

Marc Aiguier, Frédéric Boulanger, Daniel Krob and Clotilde Marchal, editors

Proceedings

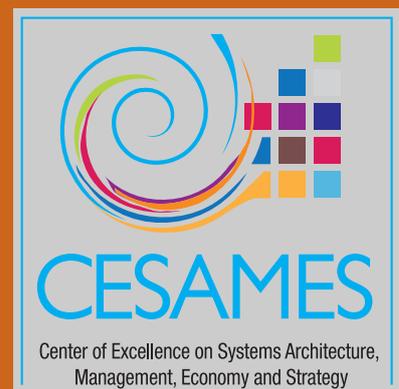
**Poster Workshop of the
Complex Systems Design & Management Conference
CSD&M 2013**

Paris, France, December 4th 2013

A workshop at

CSD&M
2013

Organized by



Copyright © 2013 for the individual papers by the papers' authors.
Copying permitted for private and academic purposes.
This volume is published and copyrighted by its editors.

Editor's addresses:

Marc Aiguier

École Centrale Paris - Laboratoire MAS
Grande Voie des Vignes
92195 Châtenay-Malabry, France
marc.aiguier@ecp.fr

Frédéric Boulanger

Supélec - Département Informatique
3 rue Joliot-Curie
91192 Gif-sur-Yvette cedex, France
frederic.boulanger@supelec.fr

Daniel Krob

École Polytechnique
91120 Palaiseau cedex, France
dk@lix.polytechnique.fr

Clotilde Marchal

EADS
12 rue Pasteur, BP 76
92152 Suresnes cedex, France
clotilde.marchal@eads.net

Contents

Preface v

Vikas Shukla and Guillaume Auriol
*Methodology for Determining Stakeholders' Criteria Weights
in Systems Engineering* 1

Vikas Shukla and Guillaume Auriol
Reinventing Goal-Based Requirements Modeling 13

André Ayoun, Gianni Inzerillo, Benoit Fonck and Alfredo Gomez
*Concurrent system engineering in Air Traffic Management:
steering the SESAR program* 25

Mikhail Belov
Agile stylized approach to manage complex project 33

Thierry Sop Njindam, Torsten Metzler, Udo Lindemann and Kristin Paetzold
*A Model-Based Assessment for the Solution Space
of a Cognitive Coffee Robot Waiter* 45

Armin Hasse and Cees Michielsen
The Missing Link - between Requirements and Design 59

Romaric Guillerm, Hamid Demmou and Nabil Sadou
Global Safety Management Method in Complex System Engineering 75

Vincent Chapurlat and Nicolas Daclin
*Proposition of a guide for investigating, modeling
and analyzing system operating modes: OMAG* 87

Fabio Roda
Systems engineering and modeling: some epistemological remarks 99

Fabien Retho, Hichem Smaoui, Jean-Claude Vannier and Philippe Dessante
*A model-based method to support complex system design
via systems interactions analysis* 115

Olivier Rosec and Pascal Rivière	
<i>Model-Based Interchange Formats: a Generic Set of Tools for Validating Structured Data against a Knowledge Base</i>	127
Imad Sanduka, Tim Lochow and Roman Obermaisser	
<i>Runtime Fault Prediction and Prevention for Emerging Services in System of Systems</i>	139
Arne Herberg and Udo Lindemann	
<i>A different view on system decomposition – subsystem-centered property evaluation in multiple supersystems</i>	153
Fernand Quartier and Frederic Manon	
<i>Simulation for all components, phases and life-cycles of complex space systems</i>	167
Yann Hourdel	
<i>Towards a formal language for systemic requirements</i>	175

Preface

This volume contains the proceedings of the poster workshop at CSD&M 2013.

This workshop was organized to foster discussions about topics presented in papers that were not advanced enough to be published at the main CSD&M conference, but were worth a shorter presentation around a poster.

The program committee of CSD&M selected these papers for presentation at the Poster Workshop and publication in separate proceedings.

About CSD&M

The purpose of the “Complex Systems Design & Management” (CSD&M) conference is to be a forum for both academic researchers and industrial actors working on complex industrial systems architecture and engineering in order to facilitate their *meeting*. The last three CSD&M 2010, CSD&M 2011 and CSD&M 2012 conferences – which were held in October 2010, December 2011 and December 2012 in Paris – were the first steps in this direction with respectively more than 200, 250 and 280 participants coming from 20 different countries with an almost perfect balance between academia and industry.

The CSD&M academic–industrial integrated dimension

To make the CSD&M conference this convergence point of the academic and industrial communities in complex industrial systems, we based our organization on a principle of *complete parity* between academics and industrialists. This principle was first implemented as follows:

- the Program Committee is 50% academics and 50% industrialists,
- Invited Speakers are coming in a balanced way from numerous professional environments.

The set of activities of the conference followed the same principle. They indeed consist of a mixture of research seminars and experience sharing, academic articles and industrial presentations, software and training offers presentations, etc. The conference topics cover in the same way the most recent trends in the emerging field of complex systems sciences and practices from an industrial and academic perspective, including the main industrial domains (transport, defense & security, electronics & robotics, energy & environment, health & welfare services, media & communications, e-services), scientific and technical topics (systems fundamentals, systems architecture & engineering, systems metrics & quality, systemic tools) and system types (transportation systems, embedded systems, software & information systems, systems of systems, artificial ecosystems).

The CSD&M 2013 edition

The CSD&M 2013 edition received 76 submitted papers, out of which the program committee selected 22 regular papers to be published in these proceedings, which corresponds to a 29% acceptance ratio. The program committee also selected papers for a collective presentation and discussion at the poster workshop of the conference, 15 of which are included in these proceedings.

Each submission was assigned to at least two program committee members, who carefully reviewed the papers, in many cases with the help of external referees. These reviews were discussed by the program committee during a physical meeting held in C.E.S.A.M.E.S. office in Paris by June 11, 2013 and via the EasyChair conference management system.

We also chose 17 outstanding speakers with various industrial and scientific expertise to give a series of invited talks covering all the spectrum of the conference, mainly during the two first days of CSD&M 2013, the last day being dedicated to a special “vision session” and the presentations of all accepted papers in parallel with three system-focused tutorials. The first day of the conference is especially organized around a common topic – Systems of systems – that gives a coherence to all the initial invited talks.

October 30, 2013
Gif-sur-Yvette

Marc Aiguier
Frédéric Boulanger
Daniel Krob
Clotilde Marchal

Methodology for Determining Stakeholders' Criteria Weights in Systems Engineering

Vikas Shukla and Guillaume Auriol

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ. de Toulouse, INSA, LAAS, F-31400, Toulouse, France
vshukla@laas.fr gauriol@laas.fr

Abstract. Multi criteria decision making involves evaluation of various alternative solutions upon a set of criteria. The result of multi criteria decision making is the best alternative which secures the highest score with the predefined criteria. Usually, these criteria are weighted in an order to represent their stake in the final selection. In multi criteria decision making this step is very critical for the selection of the right product. Often these criteria are traced back to a set of multi-disciplinary stakeholders participating in the evaluation process. Often these stakeholders differ upon their weighting of a particular criteria with other stakeholders. In this paper we provide a holistic criteria weighting method which allows to assimilate the different criteria weights from different stakeholders to provide a single set of criteria weights, uniformly acceptable by all stakeholders. This paper shows how the criteria weights can be simulated and used priorly to find the design solutions. The simulations results can be used to accept or refute the particular design alternative.

1 Introduction

Decision making is pervasive phenomenon in *systems engineering* (SE) projects. Some decisions involve single decision maker (DM). The responsibility of such a decision depends upon the sole single DM. Often, SE decisions involve more than one DM, thus adding fabric of complexity to the problem. Decision making becomes more complex, when multiple criteria get involved in the problem. Such problems, where multiple criteria and multiple decision makers are involved in a problem are called multi criteria decision making or analysis (MCDM/MCDA) problems.

A systems engineering project involves multiple stakeholders and multiple criteria decision analysis [1]. Such problems need attention and a formal methodology, to provide pertinent solutions. Variety of methodologies exist in the literature which provide more or less acceptable solutions, such as the famous AHP technique [2], Multi-attribute utility theory [3], ELECTRE Methods [4–6], PROMETHEE [7], TOPSIS[8], etc. In this paper, we are concerned with only the first step of the decision making problem, i.e., *criteria weighting*. In this paper we present a methodology for *criteria weighting* in the context of a systems engineering projects. Our criteria weighting approach can provide weights for a variety of decision making techniques. Our approach is based on the classical preference modelling technique given by Fishburn *et al.* [9].

