual is to an optimal solution (Zou et al. 2015). Table 5 shows the example of a stakeholder needs. It defines the requirements of potential solutions pre-defined by the stakeholders. Suppose that they wish to build a hybrid car that satisfies a level of performance, a limited budget, and a weight constraint.

Table 5: Example of Stakeholder Needs Table

#	Id	Name	△ Text
1	SN2	R SN2 Cost	The cheapest cost to build a car, much cheaper than the target cost
2	SN1	R SN1 Performance (HP)	The greatest performance in term of HP, closest to the target HP
3	SN3	R SN3 Weight	The lightest car, much lighter than the target weight

Figure 1 shows the engine, electric motor, and battery of the hybrid car. In this process, the systems engineers have to find a solution based on the defined parameters, such as MOE (Sayanjali & Nabdel 2013). Figure 4 shows the requirements (*Target HP*, *Maximum Cost* and *Maximum Weight*) (Roedler & Jones 2005) are satisfied by the MOE: *totalHP*, *totalCost*, and *totalWeight* respectively. The development of the requirements starts with a statement of stakeholder needs and evolves into two levels of requirements:

1. The 1st level requirements are the initial requirements, e.g.:
   - SN1: The greatest performance in term of HP, closest to the target HP.
   - SN2: The cheapest cost to build a car, much cheaper than the target cost.
   - SN3: The lightest car, much lighter than the target weight.
2. The 2nd level requirements are the detailed requirements derived from the initial requirements. They represent the component-level requirements and cast as the constraints written in terms of the values of the component attributes (Nassar & Austin 2013), e.g.:
   - SR1: Target horsepower close to 140 hp.
   - SR2: The maximum cost is 30,000 US$.
   - SR3: The weight shall not be more than 70 kilograms.

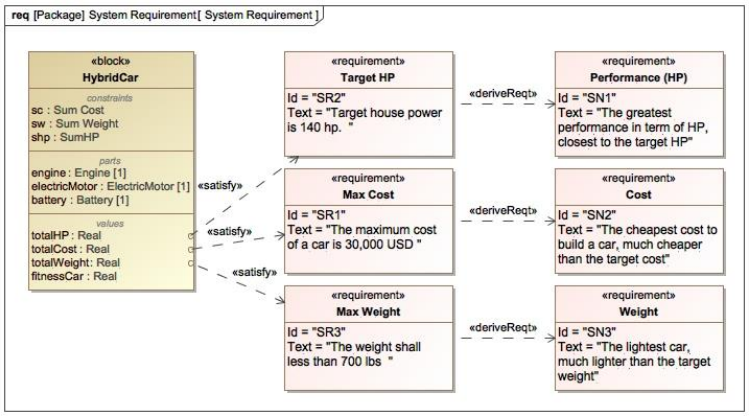

Figure 4. Requirement Diagram depicts the derive and satisfy relationship for System Requirement

Table 6 shows the example of the user requirements and the design parameters considered in the trade-off study: *Performance*, *Cost*, and *Weight*. The weighted ratio is used to define the weighted fitness function to evaluate the chromosomes.

Table 6: Example of User Requirements

Requirements	Value	Weighted Ratio
Target HP	140 hp	0.7
Maximum Cost	US$ 30,000	0.2
Maximum Weight	700 kilograms	0.1

Before measuring the fitness value, it is necessary to calculate the gained *horsepower (hp)*, *cost*, and *weight* of the potential solution in the selection process. Here are the examples of a hybrid car's performance calculations:

$$totalHP = hpEngine + hpElectricMotor$$
$$totalCost = enginePrice + electricMotorPrice + batteryPrice$$
$$totalWeight = engineWeight + electricMotorWeight + batteryWeight$$

Whereas:

- *totalHP is the total power of a solution in horsepower. The example formula for the initial prototype is copied from the example of trade study pattern with Cameo Simulation Toolkit* (No Magic Documentation 2017). *In the example, the total power of a HybridEngine is the sum of the power of the electricMotor and the dieselGenerator);*
- *totalCost is the total cost of a solution;*
- *totalWeight is the total mass of a solution.*

