A Combined Design Structure Matrix (DSM) and Discrete Differential Evolution (DDE) Approach for Scheduling and Organizing System Development Tasks Modelled using SysML

Article *in* Journal of Integrated Design and Process Science · August 2014

DOI: 10.3233/jid-2014-0013

4 authors, including:

François Christophe
Tampere University of Technology
29 PUBLICATIONS **98** CITATIONS
SEE PROFILE

Eric Coatanéa
Tampere University of Technology
76 PUBLICATIONS **201** CITATIONS
SEE PROFILE

Faisal Mokammel
Aalto University
14 PUBLICATIONS **19** CITATIONS
SEE PROFILE

Transactions of the SDPS:
Journal of Integrated Design and Process Science
2014
DOI 10.3233/jid-2014-0013
http://www.sdpsnet.org

A Combined Design Structure Matrix (DSM) and Discrete Differential Evolution (DDE) Approach for Scheduling and Organizing System Development Tasks Modelled using SysML

Sarayut Nonsiri, François Christophe, Eric Coatanéa* and Faisal Mokammel

Department of Engineering Design and Production, School of Engineering, Aalto University, Finland

Abstract During a system engineering process there are an important number of tasks that need to be organized, mapped together and recursively considered. The tasks that are mapped together are exchanging different flows of information and material. In this type of iterative processes, significant savings in term of development time can be made by providing a method that is optimizing the amount of feedbacks and iterations to the minimal level simply required for the successful development of the system. Task scheduling in a system engineering process can become extremely complex. Nevertheless it is a crucial step of the early stages of the systems engineering process for time-to-market, cost-efficiency and quality reasons. In this article, the authors are proposing to combine a computational approach (Discrete Differential Evolution) with Model Based Systems Engineering (MBSE) for minimizing iterations and reducing lead-time development. The present article is contributing to recent research works using Design Structure Matrixes (DSM) and computational methods for visualizing and analyzing systems engineering processes. The paper is proposing a framework integrating a model-based approach and a DSM based analysis of the process architecture to assist system engineers in organizing and scheduling tasks. As a result, this framework allows engineers to automatically populate DSMs generated from MBSE models developed in SysML. A specific stereotype is proposed to represent system development tasks in SysML. The sequencing of the engineering tasks is optimized with the application of a Discrete Differential Evolution algorithm (DDE) taking into account the different constraints. The practical use of the proposed framework is demonstrated on the case study of a mobile robot developed for the Eurobot competition. The article also discusses the possibility to use the current framework to analyze the impact of requirement changes on the scheduling of development tasks.

Keywords: Design structure matrix, discrete differential evolution, model-based systems engineering, sequencing, systems engineering process

1. Introduction

The complexity of engineering projects has increased in recent years due to different factors, such as the integration of technologies from various domains (e.g., mechanics, electronics, and embedded software) in new products and the necessity to collaborate with multi-disciplinary teams spread across the globe. Indeed, interactions and dependencies between team members are inevitable; they can result in increased iterations during the development of a product and increased difficulty to manage information

* Corresponding author. Email: eric.coatanea@aalto.fi

 Published by IOS Press

flows under constraints of cost, time, and resources (Blanchard & Fabrychy, 2005). The establishment of an appropriate project plan (e.g., task schedule) is of critical importance in a systems engineering process as it helps achieve constraints related to time-to-market, cost-efficiency and quality of the product.

In order to establish a task schedule, tasks must be clearly defined. (Ullman, 2009) proposed guidelines and formalisms for documenting the attributes of an engineering task. These attributes include a name, an objective, one or more predecessors and successors, a deliverable, a duration, etc. Currently the documentation linked to engineering tasks and their interactions is mostly stored in the form of hard copy or digital files and communicated within the team. However, keeping scheduling documents up-to-date with such an approach becomes cumbersome, and can lead to errors, when the number of tasks increases or tasks change following changes in requirements. As errors linked to task description, organization, and maintenance can drastically affect later stages of the development of a system, approaches should be developed to help engineers with this step of the systems engineering process.
Traditional methods such as PERT (Project Evaluation and Review Technique) (Kelley & Walker, 1959), CPM (Critical Path Method) (Malcolm, *et al.*, 1959), and Gantt chart (Gantt, 1919) are useful tools used in planning and scheduling engineering tasks. Nevertheless, they are not designed to handle the complexity of iterations in the systems engineering process. This paper introduces a four-step framework integrating a model-based approach with process architecture Design Structure Matrices (DSM) for handling this complexity and to support engineers during task identification and sequencing. DSMs are used for modeling, managing and visualizing tasks in system design (Eppinger & Browning, 2012; Steward, 1981a; Eppinger, *et al*., 1994). More precisely, the information flows between the tasks can be represented in a square matrix format, with the marks in the matrix both representing the information flow between activities and indicating the dependency between tasks.

The four steps of the proposed framework are presented in Fig. 1.(a) The first step, requirement modeling, produces SysML models of requirements based on customer needs and specific domain knowledge. In the second step, task identification, these models (i.e., stakeholder, functional, system, subsystem, and component) are used to identify tasks and the relationships among them. In the third step the process architecture DSM is built based on the identified tasks and relationships. The final step performs task sequencing based on the DSM built.

The remainder of this article is structured as follows. Section 2 presents previous works in model-based and DSM based approaches as well as task sequencing. The proposed framework is detailed in Section 3. Section 4 presents an application of the framework on the case of robot in the Eurobot competition. Section 5 presents a discussion on the developed framework and results obtained. Section 6 concludes on the work and future extensions.

2. Related Work

Systems engineering focuses on defining customer needs and required functionalities early in the development cycle, documenting requirements, proceeding with design synthesis, validating the conformance of the system developed in consideration of the complete problem (INCOSE, 2007). Modern engineering projects are growing in complexity as they involve multi-disciplinary teams with team members located in different parts of the world, diverse stakeholders, etc. This section presents a brief description of the methodology and related work.

2.1. MBSE methodology

Model Based Systems Engineering (MBSE) is a formal modeling approach for supporting the different activities throughout the systems engineering process, often represented with the so-called "V model" (Forsberg & Mooz, 1999). The main concept of MBSE approach is to represent a system and the related documents with formal models as opposed to traditional document based approaches where the information is usually represented in natural language on hard copy documents. Several MBSE methodologies have recently been developed in literature (Estefan, 2007); they enhance communications,

specifications, design, and reuse of the system design and specifications. Furthermore, model-based approach has been extended to support the designer's ability to evaluate a particular configuration (Shah, *et al.*, 2012) and to model the continuous dynamics of systems (Johnson, *et al.*, 2011).

System Modeling Language (SysML), a general-purpose visual modeling language, has been introduced to support MBSE. Indeed, SysML allows efficient communications among teams and stakeholders via different types of diagrams. Furthermore, XML Metadata Interchange (XMI) is used as a common format for exchanging information between computer software tools (Object Management Group, 2007). SysML is an extension of the Unified Modeling Language (UML) developed by the Object Management Group (OMG). It uses multiple diagrams as-is from UML, such as Use cases, Package, State Machine, and Sequences diagrams, and two modified diagrams, Activity and Block diagrams. Moreover, it integrates two new diagrams Requirement and Parametric diagrams (see Fig. 2).

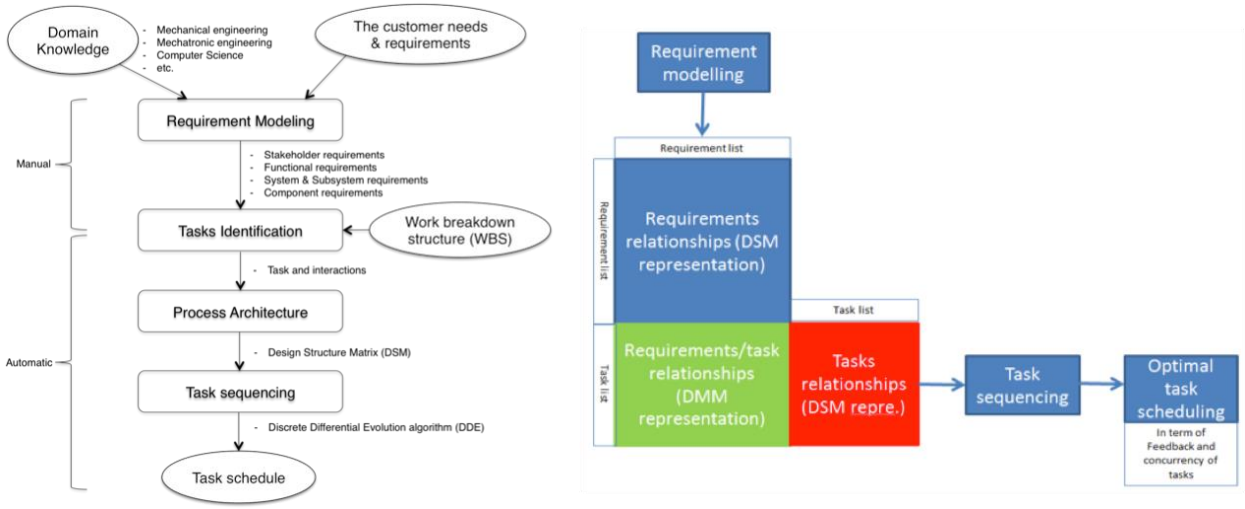

Figure 1. (a). Proposed integrated framework; (b). Proposed framework from a relationship perspective.