A systems engineering project typically involves a crowd of multi-disciplinary stakeholders. Success of a project depends upon the decisions and choices made during the analysis of architectures & alternatives, and during the selection of components during the detail design phase [10]. Systems engineering is about making the right decisions to achieve the development of a product which is exactly demanded by their clients and stakeholders. These right decisions do not come automatically from a thin air, rather precise metrics are needed to evaluate the appropriate alternatives and right design components. As these decisions are based on these multiple criteria, they need to be weighted to measure their impact on the final decision choice. This task of providing weights to the criteria may seem trivial for a single decision maker, but when multiple stakeholders are involved this task becomes fairly difficult. As the different stakeholders differ upon the weights for various criteria, while each stakeholder being correct in his own view.

It is interesting to see that industries seldom use techniques which demand high cognitive loads on DMs. Even if a technique is more correct technically but leads to high cognitive load, it will hardly find usage in industry. Industries prefer simple techniques to pacify majority of its non-technical stakeholders. Our technique claims to pose less cognitive load to the stakeholders as compared to the other technique.

In this paper, we provide a technique which allows to assimilate the various criteria weights from the different stakeholders to provide a single array of criteria weights which can be uniformly accepted by all the different decision makers or stakeholders.

The major contributions of this paper are as follows:

- It provide a holistic way to integrate the different criteria weights of different stakeholders to provide a single weights using the classical preference modelling.
- It show that how all stakeholders are uniformly satisfied with the proposed technique.

This paper is organized as follows: Section 2 presents the criteria weighting problem. Section 3 presents the state of art of criteria weighting problems. Section 4 presents our novel approach. Section 5 presents an example of our approach. Section 6 discusses about the advantages and disadvantages of our approach, and the industrial readiness assessment. Section 7 concludes and presents future perspectives.

2 Criteria Weighting Problem in Systems Engineering

In a systems engineering project, it is of great importance that most of stakeholders are satisfied with the various decisions taken during the product development and with the final resulting end product. A higher satisfaction among the stakeholders can be guaranteed if the various stakeholders criteria weights are taken into account in a transparent and holistic manner.

The *criteria weighting* is critical part of the MCDM problem during analysis of alternatives in a system engineering project. Different stakeholders own different views towards the different criteria, their perception of weights differ from each other which makes it very tedious and difficult to come up with an agreement on a particular set of criteria weights. This often causes the conflicts among the stakeholders, which may halt

the progress of the project, if it remains unresolved. The *criteria weighting* problem refers to the problem of generating a single array of criteria weights from multiple arrays of criteria weights emerging from different stakeholders. Often, such problems start with demanding the DMs' weights for all the involved criteria and then provide a mechanism to find the mean weight. This approach assumes that the DMs' know their particular weights, which is often not supported by any evidence. The *criteria weighting* problem faces four critical challenges:

- Evidence of validity for criteria weights.
- Transparency of DMs' participation in criteria weighting.
- Scalability of criteria weighting.
- Acceptable Cognitive load on DMs.

Evidence of validity for criteria weights refers to the means which can prove that the assumed criteria weight is correct for a DM or to be able to reason why it is correct for a particular DM. Transparency of DM' participation in criteria weighting is necessary to atleast make sure that the criteria weights are not illicitly decided in a manner to favour a particular DM' preferences, or in other words to make sure that every participant DM is satisfied with his bit of contribution in the process. The scalability of the techniques for a sufficiently large number of criteria which may arise in a systems engineering project. In a systems engineering project the number of broad criteria hardly rise more than ten. These broad criteria later can be divided in to multiple criteria in next level. The *criteria weighting* technique should be able to address this hierarchy of criteria. The Fourth most important challenge is about the amount of cognitive load that a technique poses on the DM. If majority of DMs find it difficult to use the technique for weighting the criteria, then the technique has less chances to be used in the process. Whereas, if a technique which poses less cognitive load on the DMs, can easily win over others and find acceptability in the approach, even if it provides less accurate results.

$$\begin{array}{c}
 c_1 \quad c_2 \quad \dots \quad c_n \\
 \begin{array}{l}
 St_1 \\
 St_2 \\
 \vdots \\
 St_m
 \end{array}
 \begin{bmatrix}
 w_{11} & w_{12} & \dots & w_{1n} \\
 w_{21} & w_{22} & \dots & w_{2n} \\
 \vdots & \vdots & \ddots & \vdots \\
 w_{m1} & w_{m2} & \dots & w_{mn}
 \end{bmatrix}
 \circ \implies Gr \begin{array}{c} w \\ [w_1 \ w_2 \ \dots \ w_n] \end{array}
 \end{array} \quad (1)$$

Eq (1) represents the criteria weighting problem mathematically. In Eq (1) St_i refers to i 'th stakeholders, c_j refers to the j 'th criterion, and w_{ij} refers to the weight of the i 'th stakeholder for the j 'th criterion.

3 State of art of Criteria Weighting Techniques

In the literature of multi-criteria decision making, often the problem of *criteria weighting* comes along the multi-criteria decision analysis. The vast literature on criteria weighting techniques found mentions of large number of different techniques, but as in this paper we try to solve criteria weighting technique for systems engineering, we'll

stay with the techniques which have earlier found usage in the systems engineering project. Criteria weighting methods can be divided into two types internal and external. Popularly known internal methods are Entropy method, Regression method, Variance method, and LINMAP method. The external methods can further be divided into two types with DMs matrix such as: Swing method, Utility method, and without DMs matrix such as: SMART Method [11], SMARTER method, Eigenvector, Minimum weighted squares method, weighted sum method (WSM), weighted product method (WPM) [12]. Reference point method for vector optimization, AHP method, PROMETHEE [7] method are equally popular to solve this problems. There are many other techniques based on the visual aids for selecting the right stakeholder weight.

The previously mentioned techniques depend completely on human judgement about the preference weighting to get the weight, but seldom human can provide reasoning about the preference weight. To best of our knowledge, in the literature of decision making, there is no comprehensive way to reason the weight of the decision maker, to allocate systematically the weights at the various levels of the preferences. In this respect our work is very different from the previous work. We provide a mechanism to weight the human perception and link it with the mathematical formulations to derive the criteria weight. To have more robust criteria weighting, we have used multiple algorithms to find the weight.

Another aspect of the criteria weighting problem is about the uniform satisfaction among the stakeholders, pointed out by the well known *Arrow's impossibility theorem* [13], which states that DM can find no procedure that can combine individual's rankings of alternatives to obtain single unified rankings. Recent research works have tried to address *criteria weighting* problem using many other different techniques such as card playing [14], using hybrid approaches of many different techniques, but in almost all of them they started with assumption about the particular criteria weight.

4 Proposed Technique

The proposed technique uses the classical preference modelling [9, 15, 16] for representing stakeholder preferences. The proposed approach is four step process. We assume that all the necessary DMs (stakeholders) and criteria are already identified for the system under study.

4.1 Prerequisite to technique

- ‘ D ’ is the set of decision makers (DMs) involved in the concerned conflict, where $2 \leq |D| < \infty$.
- ‘ C ’ is the set of distinguishable criteria, satisfying $2 \leq |C| < \infty$. For each stakeholder set ‘ C ’ consists of three distinct subsets H, M, L , such that: $\{H\} \cup \{M\} \cup \{L\} = \{C\}$, and $\{H\} \cap \{M\} = \{M\} \cap \{L\} = \{H\} \cap \{L\} = \{\emptyset\}$. Where cardinality of each set can be different. The criteria subsets are such that the perceived utility is in the order $p(H) > p(M) > p(L)$.
- For each $i \in D$, there is set of preference relationships $\{\gg_i, >_i, \sim_i\}$ defined over ‘ C ’.