In this paper, a weighted fitness function is used to evaluate the solutions. A similar research by (Grosso et al. 2007) applied a combination of GAs, linear programming, evolutionary testing, and static & dynamic information using a weighted fitness function to identify tests that expose buffer overflows in programming code (Avancini 2012). The example of weighted GA's fitness function is as follows:

$$fitness = w1 \cdot \text{fitnessHP} + \text{w2} \cdot \text{fitnessCost} + \text{w3} \cdot \text{fitnessWeight}$$

Whereas:

- *fitnessHP is the fitness value of the performance of a solution in horsepower;*
- *fitnessCost is the fitness value of the cost of a solution;*
- *fitnessWeight is the fitness value of the mass of a solution;*
- *w1, w2, and w3 are real, positive weights, indicating the contribution of each fitness value to the overall fitness function whereas w1 + w2 + w3 = 1 (See Table 6).*

$$fitnessHP = MAX_{BOUND} - \frac{|targetHP - totalHP|}{targetHP}$$

If $(totalCost \leq maxCost)$ $\quad fitnessCost = \frac{|maxCost - totalCost|}{maxCost}$

Else $\quad fitnessCost = MIN_{BOUND}$

If $(totalWeight \leq maxWeight)$ $\quad fitnessWeight = \frac{|maxWeight - totalWeight|}{maxWeight}$

Else $\quad fitnessWeight = MIN_{BOUND}$

Whereas:

- *MIN_{BOUND} is the lowest bound of fitness value, 0.00;*
- *MAX_{BOUND} is the highest bound of fitness value, 1.00;*
- *targetHP is the target performance of a solution in horsepower;*
- *maxCost is the maximum cost of a solution;*
- *maxWeight is the maximum mass of a solution.*

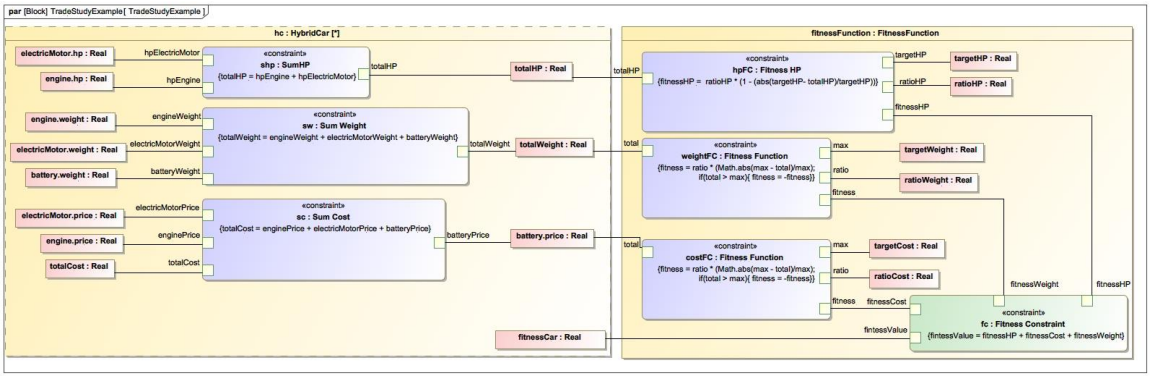

Figure 5. Example of Trade Study Analysis with Parametric diagram

The fitness function and all related equations for the GA can be represented in the OMG SysML™ model using constraint block. A constraint block is a specific kind of block, which a model practitioner creates as the reuse of a mathematical expression. It consists of two major properties, constraint property and constraint parameter. A constraint property is a mathematical expression reused in the system model, and constraint parameters are the variable definition usage in the expression. The definitions of mathematical expressions are defined in the BDD with the constraints block. To illustrate how those constraint blocks are used, the OMG SysML™ provides a Parametric Diagram (See Figure 5). A Parametric Diagram is an interconnection diagram that binds the values between the properties inside a Block. Normally a model practitioner uses a Parametric Diagram to bind the values between the Block properties and the constraint parameters to evaluate value analysis for the model. Figure 5 shows an example of how the variables of the fitness function and the related equations are bound to the other value properties or parameters.