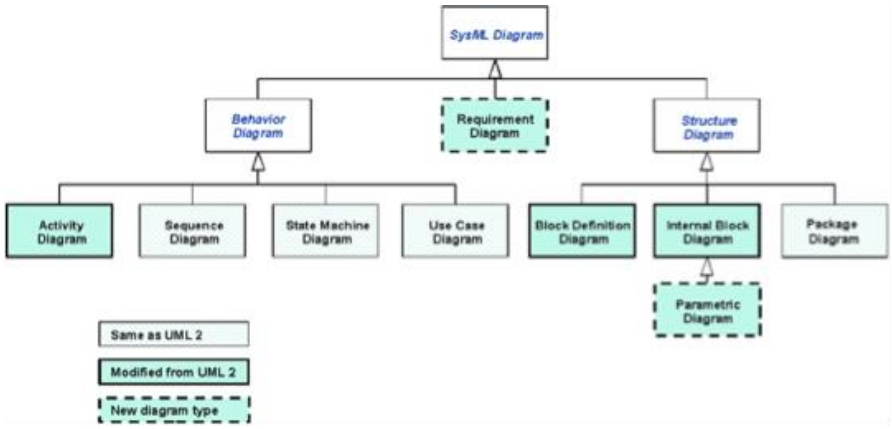

Figure 2. SysML Diagram types (Object Management Group, 2007).

SysML can be used to model the requirements in hierarchy models in order to visualize the relationships between them and for managing and communicating requirements and further system specifications among project teams and stakeholders. However, SysML is not adequate for creating,

organizing and managing engineering tasks. The use of process architecture DSM, presented in the next section, can help address these issues.

2.2. Process architecture DSM

DSMs are used as a tool for modeling, managing and visualizing tasks in the product development process (Eppinger & Browning, 2012; Steward, 1981a). In this work, process architecture DSMs are represented in Inputs in Rows/Feedback Above the Diagonal (IR/FD) convention where an input is represented in row (IR) and a feedback is represented above diagonal (FAD). For example, as shown in Fig. 3(b), the mark in the 2nd row and 1st column indicates that task A provides information to task B. In Fig. 3(c), the mark in the 1st row and 2nd column represents a feedback from task B to task A. Fig 3. (a), (b), and (c) also present the three basic types of flow characterizations: parallel, serial, and coupled, respectively. The flows represent interactions between tasks and are represented by marks or values in the corresponding DSM cell.

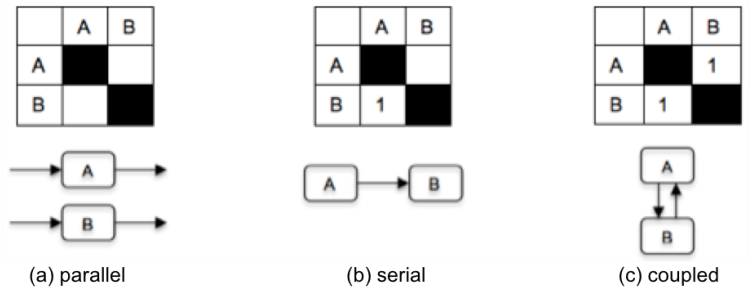

Figure 3. Three basic types of information flows in DSMs.

2.3. Task sequencing algorithms

Task sequencing algorithms can be used in our context in order to reduce feedback loops (minimizing the amount of required information flows between tasks). In the context of the system development process, algorithms mainly use a combination of two approaches to solve the scheduling problem: partitioning and sequencing.

Partitioning aims to separate a number of tasks into disjoint groups based on the types of flows. Once separated, disjoint groups of tasks can be performed concurrently. In recent years, several partitioning algorithms have been proposed (Warfield, 1973; Steward, 1981b; Tarjan, 1972). An example of an algorithm, also used in this work, is the Deep First Search algorithm proposed by Tarjan (Tarjan, 1972). This algorithm helps find strongly connected components (SCCs) in linear time $O(n+e)$, where n is the number of tasks (nodes) and e is the number of relationships (edges). An SCC is a subgraph in which there is a path from each node to other nodes as show in Fig. 4.

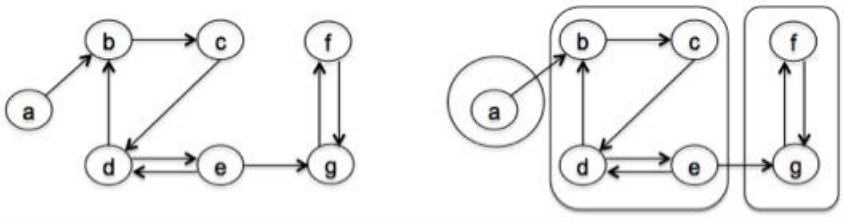

Figure 4. Strongly Connected Components graph.

The sequencing approach aims to reorder the tasks within coupled blocks in order to minimize the number of feedback relationships between tasks. The complexity of the sequencing problem can be represented as Quadratic Assignment Problem (QAP), which is a NP-hard problem (nondeterministic polynomial time problem). In recent years, several approaches have been proposed for task sequencing (Steward, 1981b; Kusiak & Wang, 1993; Gebala & Eppinger, 1991; Todd, 1997; Scott, 1998; McCulley & Bloebaum, 1996; Rogers, 1996; Altus, *et al.*, 1996; Whitfield, *et al.*, 2005; Meier, *et al.*, 2007; Nonsiri, *et al.*, 2012), including reducing iteration in binary matrices (Steward, 1981b), reducing iteration in weight matrices (Kusiak & Wang, 1993), reducing iteration length (Gebala & Eppinger, 1991), reducing iteration and crossover (McCulley & Bloebaum, 1996), and increasing concurrency (Todd, 1997).

In the literature, the Genetic Algorithm (GA) is mainly used for sequencing tasks. This is mainly due to the complexity level associated with sequencing problems. However, the use of GA requires the setting of several parameters, such as crossover probability, mutation probability, tournament size, and elitism number, in order to sequence the order of tasks. Therefore, the quality of solutions is dependent on the tuning of parameters. Ideally, an optimization algorithm should finding the true global optimum and the initial values of parameters should not affect this search. According to Storn and Price (Price, *et al.*, 2005), such an algorithm should be self-adaptive, easy to use with a minimum of parameters settings, and should provide fast convergence. They have recently proposed the Differential Evolution (DE) algorithm (Storn & Price, 1997), which can effectively perform and meet above basis.

DE is a population-based stochastic search method and the original idea was proposed for solving numerical optimization problems. In DE, the solutions within a population evolve over time using three operators, mutation, crossover, and selection. Recently, DE has been successfully applied to many numerical optimization problems and proven very competitive compared to GA and most other meta-heuristic algorithms (Holland, 1975; Kirkpatrick, *et al.*, 1983; Glover, 1989; Glover, 1990; Dorigo & Stützle, 2004; Bakhouya & Gaber, 2007). However, DE cannot be applied directly for solving combinatorial problems. The motivation of Discrete Differential Evolution (DDE) is to extend the functionalities of DE for solving permutative-based combinatorial problems (Godfrey & Donald, 2009). This paper investigates the utilization of an extended DE algorithm for solving the sequencing problem in the systems engineering process.

3. Model-Based Framework

Recently, several researchers have explored the advantage of integrating between SysML and DSM in order to executing system models that represent in graphical models for analyzing automatically in traditional DSM approaches (Waldman & Sangal, 2008; McLellan, *et al.*, 2009). Waldman and Sangal (Waldman & Sangal, 2008) have introduced the transformation of DSM from UML/SysML by exporting the graphical models into XMI standard, and then the relationships of the elements in DSM can be extracted from XMI. In McLellan *et al.*'s work (McLellan, *et al.*, 2009), they have demonstrated the generating DSMs and DMMs from SysML in order to show the predefine relationships between requirements and components, and avoid erroneous or missing data from manually input. The framework developed in this work of integation between SysML and DSM for organizing system engineering processes which consists of four steps, as illustrated in Fig. 1.(a) and detailed below. The following sections are going through those steps.

3.1. Requirements modelling

In this work, requirements are modeled based on their detail level, from abstract to specific requirements levels. They are decomposed into five categories: stakeholder, functional, system, subsystem, and component requirements, as presented in Fig. 5. This figure is based on (Hull, *et al.*, 2005) and it has been added a functional requirements level as to emphasise the need to define what the system should do in order to elaborate later the engineering tasks.