- For each $i \in D$, a complete binary relation \gg_i on C , specifies DM i 's strong preference over C . If $s, t \in C$, then $s \gg_i t$ means the DM i strongly prefers s to t .
- For each $i \in D$, a complete binary relation $>_i$ on C , specifies DM i 's weak preference over C . If $s, t \in C$, then $s >_i t$ means the DM i weakly prefers s to t .
- For each $i \in D$, a complete binary relation \sim_i on C , specifies DM i 's indifference over C . If $s, t \in C$, then $s \sim_i t$ means that s to t are indifferent to DM i .
- The relation \gg and $>$ are asymmetric, \sim is symmetric and reflexive and the triple $\{\gg_i, >_i, \sim_i\}$ is complete.
- The preference relationships can be transitive for a particular stakeholder.

4.2 Technique

The proposed approach can be divided into four steps as follows:

Criteria Categorization All the decision makers involved in the decision process should categorize the agreed set of criteria in to three sub sets high preference (H), medium preference (M), low preference (L), according to their perception of utility of the criteria. Hence, the sorting of criteria is such that the perceived utility is in the order $p(H) > p(M) > p(L)$.

Preference modelling over criteria Each DM $i \in D$, creates the preference matrix P_i , over the criteria $j \in C$, given by Eq.(2). The previously created three subsets h, m, l are used as reference for creating the preference matrix.

$$P_i = \begin{matrix} & \begin{matrix} c_1 & c_2 & \dots & c_j \end{matrix} \\ \begin{matrix} c_1 \\ c_2 \\ \vdots \\ c_j \end{matrix} & \begin{bmatrix} 0 & p_{12} & \dots & p_{1j} \\ p_{21} & 0 & \dots & p_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{j1} & p_{j2} & \dots & 0 \end{bmatrix} \end{matrix} \quad (2)$$

where the value p_{ab} of a DM $i \in D$ is given by Eq(3) below:

$$p_{ab} = \begin{cases} 2 & \text{If } a \text{ is strongly preferred criterion than } b \\ 1 & \text{If } a \text{ is weakly preferred criterion than } b \\ 0 & \text{If } a \text{ is indifferent to criterion } b \\ -1 & \text{If } a \text{ is weakly disliked criterion than } n \\ -2 & \text{If } a \text{ is strongly disliked criterion than } b \end{cases} \quad (3)$$

As the matrix P_i is skew-symmetric, it is sufficient to express the value of p_{ij} to get p_{ji} and vice versa. Once P_i matrix is available, the criteria are ranked in order with maximum number of 2, 1, 0, -1 and -2 respectively. It is possible for a DM to have two or more criteria securing exactly same ordinal ranking, depending upon the entries in matrix.

Simulating solution The criteria weighting evaluation is always a difficult step for high level stakeholders who have simply limited knowledges. So, we propose a simulation of solutions which would be retained if weights are maintained. This high level simulation is individually processed for each stakeholder. A stakeholder could display the effect of his/her choice of criteria weights. This simulation is based on data bound to some system components. at the beginning of a new project, system experts could draw a very simple organic (or functional) system breakdown. During this step, they can add on each high level component approximative values for criteria. This approach is suitable because system experts have skills on available technologies. It doesn't need to precisely know the features of each components, actually it doesn't need to precisely know the real system breakdown. At this step, system experts can only categorize the parts of solutions into sets of criteria values, for example they can chose between low, medium or high. The aim of this simulation is to provide stakeholders an approximative solution if they confirm their criteria weighting. Let's illustrate this approach with a very simple example. This one deals with the design of a new individual powered transport system proposed to city inhabitants for their short distance journeys. Let the stakeholders be: end user, technical engineer of the product development company, business engineer of the product development company, and city representative for public transport system. In a very simplified way, let design criteria be: economical, environmental, reliable, innovative, and repairable. Without knowing the details of solution, design experts have some element of solution as the breakdown as concerned, for example: chassis, engine, energy tank, and transmission. For each element of this breakdown, they are able to give some categories of solutions. Fig.1 gives an example of partial breakdown.

For each categories of solution, system experts could add approximative criteria values. The chassis could be in carbon, magnesium, aluminum or in polyethylene. For each technology, subject matter experts are able to approximatively weight each element of this structure.

For each technology, the expert is able to make a "categorization" for alternative solutions based on the proposed criteria set, completely without the specifications, it is not accurate sizing, but he knows a ranking of basic technologies. This simulation does not attempt to validate a solution but tries to establish a more precise relationship between a criterion and a family of solution. Consider the choice of the two following stakeholders. The sales engineer wants a conveyance which is economical, reliable and robust. It will therefore fill the his preference matrix given by Eq.(2) accordingly. He may not have realized that his criteria preference matrix guides the choice solution to plastic chassis and using an old noisy and polluting engine.

The ecological criterion may not be a priority for him, but the solution proposed would not be very attractive commercially. It would be difficult to place the product in the market demand. Similarly, considering the mayor of the city, responsible for public transport system. He wants an ecological and innovative solution, that represents modernity and dynamism of the city. He does not realize that his preference matrix may indicate towards the solution of the latest technologies that are likely to be less reliable and expensive to maintain for a product designed for intensive use. In these two examples, we see that the perception of a stakeholder to the criteria is necessarily biased. The head of the city was not opposed to having a robust and reliable, but he can interpret

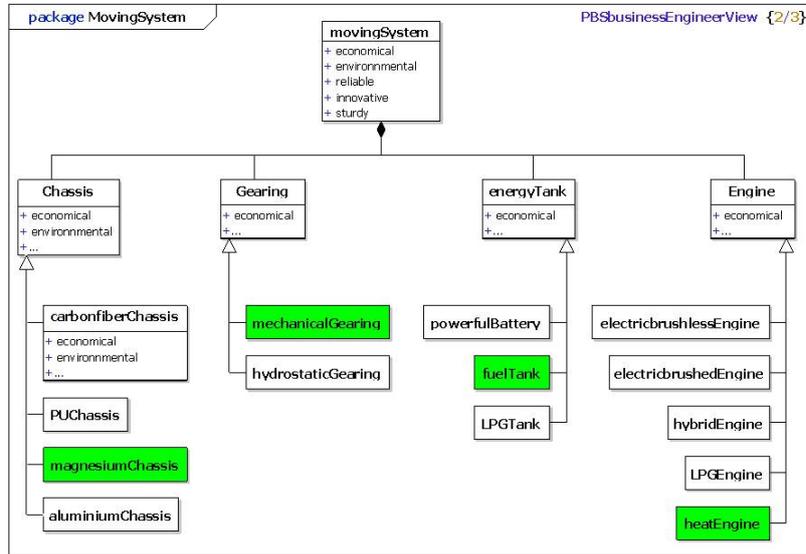


Fig. 1. Solution according business engineer center of interest

these criteria with his knowledge, which are a priori limited on technologies. His perception represents the criteria “robust” and “reliable” with respect to its repository of life for the project in question: a modern car, a scooter, ... To summarize, this simulation is a means proposed for each stakeholder should help when choosing a value for the weight of importance of the design criteria. Simulation offers the ability to view a draft solution based on criteria elements informed by expert systems.

Generating scores The scores are generated using a simple process consisting of a maximum of j number of moves, where $j = |C|$, in which every DM starts marking the criteria set C . The scores can be generated using the utility functions shown in Figure 2, depending upon the risk averseness, proneness or neutrality of the decision maker. Below, we define some risk neutral utility functions that we employ in our example in Section 5.

1. Every DM starting with the most preferred configuration to the least preferred state.
2. Individual score of every configuration is calculated and the one with the highest value.

The scores of every configuration can be calculated using the equation (4), and normalized using (5).

$$score(c_i) = \frac{(k - j.i)}{k} \tag{4}$$

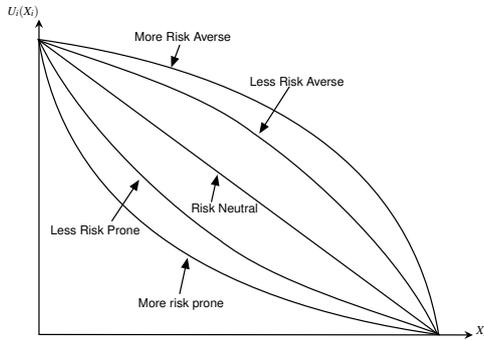


Fig. 2. Decreasing Utility function [3]

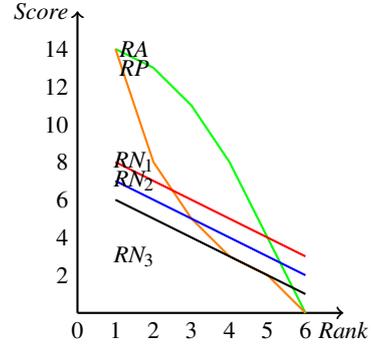


Fig. 3. Score vs. Rank

where k, j are index terms used to generate different types of utility functions.

$$Normalised\ score(c_i) = \frac{score(c_i)}{\sum_i c_i} \quad (5)$$

Once the scores are calculated they are normalized to obtain the score.