4. Selection

Selection is also known as reproduction (Chakraborty 2010). It is applied to a population to find the best chromosomes to be the parents (Obitko 1998). From the population, the parent chromosomes are chosen to perform crossover and produce the potential offspring (Chakraborty 2010). In the initial prototype, the selection process is performed using the best chromosome selection, also known as elitism. Elitism is a selection method that copies the best chromosome of the previous population to the new one (Obitko 1998). In addition, the current fittest chromosome is always kept in the population (Eiben & Smith 2003). Elitism was introduced because the chance of losing the best chromosome is high when producing a new population by performing crossover and mutation (Obitko 1998). The default configuration of the initial prototype is an elitist ranking selector that copies the top 90% (0.90) of the specified population size (Hall 2017).

5. Crossover and Crossover Rate

Crossover is also known as recombination (Chakraborty 2010). It is like simulating the "biological mating" of two parent chromosomes by swapping and mixing their genes (Meffert 2017) to pass their genetic information to their offspring (Frey, Fittkau & Hasselbr 2013). The default configuration of the initial prototype is a one-point crossover (Getrost 2006). The one-point crossover locates a crossover point and then clones the whole things behind this point from the first parent chromosome and then the rest after the crossover point from the second parent chromosomes (Chakraborty 2010) (Obitko 1998). The crossover point is chosen randomly with a probability rate of 0.35 of the specified population size (Hall 2017).

6. Mutation and Mutation Rate

Mutation is the random changes of the gene values in the chromosomes of a potential solution (Meffert 2017). The changes are mainly caused by errors in copying genes from the parent chromosomes (Obitko 1998). Once crossover helps the parent chromosomes produce their successors, the mutation is applied to each successor. By applying mutation, it can help retain the diversity of the individuals in the whole population (Frey, Fittkau & Hasselbr 2013) (Chakraborty 2010) (Meffert 2017). The default configuration of the initial prototype is a custom mutation for the string-typed gene at a rate of 12. It means that the mutation is applied to 1 in 12 genes in the whole population. It is often necessary to develop a custom type of mutation in value encoding (Chakraborty 2010). The mutation is simply done by performing a change at the random point with another permitted string value. The rate is dictated by the size of the chromosome multiplied by the size of the population divided by the rate (Hall 2017). The probability of mutation rate is 1/12.

7. Training for Optimization

The final task in the process of a design synthesis is to provide feedback to the system and to discover an approach to produce better solutions. In the initial prototype, the GAis trained by reconfiguring its parameters, e.g., the value of population size, evolutions number, selection rate, crossover rate, and mutation rate.

GAs Plugin in Cameo Systems Modeler (CSM™) using MagicDraw™ Open API. A GA plugin is integrated with CSM™ via MagicDraw™ Open API. The integration of OMG SysML™ with a trade-off tool will allow the systems engineers to make decisions faster with more confidence (Spyropoulos & Baras 2013). Figure 6 depicts the overview design of the GA for the OMG SysML™ plugin. The GA Plugin reads the OMG SysML™ structured models from MagicDraw™ or CSM™ along with the required configurations i.e., the fitness value, population, or MOEs, and returns the result as an optimization design for the Evolutionary Trade-off System.