Stakeholder requirements are non-technical requirements describing the needs for a system to be developed. They are gathered based on stakeholder viewpoints and represent different domains of knowledge and levels of expertise. Traditionally, such requirements are mainly expressed in the form of natural language and collected as documents and digital files. System engineers need to transform them into more specific requirements in order to reduce ambiguity.

Functional requirements are derived from stakeholder requirements into specific functions describing what the system should do when its implementation is completed. They should be described in terms of what the system should do and not how it should do it. Generally, system engineers develop use cases to capture the functionalities that the system should provide to actors of its environment.

System requirements describe specific needs related to technical aspects required from the system. Such requirements are non-functional requirements as they represent different technical criteria (e.g., performance, robustness, security, reliability, maintainability, dimensions, etc.). These requirements give precisions to the functional requirements level and set constraints for engineering the system. As a system is usually broken down into sub-systems following a top-down approach, system requirements are derived at sub-system level and describe more precisely the requirements affected to a specific sub-system. This level of description of requirements is called Subsystem requirements. When further breaking down the system, the expertise of system engineers helps them define components and decide whether these components can be bought, subcontracted or built in-house. At this level, requirements are called Components requirements or configuration items specification. Such requirements represent specific expected performances and constraints for domain-specific components (e.g., mechanical, electrical, software). They precise in a formal manner the expected inputs and outputs of each component as well as the services it provides (description of interfaces) and how it provides them (description of behaviors). When component requirements are described, the following task identification procedure can be performed based on these sets of requirements. The following sub-section presents this procedure in more details.

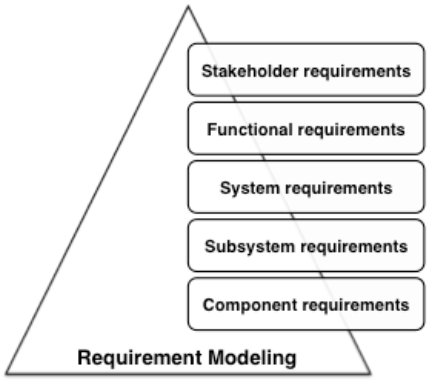

Figure 5. Requirements levels adapted from (Hull, *et al.*, 2005).

3.2. Task identification

During the task identification stage, requirements are associated to specific domains and tasks are identified to satisfy all requirements. Traditionally, work breakdown structure (WBS) (Project Management Institute, Inc., 2006) provides a collection of tasks that need to be done, and organizes them into work packages based on their domains, and then decomposes the tasks or activities in the project into smaller elements (e.g., hardware, software, data, and service). In this work, special emphasis is put on the alignment of engineering tasks with requirements at each level of description. This follows the WBS approach with higher-level requirements being the objectives of the project and component level

requirements being specifications of the deliverables. The application of this approach should ensure that each requirement is satisfied by the system at the end of the project.

As stated in Section 2.1, the diagrams and elements of representation provided by SysML do not allow the representation of engineering tasks. Therefore, this work proposes a new stereotype for the representation and modeling of tasks in SysML. Fig. 6 presents this new diagram element containing the necessary attributes for the description of an engineering task: a unique identifier (Id), a name (Name), the objective of the task (Objective), what the task should deliver (Deliverables), the tasks needed to be achieved before this task starts (Predecessors), the tasks waiting for this task to be achieved (Successors), starting date of the tasks (Start date), ending date (Finish date), the total duration of the task (Duration), the person responsible for this task (Team member), the person who prepared this task (Prepared by), the person who preliminarily checks the task (Checked by), the person who approves it (Approved by), and some additional note if needed (Note). The stereotype defined in Fig. 6 is presenting the list of attributes required for describing and defining a task.

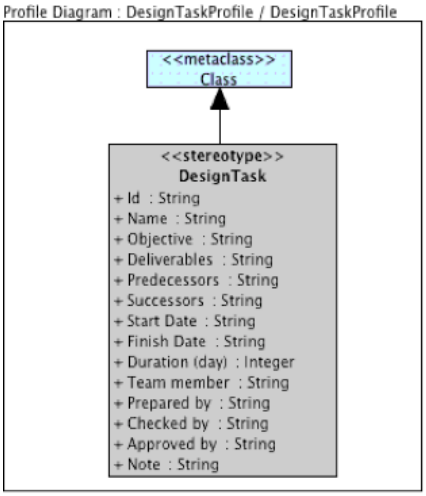

Figure 6. Proposed task stereotype.

Nevertheless, a development task is embedding a certain vision of the development process and its description is not straight forward as presented in literature about WBS (Project Management Institute, Inc., 2006). There is a need for describing how a development task can be derived. The way to describe a development task will have later important consequences on several aspects of the development organization and design performance. Those aspects should be explicitly considered at this stage of the article in order to consciously integrate the potential impacts that can be caused by the approach used to define a development task. Several viewpoints and perspectives are existing and decisions have to be taken that will have further consequences. More specifically, the decisions related to the definition of the development tasks will have consequences on the way the development teams will be organized and the way they will function and interact. It will have also consequences on the flow of information that will later be exchanged between the development teams. Finally, depending of the set of competences available in development teams, the breakdown structure of task will have major impact on the global performance of the development process. The purpose in this article is to provide a generic overview of different ways to define a development task and to briefly assess the potential consequences embedded in the different options. At the end of this section a decision is taken for the purpose of the present article based on the nature of the case study developed in the article. The different viewpoints in the following can be also considered as a development task:

- *Requirement category of the system*: A development task can be seen as a type of requirement category such as a functional, a system, a subsystem or a component requirement.

- *Phase of a development process*: A development task can be seen as a phase of the development process such as designing, coding or testing in the case of a software development.
- *Function of the functional architecture*: A development task can also be seen as the function of a system for example the moving module of a mobile robot can be seen as a development task of a mobile robot.
- *Subsystem of system architecture*: This perspective is to consider the physical architecture of a system and to take the subsystems as specific development tasks.
- *Domains of technology expertise*: It is to consider a develop task as a domain of technological expertise such as mechanical engineering, mechatronic, and computer science.

It is also possible to combine those viewpoints and to imagine combinations between the different viewpoints to form a composite definition of a development task with the final goal of satisfying requirements. For example it is possible to combine a requirement perspective with a development process phase perspective. A development task can be for example the conceptual design of the grabber of the robot. This brief summary is demonstrating that the definition of a system development task is not a trivial decision. In order to support a decision process regarding task definition in a project, it is possible to define few criteria supporting the selection of a certain viewpoint.

(1). Is it a new product development activity or an incremental product development activity?

(2). Is it a project related to the development of new product architecture or can it being expected reusing a relatively stable architecture already developed in previous projects?

(3). Is it a development activity requiring strong integration of different technological competences?

(4). Is it sensible to use the requirements structure as basis for organizing the development process?

To support the definition of a development task selected in this article; the nature of the case study should be analyzed by providing answers to those questions. The case study selected in the article is related to a rather incremental product development activity taking place every year and in which a relative stability of the product architecture is existing. The mobile robotic system used as a case study is requiring strongly integrated development competences to maximize the efficiency of information flow within a multi-disciplinary development team. Last, the approach promoted in the article is a Model Based Systems Engineering (MBSE) perspective where the structure of the initial requirements engineering model is playing a fundamental role on the further developments.

For all those reasons, it is best to consider a task as a composite concept in this article. A development task can be based on an organization integrating in a first level the phases of the development process, in a second level the subsystems existing in a stable system architecture will be used. The naming of the development tasks will then be based on the following structure: “name of the development phase”+ “subsystem name”. The integration of the technological competences is then constrained by the nature of the subsystems. The incremental nature of the development activity is also embedded in the fact that the tasks are considering subsystems names. Fig. 14 of the case study is following this organization of the tasks. For example in the task name Preliminary Grabber Design, Preliminary and design refers to a phase of the development process. Grabber relates to a subsystem of the mobile robot architecture.

To ensure later traceability, there is a need to define the links between tasks and requirements. This is not directly the goal of this section but it is required for checking the implementation of requirements. This can be achieved with a Multi Domain Matrix (MDM) representation. The MDM presentation is not used in the article since SysML is used as a modeling language. SysML is used to model the dependencies relationships between requirements and development tasks.

In the article; the relation between tasks and requirements can be defined using <<dependency>> that has an *N*-to-*N* relationship. In summary, requirements diagrams are not only used to specify, analyze, and communicate the requirements in a project, but also they are used as a principle to identify the tasks that must be developed in order to achieve the systems engineering project. The stereotype created in this research and presented in Fig. 6 is applied later in the course of the case study developed in this article. The next section is presenting the relations existing between tasks in term of precedence and successor criteria. A DSM is used to achieve this task.