5 Example

To show the ease of usage of our technique, we present here an example of Criteria weighting problem. We take an example of a hybrid car. The design team in a automotive industry wants to design a hybrid car and they need to weight design criteria to proceed with the selection of alternatives. Let the set of DMs be $D = \{a, b, c, d, e\}$, the criteria set be $C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$. The interpretation of various criteria is explained in Table 1.

Table 1. Design Criteria

Criteria	Description
c_1	Flexibility of usage
c_2	Maintainability
c_3	Robustness
c_4	Aesthetic value
c_5	Environment Friendly
c_6	Ease of manufacture

Table 2. Design Criteria categorization

DM	h	m	l
a	c_1, c_5	c_2, c_4	c_6, c_3
b	c_4, c_6	c_1, c_2	c_3, c_5
c	c_3, c_4	c_2, c_1	c_5, c_6
d	c_6, c_3	c_2, c_1	c_4, c_5

Step 1 The DMs categorize criteria set according to their perception of criteria as shown in Table 2.

Step 2 The DMs create their preference matrices according to their categorization from step 1 as shown in Table 2. The preference matrix of stakeholder a, b, c and d are shown in Eq.(6),(7), respectively.

$$P_a = \begin{matrix} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & 1 & 1 & 2 \\ -1 & 0 & 2 & 1 & -2 & 2 \\ -2 & -2 & 0 & -1 & -2 & 0 \\ -1 & -1 & 1 & 0 & -1 & 1 \\ -1 & 2 & 2 & 1 & 0 & 1 \\ -2 & -2 & 0 & -1 & -1 & 0 \end{bmatrix} & , & P_b = \begin{matrix} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{matrix} & \begin{bmatrix} 0 & 1 & 2 & -2 & 1 & -2 \\ -1 & 0 & 1 & -2 & 2 & -1 \\ -2 & -1 & 0 & -2 & 1 & -2 \\ 2 & 2 & 2 & 0 & 2 & 1 \\ -1 & -2 & -1 & -2 & 0 & 2 \\ 2 & 1 & 2 & -1 & 2 & 0 \end{bmatrix} \end{matrix} \quad (6)$$

$$P_c = \begin{matrix} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{matrix} & \begin{bmatrix} 0 & 1 & -1 & -1 & 1 & 1 \\ -1 & 0 & -1 & -1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 1 & -1 & 0 & 2 & 2 \\ -1 & -1 & -2 & -2 & 0 & 1 \\ -1 & -1 & -2 & -2 & -1 & 0 \end{bmatrix} & , & P_d = \begin{matrix} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{matrix} & \begin{bmatrix} 0 & -1 & -1 & 1 & 1 & -2 \\ 1 & 0 & -1 & 1 & 1 & -2 \\ 1 & 1 & 0 & 2 & 2 & -1 \\ -1 & -1 & -2 & 0 & 1 & -2 \\ -1 & -1 & -2 & -1 & 0 & -2 \\ 1 & 1 & 1 & 2 & 2 & 0 \end{bmatrix} \end{matrix} \quad (7)$$

Step 3 The DMs carry out the criteria marking according to their preference matrices. Criteria with higher preference are marked early and one with lower are marked later in the order of preference.

$$Marking = \begin{matrix} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} & \begin{bmatrix} \textcircled{a} & 0 & \textcircled{c} & \textcircled{b} & 0 & \textcircled{d} \\ 0 & 0 & \textcircled{d} & \textcircled{c} & \textcircled{a} & \textcircled{b} \\ \textcircled{b} & \textcircled{c,a,d} & 0 & 0 & 0 & 0 \\ \textcircled{c,d} & \textcircled{b} & 0 & \textcircled{a} & 0 & 0 \\ 0 & 0 & \textcircled{b} & \textcircled{d} & \textcircled{c} & \textcircled{a} \\ 0 & 0 & \textcircled{a} & 0 & \textcircled{b,d} & \textcircled{c} \end{bmatrix} \end{matrix} \quad (8)$$

After the marking is done, the DMs agree on a set of score generating algorithm. By using the score generating functions as mentioned in Eq.(4) with suitable index terms k and d the scores are obtained. Then normalized using Eq.(5). Similarly other scores are obtained by the various risk averse, risk prone and risk neutral functions by changing the index terms k and d . The obtained scores are shown in Table 3, using three risk neutral utility functions: $Risk - N1$, $Risk - N2$, $Risk - N3$; risk averse utility function: $Risk - A$, risk prone: $Risk - P$ and a rank order centroid function (ROC). Figure 3, shows the resulting graphs obtained through various utility functions used.

Once the criteria weight matrix is available, the final criteria can be obtained by either by the mean of weights obtained for each criteria or by accepting any particular

Table 3. Design Criteria scores

Criteria	Algorithms (Scores-Normalized Scores)					
	Risk-N1	Risk-N2	Risk-N3	Risk-P	Risk-A	ROC
c_1	16/6 (0.19)	20/7 (0.182)	24/8 (0.182)	24 (0.197)	33 (0.168)	10 (0.163)
c_2	15/6 (0.1785)	19/7 (0.173)	23/8 (0.1742)	18 (0.1475)	41 (0.209)	10-(0.163)
c_3	14/6 (0.167)	18/7 (0.163)	22/8 (0.167)	21 (0.172)	31 (0.163)	10 (0.163)
c_4	16/6 (0.1904)	19/7 (0.2)	24/8 (0.182)	23 (0.1885)	39 (0.1989)	10 (0.163)
c_5	9/6 (0.107)	13/7 (0.118)	17/8-(0.123)	12 (0.0983)	17 (0.097)	10 (0.163)
c_6	14/6 (0.167)	18/7 (0.164)	22/8 (0.167)	24 (0.197)	31 (0.163)	10 (0.163)

criteria array. In this example we took the mean of the four array of criteria weights and the resultant criteria weight array is represented by Eq.(9).

$$\begin{array}{l} \text{sol} \\ \text{entropy} \end{array} \begin{array}{c} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{bmatrix} 0.1838 & 0.176 & 0.1664 & 0.192 & 0.109 & 0.1716 \\ 0.1558 & 0.2254 & 0.2266 & 0.1483 & 0.0379 & 0.2057 \end{bmatrix} \quad (9)$$

6 Discussion

Our approach tried to provide a methodological solution to one of the ignored problems of multi-criteria decision making. Most of MCDM techniques mentioned in the literature assume that they can choose criteria weight just by barely looking at them. Often, this is not the case. We provide a technique which allows multiple DMs to achieve an appropriate array of criteria weights, while remaining in synergy with other DMs criteria weights. One of the benefit of our approach is about the scalability of the technique, it can easily take in account a large number of DMs with different perception of criteria weights and large number of criteria. But with respect to systems engineering, it would not be fruitful to employ very large number of criteria. There are seldom more than twelve criteria in a project. If the number of criteria are too many it would be advised to create a hierarchy of criteria and then use our approach recursively.

The addition to ease of application, our technique provides other multiple benefits. It helps a DM to understand his perception of criteria better and of the other DMs. It also helps to avoid conflicts among the DMs, which often arrive during the decision making process. The transparency of the approach allows easy negotiation of the criteria weights and hence maximum number of DMs are satisfied with their contribution. The low cognitive load that our technique demands can be important factor for acceptability of the technique. Our technique can be coupled with a variety of decision-making approaches, such as PROMETHEE, TOPSIS, WSM, WPM.

The decision makers are free to propose their own utility functions or scoring algorithms based on the ordinal ranking achieved. This allows a more conducive environment for criteria weight negotiation, as the whole process is transparent, with no black-box process involved.