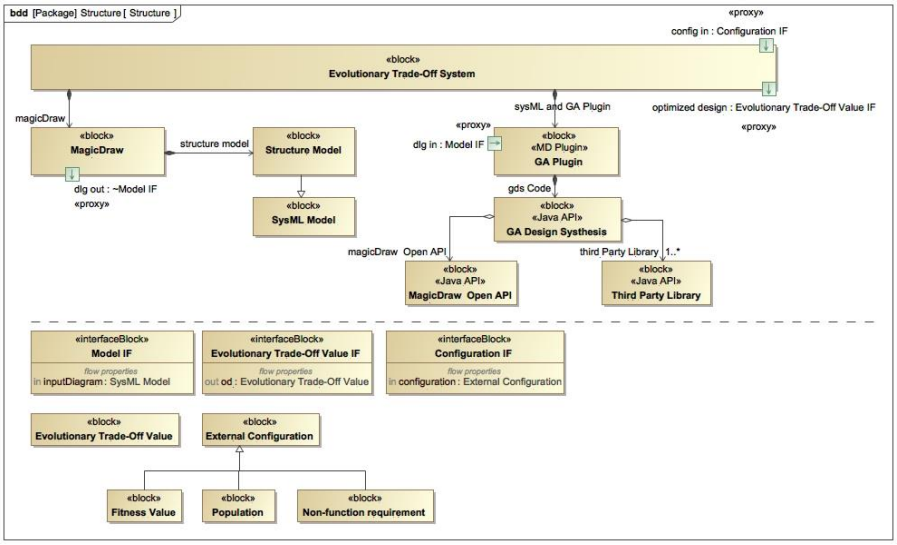

Figure 6. Genetic Algorithms for Design Support System of System Modeling

Results of Evaluation

Figure 7 shows the processes of an evolution of the GA. The parameters used in this experiment are: Population Size = 10; Number of Evolution = 20; Crossover Rate = 0.35; Mutation Rate = 12; and Selection Rate is 90 percent with Elitism selector. The advantage of Elitism selector is to keep the best alternative from the previous solution as a potential alternative for the next one. As seen in the figure, starting from the 7th generation, the best alternative is constantly hold by ADC.

No.	Population	Best Genes	Total Hp	Total Cost	Total Weight	Fitness
1	|ACB|ACB|ACA|AAB|FAD|ACB|ACA|AAB|FAD|FCA|	|ACB|	125.0	14200.0	590.0	0.746047619047619
2	|AEA|ACB|ACB|ACB|AEA|ACB|ACA|ACE|AAB|	|AEA|	148.0	16000.0	640.0	0.7619047619047619
3	|AEB|AEB|AEA|AEA|ACB|AEB|AEA|ACB|ACA|ACD|	|AEB|	148.0	16200.0	625.0	0.7627142857142856
4	|ADB|ADA|AEB|AEB|AEB|ADB|ADA|AEB|AEA|ACB|	|ADB|	138.0	15200.0	605.0	0.8022380952380952
5	|ADB|ADB|ADB|ADB|ADA|ADB|ADA|AEB|AEA|ACB|	|ADB|	138.0	15200.0	605.0	0.8022380952380952
6	|ADB|ADB|ADB|ADB|ADB|ADB|ADA|ADD|AEB|AEA|	|ADB|	138.0	15200.0	605.0	0.8022380952380952
7	|ADC|ADC|ADB|ADB|ADB|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
8	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
9	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
10	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
11	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
12	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
13	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
14	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
15	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
16	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
17	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
18	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
19	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905
20	|ADC|ADC|ADC|ADC|ADC|ADC|ADB|ADA|ADD|ADE|	|ADC|	138.0	15400.0	594.0	0.8024761904761905

Total evolution time: 50 ms

The best solution has a fitness value of 0.8024761904761905
Engine: Engine A
Electric Motor: Electric Motor D
Battery: Battery C

Figure 7. Evolutionary Process

The trade study found that the best solution can be determined using the logical design and specified user requirements. The best hybrid car based on the desired requirements (i.e., target power = 140 hp; maximum cost = US$ 30,000; and maximum weight = 700 kg) consists of Engine A, Electric Motor D, and Battery C with gained power = 138 hp; cost = US$ 15,400; and weight = 594 kg. The fitness value of the solution is about 0.802 out of 1.0.

Conclusion and Future Work

This paper demonstrated the use of GAs to perform an evolutionary trade-off in the design synthesis of the MBSE. The application presented here is to assist in the component selection process to assemble a hybrid car that consists of an engine, an electric motor, and a battery. A fitness evaluation was applied to the population to perform an Elitism selection method. The tentative conclusion was the best physical architecture level that included Engine A, Electric Motor D, and Battery C.