3.3. Generation of a task-based Design Structure Matrix (DSM) for process modelling

Traditionally, experts with experience in planning can populate DSMs representing the relations between tasks. The framework developed in this article is more advanced and automated because the article follows the process presented in Fig. 1a and 1b. The entire process is based in this work on a requirement model implemented in a model-based systems engineering language (i.e. SysML) and automatically processed using the approach developed in this article into a task-based DSM. This is a subsequent progress and it offers two major advantages. It helps updating quickly tasks scheduling and task management when changes are made in a requirement models. It helps also tracing back and verifying the implementation of tasks and the fulfilment of requirements.

Another concrete advantage; when considering large scale projects containing hundreds of tasks, filling in a matrix of interactions between tasks is demanding and error prone. In this research, tasks and their attributes (e.g., objectives, goals and deliverables) are defined in accordance with the requirements and are aligned with the conceptual modelling architecture using SysML. Then, they are extracted from the SysML model in XMI format and are used to generate the task-based DSM representing the interactions between tasks. Once this matrix is automatically generated, optimization algorithms can be applied in order to reduce iterations in the systems engineering process.

3.4. Task sequencing using Discrete Differential Evolution (DDE)

In this section, the DDE algorithm (Godfrey & Donald, 2009) (Fig. 7) is applied in order to sequence the tasks during the systems engineering process. Conventionally, the notation DDE/x/y/z is used to represent different DDE strategies. In this notation, x represents a string denoting the vector to be perturbed, y is the number of different vectors considered for perturbation of x, and z is the type of crossover being used. Other notations of interest include exp meaning exponential and bin meaning binomial. In this work, the DDE/rand/1/exp strategy is applied, where rand represents a vector randomly selected to be perturbed, 1 is the number of random vectors selected for perturbation, and exp is the exponential crossover being used. The overall procedure of the algorithm is presented Fig. 7 and its implementation is detailed in Sections 3.4.1 to 3.4.5.

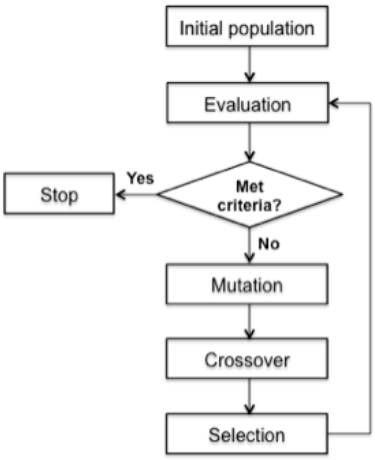

Figure 7. Flowchart of the DDE Algorithm process.

3.4.1. Initial population

The algorithm uses a population of individuals of size *NP*, initially populated with randomly generated individuals, and a DSM of dimension *D*. In this context, the size of each individual depends on the matrix size, *D*. Each individual vector is represented by a vector in the form, $\vec{X} = [X_1, X_2, X_3, ..., X_D]^T$, that also

represents a candidate solution. Each element in this vector is an integer that represents a task based on its position in the DSM as shown in Fig. 8.

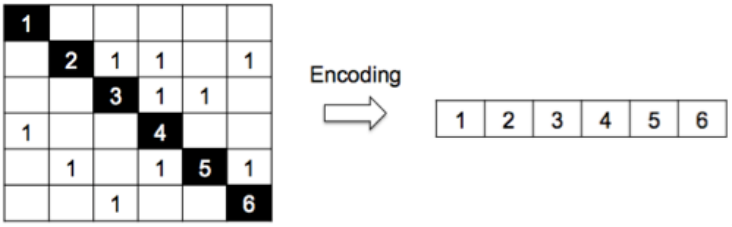

Figure 8. Individual vector representation.

The DDE is an iterative process, taking as an input the initial population and generating as an output the potential solutions. Equation 1 shows a solution generated at the *G*th generation *(G = 0, 1, …, Gmax)*, where *i = [1, …, NP]*.

$$\vec{X}_{i,G} = \left[x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}\right]^T \tag{1}$$

3.4.2. Evaluation

The fitness function is the most important optimization method to guide the searching process towards finding an optimal solution. Sequencing objective functions are used to evaluate the quality of generated solutions. Most objective functions consider either the number of iterations, iteration length, or increasing concurrency in order to evaluate potential solutions. In this work, the objective function presented in (Scott, 1998) is used as it combines the three aforementioned factors in one objective function as shown in Equation (2).

$$f = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \Omega(i,j) * w(i,j) \tag{2}$$

where,

$$\Omega(i,j) = \begin{cases} 1*[j+(n-i)]^2 & \text{if } i > j \\ 100*[j+(n-i)]^2 & \text{if } i < j \end{cases}$$

i is the row index, *j* is the column index, *w(i,j)* is a flow between activities *i* and *j* , *n* is the total number of activities in the DSM, and Ω(*i*,*j*) is a weighted distance. The objective is to minimize the number of iterations and iteration length, thus the weighted value of an element increases when it is above the diagonal and when its distance from the diagonal increases. All elements below the diagonal have less weighted values based on their positions in order to increase concurrency. As for traditional evolutionary computation algorithm, several criteria can be used to terminate the DEE process, including stall time limit, stall generations, elapsed time, and maximum number of generations.

3.4.3. Mutation

Mutation is a basic search mechanism of DDE for creating new vectors. Three indexes are randomly chosen from the range [*1,…,NP*], where $r_1^i \neq r_2^i \neq r_3^i \neq i$, in order to select target vectors, $\vec{X}_{r_1^i,G,}$ $\vec{X}_{r_2^i,G,}$ and $\vec{X}_{r_3^i,G,}$ respectively. As shown in Equation (3), a new vector $\vec{V}_{i,G}$, called *mutant vector*, is generated, by multiplying the differential variation of two vectors by a scaling factor *F* (a real and constant factor); typically its range is in the interval [0.4, 1].

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F*(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}) \quad (3)$$

However, this equation should be modified in order to ensure that all new mutant vectors are feasible solutions. Several approaches have been proposed such as Relative Position Index (RPI) (Qian, *et al.*, 2008), Forward/Backward Transformation (Onwubolu, 2006), The Permutation Matrix and Adjacent Matrix (Price, *et al.*, 2005), for solution mutation in combinatorial problems. In this paper, RPI was applied to perform the mutation operation as it provides feasible solutions producing new mutant vectors in all cases except when two or more elements have the same values. Moreover, all other approaches require an additional method to repair many infeasible solutions after generating new permutations. RPI is a transformation process of all the elements in a target vector, which is encoded in integer representation, into the floating-point interval [0,1]. Therefore, it can be operated in the continuous domain as traditional DE. The differential mutant is created using Equation (3), and then each element value is converted back into the integer domain related to its order number after sorting. For example, given three target vectors, $\vec{X}_{r_1^i,G} = [2,5,1,4,3]$, $\vec{X}_{r_2^i,G} = [3,4,1,5,2]$, and $\vec{X}_{r_3^i,G} = [1,3,5,4,2]$, all elements are transformed into an interval between 0 and 1, $\hat{X}_{r_1,G} = [0.25,1,0,0.75,0.5]$, $\hat{X}_{r_2,G} = [0.5,0.75,0,1,0.25]$, and $\hat{X}_{r_3,G} = [0,0.5,1,0.75,0.25]$. Then, Equation (3) is applied with F = 0.6 and $\hat{V}_{i,G} = [0.55,1.15.-0.6,0.9,0.5]$. As the result is based on the order after sorting, the new mutant vector is given as follows: $\hat{V}_{i,G} = [3,5,1,4,2]$.

3.4.4. Crossover

Crossover is a process for producing a new vector from randomly chosen parent individuals in the population. The crossover operator allows extending the potential diversity of the population. There are two main types of crossover methods, exponential and binomial crossovers. A trial vector, $\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, ..., u_{n,i,G}]$, is given through mating the elements from the target vector and the mutant vector based on a crossover rate (Cr) as a control parameter when Cr ϵ [0,1]. In order to extend this combinatorial problem, the exponential crossover is similar to two points crossover in GA since each element must appear exactly once. This combining process allows the creation of a new trial vector that belongs to a feasible solution; otherwise a new trial vector cannot be used in the next generation. The exponential crossover operation is illustrated in Fig. 9, where $\vec{X}_{i,G} = [2,4,1,5,3]$, $\vec{V}_{i,G} = [3,5,1,4,2]$, and *n = 1*.

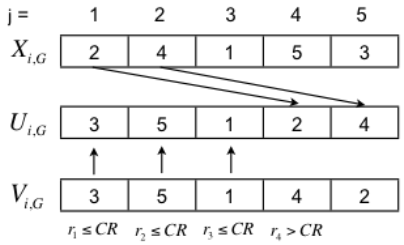

Figure 9. Exponential crossover.