In our approach, we considered that all the DMs have equal weight, but this may not be the case in a real systems engineering project. It is possible that the different stakeholders in the projects have different importance. But this is only upto Project manager to allocate the weights to various DMs. Our approach offers possibility to weight the different DMs involved in a project. In this case only one preference matrix would be required, i.e., only of Project manager. While, weighting DMs it would be advised to use a set of risk averse and risk neutral utility functions to weight them, in order to avoid penalizing the low ranked DMs.

There can be arguments that the utility score generation algorithms cannot be accepted as weighting mechanism. But the literature shows that, every individual DM has a utility function, which he uses consciously or unconsciously while providing scores directly as it usually happens. Here we attempted to provide a formalism to use this conscious/unconscious utility function, in order to help other DMs to understand the perceptions of each other. The benefit that our approaches provides over other is that, it involves the DMs to methodologically provide the ordinal ranking by first demanding them to attempt to categorize them in three categories. This categorization provides input to the next step, which can again be validated by corresponding DM; depending on the preference of the various DM's, a range of scoring algorithm can be applied and weights can be obtained.

In the beginning our approach demands slightly more participation from the DMs, as compared to the other approaches, but once the weights are methodologically obtained they are certainly more reliable then the other contemporary approaches with least amount of conflict. Better decisions allow to design the right systems, with more stakeholders getting more confident about their product.

7 Conclusion and Future perspectives

In this current work, we have provided a systems theory of how the criteria weights can be obtained using the classical theory of preference modeling. We call our technique Utility Rank Order Weighting (UROW) technique. This approach provides multiple benefits with compared to other existing approaches. Usually in systems engineering project, the engineers rely upon their intuition to provide weights, and later use other technique to combine the different DMs' preferences. Our approach provides a formalism to this systems engineer intuition and hence provides the reasoning for the various weights achieved. Our approach is very easy to understand and use, and demands very low cognitive load from the engineers and stakeholders. It allows to formally provide the scores using the DM' drawn utility functions: risk prone, risk averse, or risk neutral; it provides a mechanism to combine them together to come up with a uniformly acceptable solution. In future, we look forward to link the simulation of the DMs' preferences with the design library, in order to shorten the decision time. Our approach can easily be applied to the class of methods which require information on the attributes to carry out a decision analysis.

References

1. Sage, A., Armstrong, J.: Introduction to systems engineering. Wiley series in systems engineering. Wiley (2000)
2. Saaty, T.L.: How to make a decision: the analytic hierarchy process. *European journal of operational research* **48**(1) (1990) 9–26
3. Keeney, R., Raiffa, H.: *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press (1993)
4. Roy, B.: Classement et choix en présence de points de vue multiples (la méthode electre). *Riro* **2**(8) (1968) 57–75
5. Roy, B.: Electre iii: Un algorithme de classement fondé sur une représentation floue des préférences en présence de critères multiples. *Cahiers du CERO* **20**(1) (1978) 3–24
6. Roy, B., Bertier, P.: La méthode electre ii (une application au média-planning...). (1973)
7. Brans, J.P., Mareschal, B.: Promethee methods. *Multiple criteria decision analysis: state of the art surveys* (2005) 163–186
8. Hwang, C.L., Yoon, K., et al.: *Multiple attribute decision making: methods and applications: a state-of-the-art survey*. Volume 13. Springer-Verlag New York (1981)
9. Fishburn, P.: *Utility theory for decision making*. Technical report, DTIC Document (1970)
10. Ulrich, K., Eppinger, S.: *Product design and development*. McGraw-Hill (2008)
11. Edwards, W.: How to use multiattribute utility measurement for social decisionmaking. *Systems, Man and Cybernetics, IEEE Transactions on* **7**(5) (1977) 326–340
12. Triantaphyllou, E.: *Multi-criteria decision making methods: a comparative study*. Volume 11. Kluwer Academic Publishers Dordrecht (2000)
13. Arrow, K.: *Social choice and individual values*. Volume 12. Yale university press (1963)
14. Figueira, J., Roy, B.: Determining the weights of criteria in the electre type methods with a revised simos' procedure. *European Journal of Operational Research* **139**(2) (2002) 317–326
15. Roy, B.: The outranking approach and the foundations of electre methods. *Theory and decision* **31**(1) (1991) 49–73
16. Roy, B., Vincke, P.: Relational systems of preference with one or more pseudo-criteria: Some new concepts and results. *Management Science* (1984) 1323–1335

Reinventing Goal-Based Requirements Modeling

Vikas Shukla and Guillaume Auriol

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ. de Toulouse, INSA, LAAS, F-31400, Toulouse, France
vshukla@laas.fr gauriol@laas.fr

Abstract. Understanding user needs, requirements, architecture specifications, and design specifications for a system holds up-most importance in a systems engineering project. The early phase requirements engineering deals with elicitation of goals, objectives and environment of the system under development and determine the needs and requirements of the various stakeholders. The needs, requirements should traced to the various rationales of stakeholder with their preferences. In this paper, we provide a goal-based requirements modeling methodology and language to understand the needs, requirements,... and represent them in a much clear manner to improve the quality of requirements written for the project and their early phase traceability. We also integrate the preference modeling of the various stakeholders for the various goals and hence provide a traceability for the requirements and their preferences in early phase of requirements engineering.

Keywords: Requirements Modeling, Goal-Based Requirements Engineering, Stakeholder preference, Traceability, Features.

1 Introduction

The primary goal of the *systems engineering* (SE) is the creation of a set of high quality products and services that enable the accomplishment of desired tasks and needs of the clients or user groups. Typical systems engineering project can be divided in to three phases: definition, development, and deployment [1]. A systems engineering project involves multiple stakeholders, in their various forms of roles and actors, during and after the previously mentioned SE project phases. These stakeholders and their various roles lead to various needs and requirements. *Requirements engineering* (RE) activities en-globe activities like stakeholder identification, requirements elicitation, modeling, analysis, documentation, verification and validation, negotiation, etc. All these activities require certain understanding of the requirements itself.

Poor RE is the reason for majority of all the challenged and failed SE projects. Many empirical studies previously carried out have indicated that poor RE leads to poor requirements, faulty design, poor requirement traceability, rework and cost/budget overflows [2–4]. Requirements modeling is carried out during early phase of the RE. There are a few of approaches like: Goal-Oriented RE (GORE), Scenario-Based RE (SBRE), Problem-Based RE (PBRE), and Aspect-Oriented RE (AORE). The two most popular and referenced modeling methodology are goal-based and scenario-based RE, owing to the benefits and ease it provides during the requirement modeling phases. In

this paper, the two referenced and used methodologies are goal-based and scenario-based RE. There are tools built upon popularly known GORE methodologies i* and KAOS, and have reported lots of success in industrial application because of ease of understanding it offers to both technical and non-technical stakeholders. But still there is some scope of improvement and certain difficulties and problems to be improved as discussed later in Section 2. RE research has focused on goals as a way of providing the rationales (why) for a system under development [5].

In this paper, we provide ways to model the requirements of the core and optional features of the system from early phases of RE and equally the stakeholders' preferences associated to them. We provide ways to visual modeling, which are more scalable and comprehensible to the engineers and the various stakeholders. We show how the viewpoints based modeling can be carried out from the very beginning and be carried out separately and integrated later. We call our proposed approach Comprehensive Requirement Modeling Language (CReML), which is based on Goal-Oriented Requirements Engineering (GORE) methods. CReML can be used complementarily with UML and hence help to provide better design specifications and insights to the system under study. Previously, it has been argued and shown that the GORE and SBRE complement each other during the requirement modeling phase of RE [6].

The major contributions of this paper include, a GORE based requirements modeling language with preference modeling and provisions for core or optional goal feature modeling together with much needed traceability and notations for domain assumptions. Demonstration of our CReML tool with an example demonstrating various aspects of the CReML developed.

This paper is organized as follows: Section 2 presents the early phase RE problems and goal-based RE. Section 3 presents the relevant state of art of goal-based RE. Section 4 presents our proposed approach. Section 5 presents an example of our approach. Section 6 concludes and presents future perspectives.