A single synthesis method cannot always solve every kind of problem (Cagan et al. 2005). Therefore, there are many possible future works to improve this study:

- Some selection methods can be evaluated to find the most suitable selector to assist the systems engineers to find a better solution (Chudasama, Shah & Panchal 2011) (Mashohor, Evans & Arslan 2005) (Moh & Geraghty 2011). Beside Elitism, the other selectors are roulette-wheel, Boltzmann, tournament, rank, and steady-state (Chakraborty 2010).
- In a bigger scale of industry, GAs alone may insufficient for a complex system with more parameters. GAs can be integrated with some algorithms, such as **Pareto-Optimal** (Nassar & Austin 2013) or **Niche-Pareto** (Horn, Nafpliotis & Go 1994).
- Blind-extensive experiments to a GA can waste time and resources. It can use a self-tuning to help the adaption with a different component library, structures, and attributes (Sugihara 1997).
- GAs work best when components are relatively independent of each other. Therefore, when there are strong constraints, such as Component A only works in some ranges of parameters of Component B, then a simple crossover can often result in infeasible solutions because component A and B are incompatible. However, there should be a pool of available components. Thus, the use of *Bill of Material (BOM)* and vendor list is necessary here. To solve this issue, there are some options that could be done as the future works:
    - Develop an efficient encoding and decoding mechanism to deal with relationship between each component.
    - Use **Ripple Down Rules** to manage the requirements incrementally using rule-based representation.
    - Include the completed Block Definition Diagrams and Internal Block Diagrams elements to ensure the compatibility between components (logical grouping) via the interfaces, connector, block, data type, and item flow.
    - Include *Product Line Engineering* perspective might solve the of component availability issue, for example applying *150% architecture model* to help the components selection and suggest the different product variances based on the model by including the overall common components, alternative components, and optional components. By doing so, the systems engineers can get the advantage of GA, which is to find a number of alternative sets of solutions. Moreover, a company might consider to create a new component (Walden et al. (eds.) 2015) with superior performance and/or custom functionality (Nassar & Austin 2013).
- The fitness function is a challenging issue. A fitness function should cover only a set of requirements, depending on the context of tested system. To accomodate the different dimension of importance from each stakeholder to a fitness function, the understanding of underlying reasons, opinions, and motivations of each stakeholder's requirements is really needed. The upcoming SysML 2.0 has been extended to cover the stakeholders as an element. This concern could be linked to SysML 2.0 as a future work.
- This paper focuses only on the physical layer. However, future works can also help to optimize the **logical architecture** (Lukasiewycz et al. 2011), e.g., optimizing the logical components in Package Diagrams, Block Definition Diagrams, and Internal Block Diagrams.
- Some of other applications are optimizing the control strategy to reduce the **fuel consumption and emissions** without reducing the overall performance of a system (Montazeri-Gh, Poursamad & Ghali 2006) (Huang, Wang & Xu 2006) (Montazeri-GH & Poursamad 2005) (Fang et al. 2011)

(Wan, Canedo & Abdullah 2015). Regarding to this, **an automated user scenario generation** with **Case-Based Reasoning** (Gozali 2002) (Daengdej & Lukose 2005) (Daengdej, Lukose & Murison 1999) (Daengdej et al. 1996) can be performed in many different driving cycles.

References

Ackva, S., 2013, 'Deployment Package Functional & Physical Architecture (FA & PA) Systems Engineering Basic Profile', International Council on Systems Engineering (INCOSE).

Albarello, N., Welcomme, J-B. & Reyterou, C., 2012, 'A formal design synthesis and optimization method for systems architectures', *9th International Conference on Modeling, Optimization & Simulation MOSIM'12*, Bordeaux, France.

Austin, M., 2012, *ENES 489P Hands-On Systems Engineering Projects: Foundations for Model-Based Systems Engineering*, Institute for Systems Research, University of Maryland, College Park.

Avancini, A., 2012, 'Security Testing of Web Applications: A Research Plan', *ICSE 2012*, IEEE, Zurich, Switzerland.

Branscomb, J.M., Paredis, C.JJ., Che, J. & Jennings, M.J., 2013, 'Supporting Multidisciplinary Vehicle Analysis Using a Vehicle Reference Architecture Model in SysML', *Conference on Systems Engineering Research (CSER'13)*, Elsevier B.V., Atlanta, GA.