3.4.5. Selection

Selection is a process for determining whether a mutant or target vector survives to the next generation *(G+1)* based on its fitness value. $f(\vec{X})$ is an objective function to be minimized. When comparing between a new trail and a target vector, if the fitness value of the new trail is equal or lower than that of the target vector, then the new trail vector should be kept for the next generation. This causes an evolution

in the population and better fitness values are achieved with each generation. The selection process is shown in equation (4).

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \le f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \text{if } f(\vec{U}_{i,G}) > f(\vec{X}_{i,G}) \end{cases} \quad (4)$$

It is worth noting that the DDE algorithm is an iterative method, which is an extension of the traditional DE algorithm that can be applied for the design process-sequencing problem. When applying DDE, each task can only exist exactly once within an individual vector, thus guaranteeing that all vectors are feasible solutions. The algorithm is composed of four basic steps: population initialization, mutation, crossover, and selection as presented above. The process runs until the conditions are met. DDE is simple, effective and easy to use since there are only four control parameters: number of generation (G), population size (NP), scaling factor (F), and crossover rate (Cr).

4. Case Study

In order to demonstrate the proposed methodology, this section covers the development of an autonomous robot taking into account the rules of the 2010 Eurobot competition. The development of such a robot requires multidisciplinary co-operation combining knowledge and skills from mechanical, mechatronics and software engineering. Moreover, the system architecture of the robot should be well documented to facilitate reuse in future editions as the rules change very little from year to year. The complexity of this type of project is usually handled by a team leader who develops a project plan by gathering requirements, schedules tasks, and communicates with the team in order to achieve the desired goal.

In this case study, four stakeholders are involved in the development of the project: a mechanical engineer, a mechatronic engineer, a computer engineer, and a system engineer. The system engineer takes on the role of team leader and manages the team, plans, and communicates with team members to ensure that the system meets the final goal in a timely manner. The mechanical engineer develops the physical parts of the robot. The development of electric system and control unit are left to the mechatronic engineer. The software engineer focuses on the development of algorithms and of the software system.

The rest of this section illustrates the practical use and the different steps of the proposed methodology. Requirements are first modeled using SysML. Tasks are then identified from the requirements models. These tasks are finally transformed into a DSM for further planning.

4.1. Organization using views and viewpoint approach

In this work, SysML, which can model systems and help communicate them to stakeholders, is used to model the requirements of a project entering the Eurobot competition. The open source software Topcased was selected to implement SysML. For modeling a system that is composed of multiple layers of subsystems and different domains of knowledge are required to fulfill stakeholder concerns. Due to the complexity of a system's architecture, it is not possible to use a single model to represent all of its aspects, such as the relationships between the requirements, functions, behavior, and structure of the system. A single comprehensive model unavoidably leads to problems that can be complicated and very difficult to understand for stakeholders due to the fact that they each have their own concerns that need to focus on only a specific part of the system and do not see the project in its globality.

The system architecture is represented in system models but, in order to efficiently communicate with stakeholders, a views and viewpoints approach is applied as a way of organizing the different SysML models based on which aspect of the system they focus on. This approach assists a system modeller in logically designing and analyzing system models and in efficiently communicating with stakeholders to ensure that their concerns are addressed. A package diagram represents a set of views organized as shown

in Fig. 10. It is composed of six views: requirements, functional, process, development, operational, and cross cutting/scenario view. The rest of this work will only focus on the requirements and process views, described in more detail later.

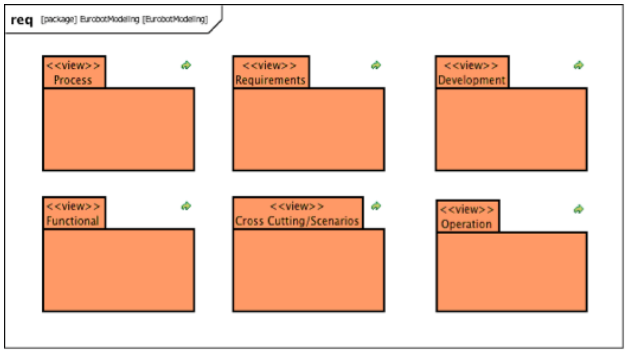

Figure 10. System models organization.

4.2. Modeling requirements in SysML

After the requirements are gathered, the next step in the systems engineering process is requirements analysis and management and it is crucial in the development of the system. Stakeholder needs are traditionally gathered using different approaches such as interviews, printed documents, brainstorming, questionnaires, prototypes, etc. In this work, the needs are extracted from printed documents written in natural language. After all the requirements are obtained, they are classified into different types of requirements based on their level (i.e., stakeholder, functional, system, subsystem, and component requirements levels).

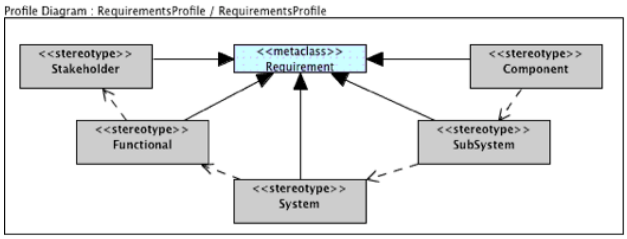

Figure 11. System models organization.

The requirements view in Fig. 10 represents the requirements model for the case study. This view supports the system engineer in understanding the overall requirements in the project being development. In order to model multi-level requirements in SysML, new stereotypes of requirements with the keyword <<stereotype>> are created by extending a requirements metaclass as shown in Fig. 11. These particular stereotypes are used to explicitly differentiate between requirements based on their level in order to model them in SysML efficiently. For example, functional requirements are represented by a stereotype with the keywords <<requirement, Functional>> at the top of the requirements diagram. Non-functional and constraints requirements (e.g., performance, modifiability, dimensions of the robot) are also very important for developing the system architecture. These requirements will be mainly contained within the system, sub-system, and component requirements levels.

Fig. 12 presents the hierarchy of the relationships between functional requirements level. For example, the requirement "the system of robot shall be capable of turning on/off" can be divided into two sub-functional requirements in order to refine the context. These two sub-functional requirements can be, for

example, “the system of the robot shall be capable of turning on” and “the system of the robot shall be capable of shutting down” (see Fig. 12). The requirement “the system of robot shall be capable of shutting down” can be further divided into two even more specific sub-functional requirements. In the model, sub-functional requirements are linked with a containment relationship (a circle with a cross sign), meaning that if these sub-requirements are satisfied, the master requirement is also satisfied.

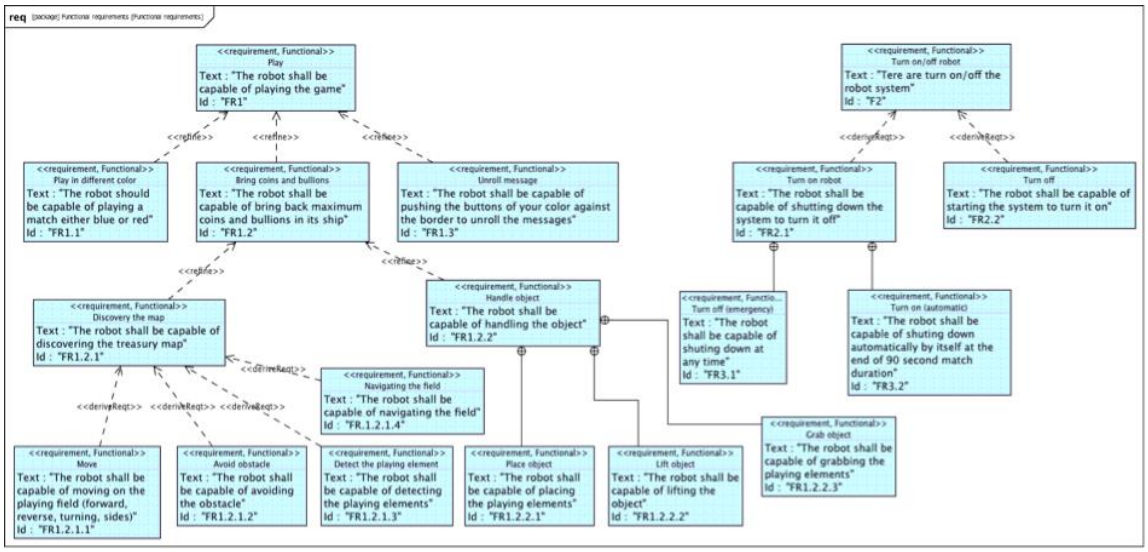

Figure 12. Functional requirements level.

4.3. Task identification

In order to properly identify system level tasks, the actions necessary to fulfill the given requirements should be considered. Therefore, SysML requirements diagrams are not only used for presenting the relationships between requirements and the system model, but they can also be used for representing the relationships between requirements and tasks.