2 Early phase requirements engineering problems

Early phase requirements engineering start once the requirement elicitation process starts following to the interviews, questionnaires, ethnography and other elicitation techniques mentioned in literature. Through all the elicitation techniques the information gathered is converted to textual documents, often known as user-stories. These user-stories form the foundation of the requirements modeling (RM) processes. We have identified a few of the problems which seek attention and proper resolution during RM. ***Ease of Scalability:*** recent empirical studies using i* GORE methodology have shown that scalability can turn out to be big problem when modeling requirements for a sufficiently large projects either with different viewpoints or integrated modeling [7]. GORE based approach needs to be organized in a manner that their comprehensibility is independent of number of participants and number of requirements and views used during the modeling. ***Traceability of goals:*** often, during RM goals are elicited through stakeholders and as the project evolves, the complexity of the models may increase to an extent where it becomes tedious task to answer why a particular goal exists in the model and which particular stakeholder needs it. Also, it can be equally cumbersome to link a goal to a particular user

story previously elicited, owing to syntactic differences [7]. There is need of dedicated mechanism to link goals to the user-stories previously documented during elicitation. **Preference of multiple stakeholders over goals:** in a systems engineering project, it is of great importance that most of stakeholders are satisfied with the various decisions taken during the product development and with the final resulting end product. A higher satisfaction among the stakeholders can be guaranteed if the various stakeholders preferences are taken into account in a transparent and holistic manner during the goal modeling. **Multiple view-point requirement modeling:** during the requirement modeling multi-view point modeling is often instrumental in understanding the system under study. Multi-view requirements modeling allows separation of concerns and can help to elaborate particular aspects of the system under study. But it becomes tedious task to combine these multiple views and present one single coherent and comprehensive models. Often, the resultant combined model is inconsistent and hard to understand, which demands significant amount of resources. Need of multi-view modeling has been often demanded in many previous works for particular actors, traceability and events [8, 9]. **Modeling of core and optional features:** modeling of core features and optional features from the very beginning of project can allow the engineers to have better understanding of the systems under study and lots of effort and resources can be saved if they can be modeled in early phase of RM. There are some dedicated feature modeling languages in the literature but this often leads to redundant efforts. **Representation of domain assumptions:** domain assumptions or beliefs are often implicit during the requirement modeling but often lead to goals and requirements. They are usually held by the stakeholders and sometimes designers. Their implicit nature during the RM may cause potential traceability errors and may cause worries for the quality control of the product. Domain assumptions usually become the basis of many decisions during the RE activities, they too need to be given requisite attention.

3 State of Art

Literature of GORE based methodologies is vast if we take in account all the RMLs mentioned. There are a few notable GORE frameworks such as i*, Tropos, KAOS, GBRAM, NFR, etc., but still there are various aspects to be improved upon as mentioned in Section 4. As previously mentioned, our approach refers to i* and KAOS owing to the proximity of our approach. i* and KAOS frameworks are apparently the two most popular GORE methodologies. The seminal work of Erik Yu [10] introduced the i* framework. It is hard to provide a fair comparison between them, as both of them have certain benefits and disadvantages when compared to each other [11]. Underlying principles of GORE were reinforced by Regev *et. al.* [12] by bringing together the various concepts from various methodologies. Recent works involving goals, preferences, and inconsistency have led to development of an abstract requirement modeling language called Techne [13]. Recent work have tried to address the optionality and preference of the requirements during the modeling [14]. The preference of the goals are marked on the the goal notation thus allowing to evaluate the importance of one goal with respect to another. Goal argumentation methods (GAM) were introduced to make it explicit the reasons of selecting a goal [15].

Tropos [16] framework was founded on social and intentional aspects of information system, used requirement modeling based on their operational environment. Tropos introduced textual syntax to allow the later phase formal analysis of the early requirements models done using i^* to represent their social milieu. The major advantage given by i^* based frameworks is they allow to represent the strategic relationships existing between the various stakeholders of the project. But many empirical studies [7, 11] have shown that the readability of the models designed using i^* are greatly marred when the number of participants is sufficiently large. This problem comes due to layout used by the i^* models, on the contrary it can be argued that the tree based layout of KAOS models are much better in this aspect of the goal modeling. One of the disadvantages of the KAOS models is its deficiency in modeling the strategic relationships of the stakeholders. Previously mentioned techniques for integrating preference in goal models do not help user in any readability aspect [14, 13]. It can be argued that surcharge of information increases the pressure on the designer or stakeholders. The way the data is represented and made available to the design engineer can be rendered more readable.

Currently, there are a variety of tools available for the GORE modeling such as Objectiver based on KAOS [17], recently introduced RE-TOOLS [18]. There are a few of light weight RMLs introduced recently like VLML [19]. Still there are numerous issues to be solved by a RML and lots of lessons learned during all these years of RE needs to be brought together to a standard GORE language. A standardized GORE notation based on KAOS seems to be most appropriate for this unifications of lessons learned and hence we propose in this paper few modifications into the KAOS modeling notations to the benefit of RE community.

4 Proposed Goal-Based RE Language

Our approach aims to use and devise techniques previously used and matured in domain of software engineering for the benefit of systems engineering community. Many advances in the RE community come from software industries. These advances provide new opportunities to systems engineer to make their process more lean and mean. Our approach is designed for early phase requirements engineering. It is not here to replace use-cases or requirements block used in UML/SysML, it is their precursor and complementary technique to them in RE activities.

4.1 Comprehensive Requirements Modeling Language(CReML)

Our approach identifies nine types of early phase RE artifact: *Stakeholders, Goals, Rationale, Viewpoint, Objectives, Constraints, Domain Property, Assumptions and Requirements*, we identify three types early phase relationships : *Contribute, Derive, Conflict*. Requirement artifact diagrams and their semantic meanings is shown in Table 1. They are used to address the problems previously raised in Sections 2 and 3. UML components such as class diagrams and use cases can also used complementarily to enrich CReML models. There are three types of models introduced in CReML: *Goal models, Responsibility model and Strategy model*. *Goal model* and *Strategy models* are developed simultaneously. Strategy models can be started once the goal modeling begins. *Strategy*

Table 1. Requirement Artifacts definition in CReML

Graphical representation	Semantic meaning
 Goal	<i>Goals</i> represents the fundamental state, that the stakeholder would like to achieve by using the system under study. A system can have one primary goal and many other secondary goals. Goals may not be strictly measurable or tangible but stakeholders agree are upon certain conditions for determining the acceptability of goal by system under study.
 Rationale	Rationales are the fundamental reasons, why the goals needs to be achieved by the system under study. Rationales are extracted from stakeholders, and often a single stakeholder may have multiple rationales. These rationales are actually linked to various responsibilities and roles played by a given stakeholder.
 Viewpoint	System viewpoints specify, from the developer's perspective, what characteristics system has to possess and with what magnitude in order to satisfy goals.
 Objective	<i>Objectives</i> are the measurable set of tasks and conditions, which the system needs to meet in order to satisfy customer. Goals projected with a specific viewpoint lead to an objective in a direction of particular viewpoint to achieve the goal.
 Constraint	<i>Constraints</i> are the limitations imposed on the system by the non-development stakeholders or they may represent some challenges to overcome by the system.
 Domain-Property	A domain-property can defined as a knowledge or information about the domain of the system under study which is uniformly acceptable by all stakeholders: technical or non-technical and which can be verified and validated using scientific methods.
 Assumption	Assumption are hypothesis or non-verifiable information or condition which is considered valid for the system under study. They are close to domain-property but domain-properties can be verified and easily validated.
 Requirement	<i>Requirements</i> specify, from the stakeholders' viewpoint, what characteristics it is to possess and with what magnitude in order to achieve the stakeholders' objectives. Requirements are derived from a particular or a set of objectives, constraints, assumption and domain properties.
 Conflict	<i>Conflict</i> relationship is used to represent the conflict occurring between the two objectives, or two requirements and hence between the source stakeholders. Conflict means that the implementation of the two requirement artifact cannot be achieved by the system under study at the same time.
 Derive	<i>Derive</i> relationships represent the parent-child relationships existing between the various requirements artifacts. A derive relationship exists between the goal and rationales, rationales and viewpoint, viewpoint and objective, objective and requirements, constraints and requirements, domain-property and requirements, assumption and requirements, and between requirements and requirements.
 Contribute	<i>Contribute</i> relationship is used to represent the direct contribution of information about requirement artifact from an stakeholder for system under study.
 Stakeholder	<i>Stakeholders</i> of the project are the entities which have genuine interest in the project. They are of two types: stakeholders from the client side or end-users and stakeholders responsible for the development of the project. In our terminology, we use Agents also as an stakeholder, as they interact with the system.

model and *goal model* provide inputs to each other over different iterations to better understand the system goals and environment. *Responsibility model* is development can be carried out once the *goal-model* and the *strategy model* are available.