Cagan, J., Campbell, M.I., Finger, S. & Tomiyama, T., 2005, 'A Framework for Computational Design Synthesis: Model and Applications', *Journal of Computing and Information Science in Engineering*, vol 5, no. 3, pp. 171-181.

Cameo Systems Modeler: User Guide 18.1 2015, No Magic, Inc., viewed 10 November 2017, from https://www.nomagic.com/files/manuals/Cameo%20Systems%20Modeler%20UserGuide.pdf.

Chakrabarti, A., 2002, *Engineering Design Synthesis*, 1st edn, Springer-Verlag, London.

Chakraborty, RC., 2010, *Genetic Algorithms & Modeling*, viewed 16 November 2017, from http://www.myreaders.info/html/soft_computing.html.

Chudasama, C., Shah, S.M. & Panchal, M., 2011, 'Comparison of Parents Selection Methods of Genetic Algorithm for TSP', *International Conference on Computer Communication and Networks*.

Daengdej, J. & Lukose, D., 2005, 'How case-based reasoning and cooperative query answering techniques support RICAD', *International Conference on Case-Based Reasoning*, Springer, Berlin, Heidelberg.

Daengdej, J., Lukose, D. & Murison, R., 1999, 'Using statistical models and case-based reasoning in claims prediction: experience from a real-world problem', *Knowledge-Based Systems*, Elsevier Science B.V.

Daengdej, J., Lukose, D., Tsui, E., Beinat, P. & Prophet, L., 1996, 'Dynamically Creating Indices for Two Million Cases: A Real World Problem', *EWCBR '96 Proceedings of the Third European Workshop on Advances in Case-Based Reasoning*, Springer-Verlag London, UK.

Dinger, R.H., 1998, 'Engineering design optimization with genetic algorithms', *Northcon/98 Conference Proceedings*, Seattle, WA, USA, USA.

Eiben, A.E. & Smith, J.E., 2003, *Introduction to Evolutionary Computing*, 1st edn, Springer-Verlag Berlin Heidelberg.

Evolution News, 2014, *Comparing Explanations for "Trade-offs" in Darwinian Theory and Intelligent Design*, viewed 11 September 2017, from https://evolutionnews.org/2014/01/comparing_expla/.

Fang, L., Qin, S., Xu, G. & L.,., 2011, 'Simultaneous Optimization for Hybrid Electric Vehicle Parameters Based on Multi-Objective Genetic Algorithms', *Energies*, vol 4, pp. 532-544.

Frey, S., Fittkau, F. & Hasselbr, W., 2013, 'Search-Based Genetic Optimization for Deployment and Reconfiguration of Software in the Cloud', *ICSE*, IEEE, San Francisco, CA, USA.

Friedenthal, S., Moore, A. & Steiner, R., 2015, *A Practical Guide to SysML: The System Modelling Language*, Elseiver Inc.