The process view is one of the views depicted in Fig. 10; it contains the diagrams that represent the process aspect of the project. Therefore, the sequencing of the tasks in the systems engineering process is also included into this view. Fig. 13 presents the different packages necessary to complete the project, they are based on the different disciplines of the team members. Within each package, there are the requirements that belong to its domains that are used for communicating with specific team members. For example, the requirement “The robot shall be capable of grabbing the playing elements” belongs to the mechanical engineering domain package as it is up to the mechanical engineer to develop the grabber part satisfying this requirement. However, it is possible that the requirements can belong to more than one package domain. For example, the requirement “The robot shall be capable of detecting the playing elements” belongs to both the mechatronic engineering and computer science domains as the system developed to satisfy this requirement should both contain a sensor and software.

As presented in Fig. 6, a block represents a task and contains all the attributes that will be used for planning. The relationship between a task and requirement is defined using a <<dependency>> relationship. Fig. 14 represents the identification of the tasks that correspond to the “Handle object” requirement. In order to fulfill this requirement, the following five tasks should be completed: design primary gabber, design final grabber, produce grabber, test grabber, and integrate grabber into the robot body. Each task block is represented as a model element in the system model. The information flows is the transmission of information between the tasks. It is given a weight between 0 and 1, corresponding to the strength of the information flow. It is used to identify what inputs and outputs of a task are exchanged.

This information can be captured from documents, interviewing, meeting, and survey sheets. For weighting the information flow, it can be obtained by domain experts.

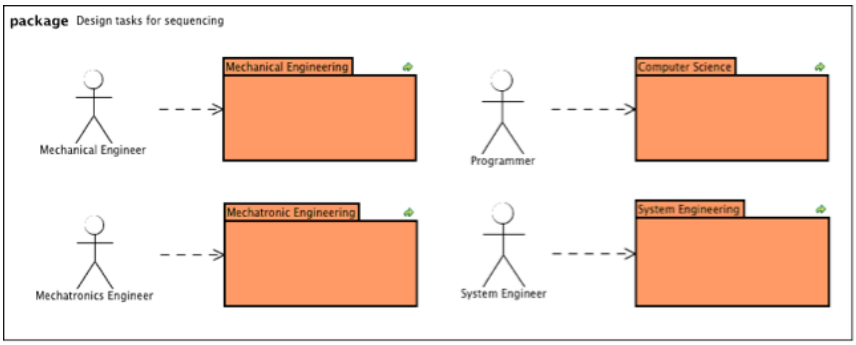

Figure 13. Tasks for the sequencing package.

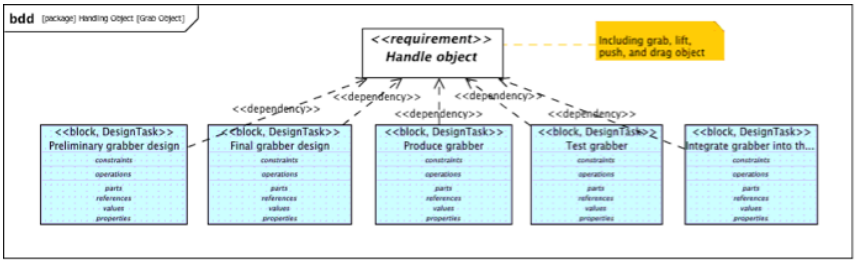

Figure 14. Relationships between tasks and the corresponding requirement.

Table 1. Process architecture DSM of the project

#	Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	Preliminary grabber design				0.5																								
2	Final grabber design	1																											
3	Produce grabber		1																										
4	Test grabber			1														1											
5	Integrate grabber into the system				1							1																	
6	Concept selection for the robot movement								0.5																				
7	Moving system integration						1					1						1											
8	Moving system testing							1																					
9	Robot's body design										0.5																		
10	Produce robot''s body									1																			
11	Robot's body testing										1																		
12	Start/stop system integration												1					1											
13	Start/stop the system design													0.5															
14	Start/stop system testing												1																
15	Concept selection of power source																0.5												
16	Power source integration															1													
17	Power source testing																1												
18	Develop software for detection system																				0.5								
19	Install sensors and camera																			1									
20	Detection system testing																			1									
21	Navigation system development								1			1											0.5							
22	Navigation system testing																					1							
23	Provide the presentation and report																												1
24	Coins and bullions collection testing					1			1					1							1						1		
25	Beacon stand integration											1																	
26	Operation mode testing																				1								
27	Discovery the map testing																						1						
28	Final testing																								1	1		1	

4.4. Transforming tasks into a DSM

The use of SysML allows the model to be represented in XMI format and easily exchanged between computer software tools. Therefore, a set of tasks in a SysML model can be transformed into a DSM in

order to analyze and visualize the process architecture. As shown in Fig. 15, it presents XMI code of identifying tasks in SysML. For example, Task T2 Predecessors= "T1(1)" Successors="T3(1)". The meaning is task T2 can be developed after task T1 has done. The strength of information flow from task T1 to task T2 is 1 (the value in parentheses). After task T2 is developed, Task T3 can be started by receive the information from task T2 with value 1. Table 1 lists the 28 tasks in an unordered sequence, which represents the tasks necessary to develop the autonomous robot. This can be done automatically with the software tool, thus avoiding any possibilities of human error.

```
<DesignTaskProfile:DesignTask xmi:id="_JC9dkDoeEeKz5abTbKctmA" Id="T2" Name="Final grabber design"
    base_Class="_DI3agDoaEeKz5abTbKctmA" Predecessors="T1(1)" Successors="T3(1)" Durationday="7" Checkedby="jimi"/>
<DesignTaskProfile:DesignTask xmi:id="_ZxxkQDoeEeKz5abTbKctmA" Id="T3" Name="Produce grabber"
    base_Class="_G4DCADoaEeKz5abTbKctmA" Deliverables="" Predecessors="T2(1)" Successors="T4(1)" Durationday="3" Checkedby="Jimi"/>
<DesignTaskProfile:DesignTask xmi:id="_fmbDcDoeEeKz5abTbKctmA" Id="T4" Name="Test grabber"
    base_Class="_NhIO0DoaEeKz5abTbKctmA" Predecessors="T3(1),T17(1)" Successors="T1(0.5),T5(1)" Durationday="2" Checkedby="Jimi"/>
<DesignTaskProfile:DesignTask xmi:id="_nbapwDoeEeKz5abTbKctmA" Id="T5" Name="Integrate grabber into the system"
    base_Class="_ZRGc8DoaEeKz5abTbKctmA" Objective="" Predecessors="T4(1),T11(1)" Successors="T24(1)" Durationday="5" Checkedby="Jimi"/>
```

Figure 15. XMI code of design tasks.

4.5. Task sequencing

All the tasks in the SysML diagrams are transformed into a process architecture DSM as shown in Table 1. The interactions of tasks are asymmetric. The relationships give a weight between 0 and 1, depending on the strength of the information flow between them. This can be seen in Table 1 above following the distinction made between values 0.5 given to most of the relationships between tasks and 1 given to a small amount of feedback relationships (upper side of the diagonal). The value 0.5 is set systematically in this work when a feedback associated with testing and comparison with an initial list of requirements is made. The reason for this lower weight is that the testing and verification is partially automatic in this type of framework due to the development of this model-based approach. In addition, a lower weight allows differentiating tasks precedence in the scheduling algorithm. At this stage, the tasks in the process model are performed in sequence in order to minimize the iterations and long feedbacks that exist above the diagonal in the matrix. Moreover, in order to accelerate the process, the heuristics equation as presented in equation (2) also allows the process to be organized in a way so that tasks can be performed simultaneously. As mentioned in Section 3.4, there are two steps for sequencing tasks. The first step is partitioning the tasks using a modified depth first search algorithm, thus separating the tasks into disjoint groups. The second step is sequencing the tasks using the Discrete Differential Evolution (DDE) algorithm.

Following the application of the two steps for task sequencing to the case of the Eurobot project, the process model was partitioned using SCC (Table 2). Seven couples of tasks were detected. For example, the fourth couple of tasks is composed of three tasks related to the development of the start/stop system. With this approach, there is an increase in performance as all task couples can be sequenced concurrently and independently from the rest of the process model. Thus, in the case of the case study project, at the end of the coupling procedure, fourteen elements remain in the process for sequencing in the next steps.

In this work, Discrete Differential Evolution (DDE) was applied to sequence the tasks in order to minimize the iterations in the process model. In DDE, there are four parameters including scaling factor (F), and crossover rate (Cr), the number of population, and the number of generation. As shown in Table 3, the scaling factor was set to 0.6 and the crossover rate was set to 0.9. The population and generations were both set to 100.