In **first stage** Goal models and strategy models are developed simultaneously. For *Goal modeling* the stakeholders are identified and their goals are extracted out of their corresponding available user stories. Since the stakeholders contribute the goals the relationship between the stakeholders and goal is called *contribute*. These stakeholders are then analyzed and their responsibilities are analyzed taken in account both in presence and absence of the system under study. This analysis leads to various rationales of the corresponding stakeholders role. These rationales of the stakeholders provide the basis of strategy modeling. Rationales provide the inputs regarding how the different stakeholders can be satisfied. Strategy models provide the statements (in form of rationales), which bind together the stakeholders and the statements for their potential conflict. At this stage stakeholders' preference about the various goals are gathered and core and optional goals are identified. This preference gathered over the goals from different stakeholders provides the inputs for the strategy formulations for system development. Stakeholders' preference about traceability of various rationales are also gathered, they are later used to formulate the traceability policies according to the needs of the stakeholders.

In the **second stage**, the various viewpoints are which are of concern to each stakeholders are gathered; a viewpoint is a particular aspect of the system under study which is of interest to stakeholder: client or developer. The analysis of viewpoints lead to the formulation of objectives corresponding to a particular stakeholder. This allows us to understand very clearly the capabilities stakeholder wants to acquire with respect to each viewpoint. At this very stage, potential conflicts among the objectives can be observed, a conflict relationship is marked for further resolution and negotiation for the magnitude of accomplishment of particular objectives.

In the **third stage**, the *responsibility model* is created by separately mapping the stakeholders, viewpoint, objectives, constraints and assumptions. Mapping of objectives with the actors (roles of stakeholders) allows to model the responsibility and point of interactions between the agents (roles of stakeholders) and system. This mapping allows to determine the constraints and assumptions held by the stakeholders and their interrelationship.

In the **Final stage**, the *Goal model* is enriched with the assumptions, and constraints previously extracted from stakeholders. Developers held domain properties are included at this stage to transform them together with objectives into achievable requirements. Each objective may lead to one or more requirement.

Tool Implementation of CReML We implemented a software platform which supports CReML, the platform is called **SysEngLab**: it is composed of three major components: requirements engineering module, decision-engineering module [20], and reliability engineering module. In this paper we are concerned with requirements engineering module and partially with decision-engineering module. The capabilities of requirements engineering module is discussed in this section:

Tagging User Stories with Goals: During the stakeholders requirement elicitation process various techniques are used to elicit the stakeholder requirements. Often, empiri-

cal studies have shown that during the first encounter with the clients the needs, desires, and wants are first hand written down in natural languages. The implemented tool allows to create tags for the various user stories based on the goals determined allowing to keep trace of exact origin of a goal in a user story. Often, these requirements represented by various stakeholders also represents the various roles attached to them, which are sometime hidden in the first iteration of the project. The stakeholder identification process first should be carried out to determine all the potential stakeholders with their all potential roles. With each of their roles there are some potential rationales attached. Requirements are actually projection of these rationales in stakeholders' statements often known as user needs or stakeholder requirements. This information which provides links to the stakeholder requirements and various roles is critical for providing the traceability in the later stages.

Integrated Traceability: The problem of traceability lies actually the way it is done. Usually, requirement traceability is carried out when it is demanded by the quality control departments, i.e., when it is solicited. Our proposed tool tracks the links from the very beginning of the goal modeling, every requirement artifact is linked to its parent and child artifact and the root is linked to the user stories. RE module in our tool allows to model the traceability preference of the system, from the very early stage the system tracks which stakeholder has more affinity to which goal and in which viewpoint. As the goal models can be bridged to some of UML/SysML diagrams (Use-case and Block definition diagram) the traceability is continued to the next stage of system design.

Modeling preference In a systems engineering project, it is of great importance that most of stakeholders are satisfied with the various decisions taken during the product development and with the final resulting end product. A higher satisfaction among the stakeholders can be guaranteed if the various stakeholders criteria weights are taken into account in a transparent and holistic manner. Preference modeling if the goals and prioritization of requirements is essential activity, our tools allows to elicit and model both of them over the goal and requirements notations. Unlike goal preference modeling in [14], our approach takes in account the preferences of group of stakeholders and calculates the net preference of each goal using the integrated decision module, and shows it above the goal notation. Similarly, the core and optional features of the system under study can also be marked to keep track of requirements of a product line.

Including Boiler Plates: The requirements diagram used in goal-modeling are equipped with boiler plates mentioned in [3, 21] to help user to write the requirements. The boiler templates aide user to put their capability, capacity, constraint, performance requirements and other type of requirements.

Generation of reports: Our tool supports automatic generation of reports supporting various types of formats. The requirements specification document can be generated in pdf format, user stories can be exported using excel, goal and responsibility models can be generated in image forms, traceability information can be generated in form of matrix with demanded parameters.

5 Application Example using *CReML* and *SysEngLab*

To show the ease of usage of *CReML*, we present here an example of goal modeling using preference and other features previously mentioned implemented in our developed tool *SysEngLab*, Figure 2 shows an screen-shot with user stories. The current example is based on IBIS project currently under implementation in our research group. The project aims at developing a platform which is capable of demonstrating to show the life-cycle of simulation based systems engineering for an aircraft. We take simplified parts of the original diagram developed for the project to show the various capabilities of *CReML*.

Fig. 1. Client-Stakeholder analysis

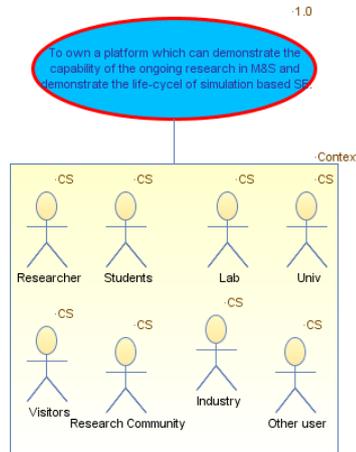


Fig. 2. SysEngLab Screen shot

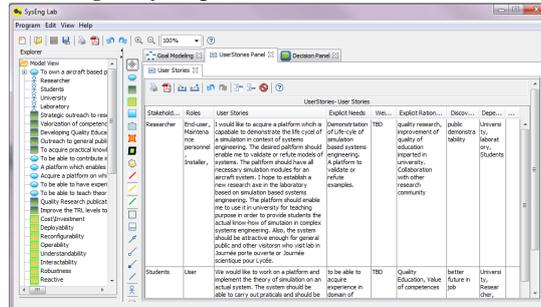


Figure 1, shows the potential client stakeholders of the project. The identified client stakeholders are: Researcher, Students, Laboratory, University, Research Community, Industrial partners, General public and Other users. Notation 'CS', DS, and 'A' above stakeholders represent the type of stakeholders client-stakeholders, developer stakeholder and actor/agent respectively.