Frye, A., 2017, *Genetic Algorithms and Pareto-frontiers*, viewed 2 October 2017, from https://www.youtube.com/watch?v=k4AxbXSy76U&t=200s.
Getrost, T., 2006, *2-Point Crossover Operator and Stoch?*, viewed 16 November 2017, from https://sourceforge.net/p/jgap/mailman/message/1429908/.
Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
Gozali, F., 2002, 'Case Based Reasoning in Engineering Design', *JETTri*, vol 2, no. 1, pp. 13-28.
Graves , H. & Bijan, Y., 2011, 'Using formal methods with SysML in aerospace design and engineering', *Annals of Mathematics and Artificial Intelligence*.
Grosso, C.D., Antoniol, G., Merlo, E. & Galinier, P., 2007, 'Detecting buffer overflow via automatic test input data generation', Elsevier Ltd.
Hall, M., 2017, *JGAP Default Initialisation Configuration*, viewed 16 November 2017, from https://mathewjhall.wordpress.com/2013/02/18/jgap-default-initialisation/.
Harman, M. & Jones, B., 2001, 'Software Engineering using Metaheuristic Innovative Algorithms', *International Conference on Software Engineering (ICSE)*, Toronto, Ontario, Canada, Canada.
Hermawanto, D., 2013, 'Genetic Algorithm for Solving Simple Mathematical Equality Problem', Indonesian Institute of Sciences (LIPI).
Horn, J., Nafpliotis, N. & Go, D.E., 1994, 'A Niched Pareto Genetic Algorithm for Multiob jective Optimization', *IEEE World Congress on Computational Intelligence*, IEEE, Orlando, FL, USA.
Huang, B., Wang, Z. & Xu, Y., 2006, 'Multi-Objective Genetic Algorithm for Hybrid Electric Vehicle Parameter Optimization', *International Conference on Intelligent Robots and Systems*, IEEE, Beijing, China.
International Council on Systems Engineering (INCOSE), 2015, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Wiley.
Kerzhner, A.A. & Paredis, C.JJ., 2009, 'Using Domain Specific Languages to Capture Design Synthesis Knowledge for Model-Based Systems Engineering', *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, San Diego, California, USA.
Kerzhner, A.A. & Paredis, C.JJ., 2011, 'Combining SysML and Model Transformations to Support Systems Engineering Analysis', *Proceedings of the 4th International Workshop on Multi-Paradigm Modeling (MPM) 2010*.
Kerzhner, A.A. & Paredis, C.JJ., 2011, 'Model-Based System Verification: A Formal Framework for Relating Analyses, Requirements, and Tests', *Lecture Notes in Computer Science*, vol 6627, pp. 279-292.
Lazko, O., 2006, 'Genetic Algorithms Application for Components Parametric Synthesis Optimization', *Modern Problems of Radio Engineering, Telecommunications, and Computer Science*, IEEE, Lviv-Slavsko, Ukraine.
Leserf, P., Saqui-Sannes, PD., Hugues, J. & Chaaban, K., 2015, 'Architecture Optimization with SysML Modeling: A Case Study Using Variability', *Third International Conference on Model-Driven Engineering and*.
Lukasiewycz, M., Glaß, M., Reimann, F. & Teich,., 2011, 'Opt4J-A modular framework for meta-heuristic optimization', *13th Annual Genetic and Evolutionary Computation Conference (GECCO 2011)*, Dublin, Ireland.
Mashohor, S., Evans, J.R. & Arslan, T., 2005, 'Elitist selection schemes for genetic algorithm based printed circuit board inspection system', *IEEE Congress on Evolutionary Computatio*.
Meffert, K., 2017, *JGAP Documentation*, viewed 19 July 2017, from http://jgap.sourceforge.net/doc/jgap-doc-from-site-20071210.pdf.
Moh, N.R. & Geraghty, J., 2011, 'Genetic Algorithm Performance with Different Selection Strategies in Solving TSP', *World Congress on Engineering*, London, UK.

Montazeri-GH, M. & Poursamad, A., 2005, 'Optimization of Component Sizes in Parallel Hybrid Electric Vehicles via Genetic Algorithms', *ASME International Mechanical Engineering Congress and Exposition*, Orlando, Florida USA.

Montazeri-Gh, M., Poursamad, A. & Ghali, B., 2006, 'Application of genetic algorithm for optimization of control strategy in parallel hybrid electric vehicles', *Journal of the Franklin Institute*, vol 343, pp. 420-435.

Nassar, N.N., 2012, 'Systems Engineering Design and Tradeoff Analysis with RDF Graph Models', University of Maryland.

Nassar, N. & Austin, M., 2013, 'Model-Based Systems Engineering Design and Trade-Off Analysis with RDF Graphs', *Conference on Systems Engineering Research*, Elsevier B.V., Atlanta, GA.

No Magic Documentation 2017, viewed 19 July 2017, from https://docs.nomagic.com/.

Obitko, M., 1998, *Introduction to Genetic Algorithms*, viewed 16 November 2017, from http://www.obitko.com/tutorials/genetic-algorithms/.

Roedler, G.J. & Jones, C., 2005, 'Technical Measurement: A Collaborative Project of PSM, INCOSE, and Industry', INCOSE and PSM.

Sayanjali, M. & Nabdel, O., 2013, 'Remote Sensing Satellite Design using Model Based System Engineering', *Journal of Science and Engineering*, vol 1, pp. 43-54.