Table 2. The process model after partitioning using SCC

#	Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	Preliminary grabber design				0.5																								
2	Final grabber design	1																											
3	Produce grabber		1																										
4	Test grabber			1														1											
5	Integrate grabber into the system				1							1																	
6	Concept selection for the robot movement								0.5																				
7	Moving system integration						1					1						1											
8	Moving system testing							1																					
9	Robot's body design										0.5																		
10	Produce robot''s body									1																			
11	Robot's body testing										1																		
12	Start/stop system integration												1					1											
13	Start/stop the system design													0.5															
14	Start/stop system testing												1																
15	Concept selection of power source																0.5												
16	Power source integration															1													
17	Power source testing																1												
18	Develop software for detection system																		0.5										
19	Install sensors and camera																		1										
20	Detection system testing																			1									
21	Navigation system development								1			1										0.5							
22	Navigation system testing																					1							
23	Provide the presentation and report																												1
24	Coins and bullions collection testing					1			1						1						1						1		
25	Beacon stand integration											1																	
26	Operation mode testing																				1								
27	Discovery the map testing																						1						
28	Final testing																								1	1		1	

Table 3. The process model after partitioning using SCC

Parameters	Value
F	0.6
CR	0.9
Population	100
Generation	100

The result of sequencing using DDE is shown in Table 4. The tasks in the process architecture DSM are reordered, minimizing the number of iterations and long feedbacks, and increasing the concurrency of tasks.

To help with process planning, once the DSM was populated, it was transformed into a chart representing a project schedule that could be implemented by the interdisciplinary team (Fig. 16). The schedule highlights the different domains specific to each stakeholder, facilitating the reading.

Fig. 16 shows the project schedule and the tasks to be performed by the interdisciplinary team working on the case study of the Eurobot project. The testing feedbacks from Table 4 having weights of 0.5 in the upper part of the DSM in Table 1 are not presented in the Fig. 16. This is due to the fact that GANTT charts are used for representing the succession of tasks in a chronological order and consequently this type of representation is not adapted to represent feedback dependencies in time. For example, a dependency exists between task 17 and 15 represented by a feedback of value 0.5 in Table 4. This means that if during the testing phase of task 17, the solution appears to fail compared with the requirements list then we have to start again the design process by going back to the task 15.

The tasks are classified using different colors based on their domain in order to clarify for each team member the order of tasks that they need to perform and how they interact with other tasks in the project. The numbers indicated represent the task IDs. For example, tasks 15, 16, and 17 belong to the same work package for developing the power source; they were detected as a task couple and can, thus, be collapsed into a single task. As they do not require any information from predecessor tasks, the development of the robot body, power source, and detection system can be performed at the same time by the mechanical engineer, mechatronic engineer, and computer scientist, respectively. The system engineer, as the team leader, mostly focuses on managing and testing the system. The numbers of feedbacks in the process

architecture were reduced using the DDE algorithm in order to simplify the process, which makes it easier to estimate time, cost, and resources for allocating to each task.

Table 4. Process architecture DSM of the Eurobot project after sequencing

#	Tasks	9	10	11	15	16	17	18	19	20	1	2	3	4	6	7	8	25	26	13	12	14	5	21	22	27	24	28	23
9	Robot's body design		0.5																										
10	Produce robot''s body	1																											
11	Robot's body testing		1																										
15	Concept selection of power source					0.5																							
16	Power source integration				1																								
17	Power source testing					1																							
18	Develop software for detection system						0.5																						
19	Install sensors and camera						1																						
20	Detection system testing						1																						
1	Preliminary grabber design									0.5																			
2	Final grabber design									1																			
3	Produce grabber									1																			
4	Test grabber						1																						
6	Concept selection for the robot movement																0.5												
7	Moving system integration			1			1								1														
8	Moving system testing														1														
25	Beacon stand integration			1																									
26	Operation mode testing									1																			
13	Start/stop the system design																						0.5						
12	Start/stop system integration						1															1							
14	Start/stop system testing																				1								
5	Integrate grabber into the system			1										1										0.5					
21	Navigation system development			1														1							1				
22	Navigation system testing																									1			
27	Discovery the map testing																										1		
24	Coins and bullions collection testing									1								1	1			1	1					1	1
28	Final testing																		1										
23	Provide the presentation and report																												1

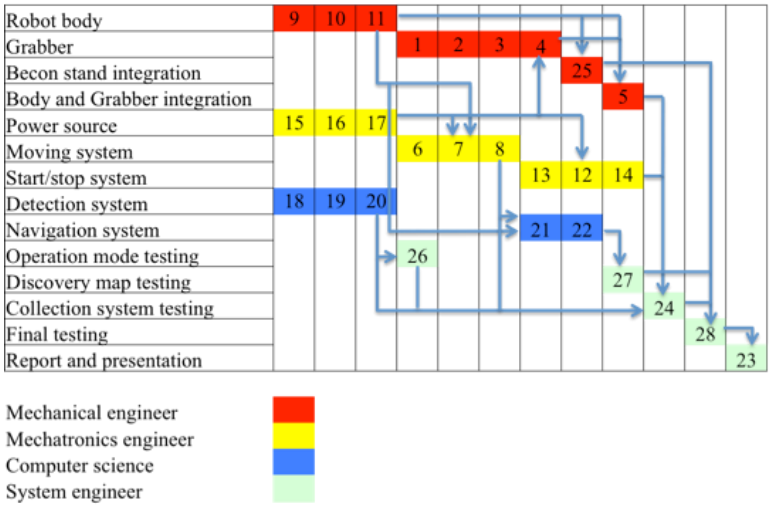

Figure 16. Task scheduling in the interdisciplinary project after optimization.

5. Discussion

This section presents the comparison of this approach, first with the initial sequence of tasks derived from XMI format, second with a sequence of tasks created manually by an expert of the domain. These results are shown in Table 5. This section analyzes the results from Table 5, providing comparison between results obtained by experts and through the application of the DDE algorithm. The “initial plan” represents the sequence of tasks directly taken from Table 1, immediately after the transformation obtained from the XMI format into a DSM (see Table 1). At this stage, the tasks are not in a valid sequence as the identification of the tasks was done in an unordered sequence. The numbers in Table 5

represent the comparison of the results between experts and the application of the DDE algorithms in terms of the number of feedbacks, feedbacks cost and concurrencies cost; they were derived from the application of Equation 2 presented in section 3.4.2. The fitness represents the convergence toward an optimal solution, the smaller value the better. The results show that both the experts and the DDE algorithm were equally able to minimize the number of feedbacks. These feedbacks are the information flows that needed to send back to upstream tasks for reworking which represented above diagonal. For example, there are 7 feedbacks remaining after sequencing the tasks as shown in Table 4. Feedbacks cost is a value of fitness function in Equation 2 that considering both the number of feedbacks and iteration length. If a feedback is far away from a diagonal line, iteration length is higher value. This means that, it takes longer times to rework in upstream tasks and consequence tasks again. Therefore, smaller feedbacks cost is better. Concurrencies cost represent the level of parallelization in the process model. Due to Equation 2, smaller value is better. For fitness cost as, smaller value is better. This result shows that in case of a small scale problem, as the one studied in this paper, computer application provides similar result than the results provided by experts. It can be assumed that in the case of a larger scale problem containing over 500 tasks, a DDE application would reduce significantly the amount of feedbacks and increase concurrency in a faster and more systematic manner than experts. There is here room for significant time saving in the development process using such type of approach. In addition, expert was prone to errors wheras the DDE application is applied systematically regardless of the complexity of the problem. In addition, in terms of concurrencies, the sequence of tasks provided by the DDE algorithm is better than that from the experts as the number of tasks depending on previous tasks is decreased (e.g. 18101 in this case) which shows that more tasks are independent and therefore can be conducted concurrently.

Table 5. Comparison of the results from experts and DDE

	Initial plan	Expert	DDE
Number of feedbacks	15	7	7
Feedbacks cost	1234550	315100	315100
Concurrencies cost	14966	19473	18101
Fitness cost	1249516	334573	333201

6. Conclusion and Future Work

This work introduced a framework for project planning and developed a case study showing the practical use of this framework. It was shown that using this framework, engineers are able to build requirements models and identify possible tasks required for developing a mechatronic product. The main contribution of this proposed framework includes: (i) The integration of a model-based approach and process architecture DSM allowing the automatic population of a DSM, thus avoiding the introduction of possible human errors, (ii) For the performance of DDE, the authors had compared the performance between DDE and Genetic algorithms (GA) in publication (Nonsiri, *et al.*, 2012). The result shown that DDE provides a very competitive result in term of the quality of the obtained solutions when compared with GA using examples taken from the literatures. In addition, it is a simple, effective and easy to use since there are less control parameters for setting. Therefore, the use of the DDE algorithm for sequencing the tasks in a process, providing same quality results as with genetic algorithms (GA) but necessitating the tuning of fewer control parameters, (iii) A framework that supports systems engineers in identifying tasks in a SysML model starting from the list of the project requirements and also considering the nature of the system development process. This part of the framework needs to be expended and tested also on other type of more challenging development projects such as New System Development. Otherwise the framework introduced is allowing them to explore the relationship between requirements and tasks using

a computer softw are tool. In future work, a change impact analysis technique will be developed to support systems engineers for studying how changes in requirements could impact the development process.