Spyropoulos, D. & Baras, J.S., 2013, 'Extending Design Capabilities of SysML with Trade-off Analysis: Electrical Microgrid Case Study', *Conference on System Engineering Research*, Elsevier B.V., Atlanta, GA.

Sugihara, K., 1997, 'Measures for Performance Evaluation of Genetic Algorithms', *3rd Joint Conference on Information Sciences*.

'Toyota Prius User-Guide: 2nd Edition for The 2010-2012 Models' 2012.

Trcka, N., Hendriks, M., Basten, T., Geilen, M. & Somers, L., 2011, 'Integrated Model-Driven Design-Space Exploration for Embedded Systems'.

Vanderperren, Y. & Dehaene, W., 2005, 'SysML and Systems Engineering Applied to UML-Based SoC Design', *2nd UML-SoC Workshop at 42nd DAC*, Anaheim (CA), USA.

Walden, D.D., Roedler, G.J., Forsberg, K.J., Hamelin, R.D., Shortell, T.M. (eds.), 2015, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Fourth Edition edn, Wiley.

Wan, J., Canedo, A. & Abdullah, M., 2015, 'Functional Model-Based Design Methodology for Automotive Cyber-Physical Systems', *IEEE Systems Journal*, vol PP, no. 99, pp. 1-12.

Wilhelmstotter, F., 2017, *JENETICS: Library User's Manual*, viewed 19 July 2017, from http://jenetics.io/manual/manual-3.8.0.pdf.

Winter, G., Periaux, J., Galan, M. & Cuesta, P., 1996, *Genetic Algorithms in Engineering and Computer Science*, John Wiley & Sons, Inc., New York, NY, USA.

Zou, D., Wang, R., Xiong, Y. & Zhan, L., 2015, 'A Genetic Algorithm for Detecting Significant Floating-Point Inaccuracies', *37th IEEE International Conference on Software Engineering*, IEEE.

Biography

Ho Kit Robert Ong started as a Borland C++Builder 6 product certified trainer and consultant for Borland Together and CaliberRM. He joined No Magic Inc. as a Senior Analyst and proceeded to work as a Director of Product Development. Having been in the position for years, he challenged himself to take on new responsibilities as a Director of Business Development. With his 16+ years of experience, Robert accumulates strengths in MBSE/MBRE, project management, requirements engineering, IT solutions finding, domain, and business process analysis, business process re-engineering, and Enterprise Architecture.

Habibi Husain Arifin started as a full-time programmer in 2010. Since 2012, he started to work as an independent solution architect for Document Management Systems (DMS), Data Dictionary, Business Process Management (BPM), and Business Intelligence. He received a bachelor degree in Computer Engineering and a master degree in Information Technology (Software Engineering). He has been awarded from many international innovation competitions during his student exchange program to Malaysia in 2009/2010. He is also active in some publications in AI and Software Engineering. His current research focuses on the areas of MBSE.

Nasis Chimplee brings more than 10 years of experience in software engineering, software architecture, and project management with a proven track record for designing, developing, managing, and implementing various projects. He is one of No Magic's top Java and UML/SysML/BPMN experts. He shows exceptional ability to effectively train developers, architects, and business managers on programming, modeling, and implementation of technology to large corporations.

Dr. Jirapun Daengdej was a former Dean of Vincent Mary School of Science and Technology, Assumption University of Thailand. After receiving his Ph.D. in Computer Science from University of New England, Australia in 1999. He received IBM Faculty Awards in 2009 and 2012. He was the founder of the Thailand Practical Software Engineering Conference. He is currently the president of a community called Thailand Software Process Improvement Network. He also serves as an Executive Technical Director of No Magic Asia Ltd.

Dr. Thotsapon Sortrakul is an Assistant Professor at Vincent Mary School of Science and Technology at Assumption University of Thailand. He received a research grant from the US Department of Defense (DoD) during his master and PhD studies in Electrical Engineering at Southern Illinois University. He has extensive experience working with over 40 government and private industries on projects related to ICT and engineering. He is the author of more than 40 publications. His current research focuses on the areas of Robotics, Data Analytics, Machine Intelligence, and Image Processing for the Royal Thai Army R&D Department.