Acknowledgments

The author Sarayut Nonsiri was funded by the MOSES project financed by the Finnish research funding TEKES.

References

Altus, S. S., Kroo, I. M., & Gage, P. J. (1996). A genetic algorithm for scheduling and decomposition of multidisciplinary design problems, *ASME J. Mech. Des. 118*(4), 486–489.

Bakhouya, M. & Gaber, J. (2007). An immune inspired-based optimization algorithm: Application to the traveling salesman problem advanced modeling and optimization, *ICI Publishing 9*(1), 105-116.

Blanchard, B. S., and Fabrychy, W. J. (2005). *Systems Engineering and Analysis*, 4th ed., Prentice Hall, New Jersey.

Dorigo, M. & Stützle, T. (2004). *Ant Colony Optimization*, MIT Press, Cambridge, MA.

Eppinger, S. D., & Browning, T. R., (2012). *Design Structure Matrix Methods and Applications (Engineering Systems)*., MIT Press, Cambridge.

Eppinger, S.D., Whitney, D. E., Smith, R. P. & Gebala, D. A., (1994). A model-based method for organizing tasks in product development, *Research in Engineering Design 6*(1), 1–13.

Estefan, J. (2007). *Survey of Model-Based Systems Engineering (MBSE) Methodologies*, Jet Propulsion Laboratory, California Institute of Technology, Pasedena, CA.

Forsberg, K. & Mooz, H. (1999). Systems engineering for faster, cheaper, better. *In Proceedings of the ninth annual international symposium of the INCOSE*, Brighton, England.

Gantt, H. L. (1919). *Organizing for Work*, Newyork: Harcourt, Brace and Howe.

Gebala, D. A., & Eppinger, S. D. (1991). Methods for analyzing design procedures, In *Proceedings of the ASME Third International. Conference On Design Theory and Methodology*, Miami, FL, September 1991.

Glover, F. (1989). Tabu search: part 1, *ORSA J. Comput. 1*(3), 190–206.

Glover, F. (1990). Tabu search: part 2, *ORSA J. Comput. 2*(1), 4–32.

Godfrey, C. O., & Donald D. (2009). *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*, Springer, Heidelberg.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI.

Hull, E., Jackson, K., & Dick, Jeremy. (2005). *Requirement Engineering (2nd ed.)*, Springer, London.

INCOSE, (2007). *Systems Engineering Handbook – A guide for System Life Cycle Processes and Activities*.

Johnson, T, Kerzhner, A., Paredis C. J., & Burkhart R. (2011). Integrating models and simulations of continuous dynamics into SysML. *J. Comput. Inf. Sci. Eng. 12*(1), 011002-011002-11. doi:10.1115/1.4005452.

Kelley, J. & Walker, M. (1959). Critical-path planning and scheduling, in *Proc. Eastern Joint IRE-AIEE-ACM computer conference* (pp. 160-173). ACM, December 1-3, 1959.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing, *Science 220*(4598), 671–680.

Kusiak, A., & Wang, J. (1993). Decomposition of the design process, *ASME J. Mech. Des. 115*(4), 687–695.

Object Management Group, (2007). *OMG Systems Modeling Language Specification*

Onwubolu, G. C. (2006). Scheduling flow shops using differential evolution algorithm, *European Journal of Operational Research 171*(2), 674-692.

Malcolm, D. J., Roseboom, C. Clark, & Fazar, W. (1959). Application of a technique for research and development program evaluation, *Oper. Res. 7*(5), 646–669.

McCulley, C., & Bloebaum, C. L. (1996). A genetic tool for optimal design sequencing in complex engineering systems, *Struct. Optim. 12*(2), 186–201.

McLellan, J.M., Maier, J.R.A., Fadel, G.M., Mocko, G.M. (2009). *Generating Design Structure Matrices And Domain Mapping Matrices Using SysML*, 11th International Design Structure Matrix Conference (DSM'09), Greenville, SC, Oct 12-13.

Meier, C., Yassine, A. A., & Browning, T. R. (2007). Design process sequencing with competent genetic algorithms, *Journal of Mechanical Design, Transactions of the ASME 129*(6), 566-585.

Nonsiri, S., Coatanea, E., & Bakhouya, M. (2012). A discrete differential evolution algorithm for product development scheduling, In *Proceedings of ASME International Design Engineering Technical Conference and Computers and Information in Engineering Conference* (pp. 1229-1236), Chicago, Illinois, USA, August 12–15, 2012. doi:10.1115/DETC2012-70390.

Price, K., Storn, R., & Lampinen, J., (2005). *Differential Evolution - A Practical Approach to Global Optimization*, Springer, Heidelberg.

Project Management Institute, Inc. (PMI). (2006). *Practice Standard for Work Breakdown Structures, Second Edition, Project Management Institute*, USA, ISBN 10: 1-933890-13-4

Qian, B., Wang, L., Hu, R., Wang, w.-L., Huang, D.-X., & Wang X. (2008). A hybrid differential evolution method for permutation flow-shop scheduling, *The International Journal of Advanced Manufacturing Technology* 38, 757-777.

Rogers, J. L. (1996). *DeMAID/GA: An Enhanced Design Manager's Aid for Intelligent Decomposition*, NASA TM-11024.

Scott, J. A. (1998). *A strategy for Modeling the Design-Development Phase of a Product*, in Department of Marine Engineering, University of Newcastle upon Tyne, Newcastle upon Tyne.

Shah, A. A., Paredis, C. J. , Burkhart, R, & Schaefer, D. (2012). Combining Mathematical Programming and SysML for Automated Component Sizing of Hydraulic Systems. *J. Comput. Inf. Sci. Eng. 12*(4), 041006-041006-14. doi:10.1115/1.4007764.

Steward, D. V. (1981a). The design structure system: a method for managing the design of complex system, *IEEE Transactions on Engineering Management* 28, 71-74 .

Steward, D. V. (1981b). *Systems Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, New York.

Storn, R. & Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimization 11*(4), 341–359.

Tarjan, R. (1972). Depth-First Search and Linear Graph Algorithms, *SIAM J. Comput. 1*(2), 146–160.

Todd, D. (1997). *Multiple Criteria Genetic Algorithms in Engineering Design and Operation*, Ph.D. thesis, Engineering Design Centre, University of Newcastle upon Tyne, UK.

Ullman, D. (2009). *The Mechanical Design Process* (4th ed.), McGraw-Hill, New York.

Waldman., F., Sangal., N. (2008). *Uniting systems modeling approaches with DSMs*, 10th International Design Structure Matrix Conference (DSM'08), Stockholm, Sweden, Nov 11-12.

Warfield, J. N. (1973). Binary Matrices in System Modeling, *IEEE Trans. Syst. Man Cybern. 3*(5), 441–449.

Whitfield, R. I., Duffy, A. H. B., Gartzia-Etxabe, L. K., & Kortabarria, (2005). *Identifying and evaluating parallel design activities using the design structure matrix*, International Conference on Engineering Design 2005, Melbourne, August 15–18.

Author Biographies

Sarayut Nonsiri is a doctoral student at Aalto University School of Engineering, Department of Engineering Design and Production. His research work focuses on investigating in both theory and application for modeling and simulation during the early stages of the design process, model-based systems engineering, computation algorithm, and artificial intelligence areas.

François Christophe is a post-doctoral researcher at Aalto University School of Engineering, Department of Engineering Design and Production. His research is focused on Knowledge Representation and semi-formal modeling. His research work consists in the transformation of weakly defined design problems expressed in Natural Language into more formally expressed specification models with the application of tools and methods from Artificial Intelligence.

Eric Coatanéa has received is double doctoral degree in Engineering design in October 2005 from Helsinki University of Technology in Finland (nowadays Aalto University) and from University of West Brittany (France). Before embarking in doctoral studies, Eric Coatanéa has worked during 11 years has a manufacturing engineering teacher in University of West Brittany. He studied in University of West Brittany, INSA Toulouse and in Ecole Normale Supérieure Cachan. From 2005 to 2007, he has been a Marie Curie fellow. From 2008 to 2013, he has been fixed term professor of product development in Aalto University where he has established a new research group with a focus on modeling, simulation and decision making at early development stages with a system perspective.

Faisal Mokammel is a doctoral student at Aalto University School of Engineering, Department of Engineering Design and Production. He received M.Sc. degree in electrical engineering from Dalarna University, Sweden. Since 2011, he is member of product development research group, led by Dr. Eric Coatanéa. Mokammel is working on early stages of the design process mainly requirement engineering using natural language processing, graph theory and computational algorithm.