Figure 3, shows the four high priority stakeholders in the goal map, primary goal is derived from the Researcher stakeholder from his user story and the secondary goals are extracted from the other three high priority stakeholder from their user stories. The rationales are the reasons for which they need their goal to be accomplished. The notation above the secondary goals marks their weight in the project, which is decided by the decision makers, *SysEngLab* allows to weight the subgoals using Utility Rank Order Weighting technique (UROW) [20], which is integrated with its decision support module. Figure 4, shows the various rationales and derived viewpoints which are of interest to them. As it is clear that the diagram gets more and more complex. *SysEngLab* allows to export a particular rationale and associated viewpoint in a separate files while keeping trace with the original file. This allows to concentrate on a particular rationale and the related viewpoints. In Figure 5, we take only two viewpoints from the previous diagram

Fig. 3. Goals, subgoals, Rationales analysis

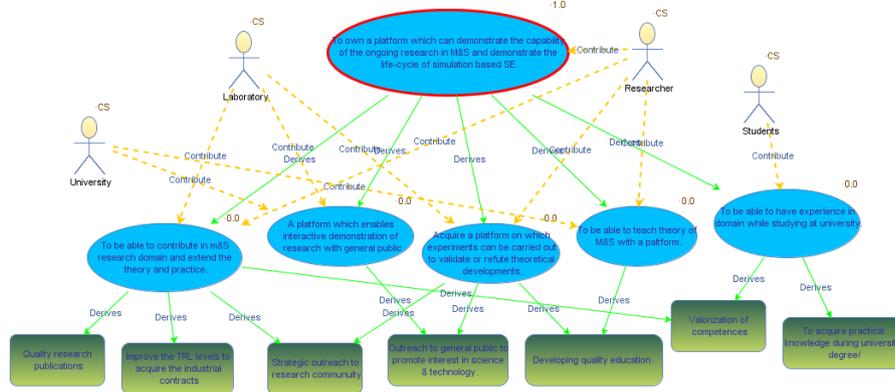
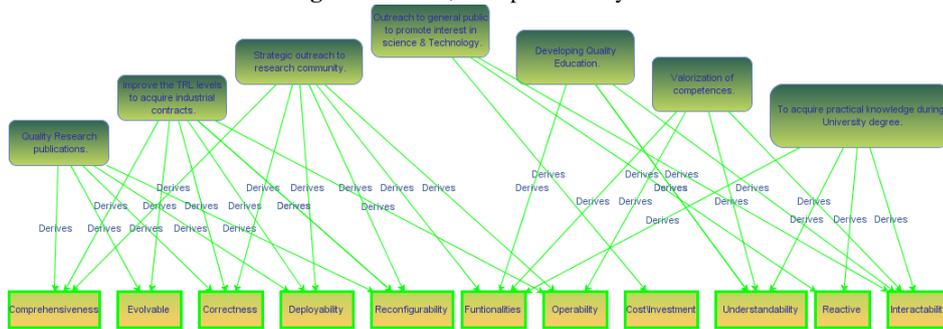
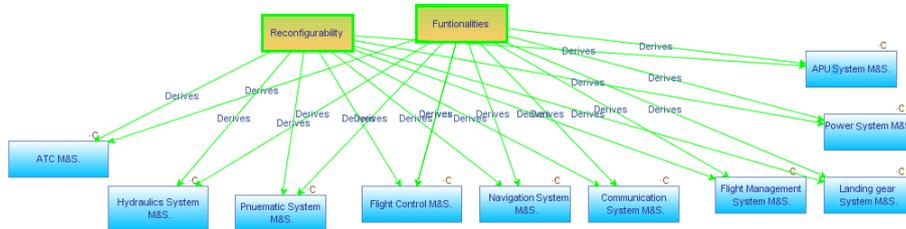


Fig. 4. Rationale, Viewpoints analysis



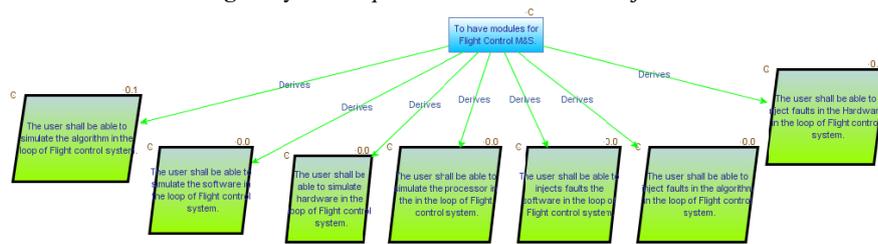
and show which are the objectives derived from the two viewpoints: Reconfigurability viewpoint and Functionality viewpoint. The objectives derived are listed in diagram into various functional and reconfigurable systems. Figure 6, shows the derived requirements

Fig. 5. Viewpoints and objectives analysis



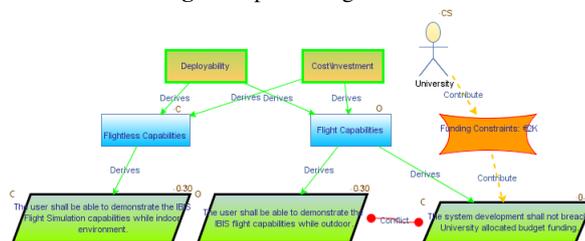
from the objective representing the need for module of flight control system Modeling & Simulation. The notation 'C' above the objective shows that it is core objective and the numbers and notations 'C' or 'O' above the requirements diagram show core requirement or optional requirements and digits show their weight in the objective to be realized. Figure 7, shows how the conflicts can be represented easily between the two requirements.

Fig. 6. System requirements derived from objectives



Conflict relations mark the impossibility of co-existence of two requirements without negotiation.

Fig. 7. Representing Conflicts



6 Conclusion and Future perspectives

We proposed a graphical modeling language which is capable of functionalities typical to popular GORE techniques like i* and KAOS and other functionalities which are of concern to systems engineers and other stakeholders. Proposed language and supporting tool allows requirement engineer to represent the preferences of the various stakeholders on the various goals and objectives. It allows to model both the core and optional features of the system under study. The goals can be traced back to the user stories which are linked to the goal modeling diagram. The responsibility and interaction among the agents is separately modeled and can be integrated if the developer wishes. The other interesting capability our tool provides is to model the rationales using view-points. The

stakeholder rationales are projected and divided into various viewpoints from the very early stage, which allows to better understand the user requirements. The end-product of goal modeling leads to system requirements which can be allocated to the UML/SysML diagrams. Our tool supports a few of the diagrams of the UML/SysML notably Use-case diagrams and Block definition diagrams. This is to provide direct traceability throughout the V-cycle. In future we look forward to implement and integrate all the structural and behavior diagrams of UML/SysML in our tool, to make it more comprehensive and useful.

References

1. A.P. Sage and J.E. Armstrong. *Introduction to systems engineering*. Wiley series in systems engineering. Wiley, 2000.
2. J.L. Eveleens and C. Verhoef. The rise and fall of the chaos report figures. *Software, IEEE*, 27(1):30–36, 2010.
3. E. Hull, K. Jackson, and J. Dick. *Requirements engineering*. Springer London, 2011.
4. Edmond Tonnellier and Olivier Terrien. Rework: models and metrics. In *Complex Systems Design & Management*, pages 119–131. Springer, 2012.
5. A. Van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262, 2001.
6. S. Misra, V. Kumar, and U. Kumar. Goal-oriented or scenario-based requirements engineering technique - what should a practitioner select? In *Electrical and Computer Engineering, 2005. Canadian Conference on*, pages 2288–2292, 2005.
7. Steve Easterbrook, Eric Yu, Jorge Aranda, Yuntian Fan, Jennifer Horkoff, Marcel Leica, and Rifat Abdul Qadir. Do viewpoints lead to better conceptual models? an exploratory case study. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 199–208. IEEE, 2005.
8. A. Gross and J. Doerr. What you need is what you get!: The vision of view-based requirements specifications. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 171–180, 2012.
9. O. Gotel, J. Cleland-Huang, J.H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic. The grand challenge of traceability (v1. 0). *Software and Systems Traceability*, pages 343–409, 2012.
10. E.S.K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235, 1997.
11. Vera Maria Bejamim Werneck, Antonio de Padua Albuquerque Oliveira, and JCSdP Leite. Comparing gore frameworks: i-star and kaos. In *Workshop em Engenharia de Requisitos (WER 2009), Val Paraiso, Chile, 2009*.
12. G. Regev and A. Wegmann. Where do goals come from: the underlying principles of goal-oriented requirements engineering. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*, pages 353–362, 2005.
13. I.J. Jureta, A. Borgida, N.A. Ernst, and J. Mylopoulos. Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 115–124, 2010.

14. S. Liaskos, S.A. McIlraith, S. Sohrabi, and J. Mylopoulos. Integrating preferences into goal models for requirements engineering. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 135–144, 2010.
15. I.J. Jureta, S. Faulkner, and P. Schobbens. Justifying goal models. In *Requirements Engineering, 14th IEEE International Conference*, pages 119–128, 2006.
16. Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. *Information Systems*, 27(6):365 – 389, 2002.
17. Objectiver. <http://www.objectiver.com/>, June 2013.
18. S. Supakkul and L. Chung. The re-tools: A multi-notational requirements modeling toolkit. In *Requirements Engineering Conference (RE), 2012 20th IEEE International*, pages 333–334, 2012.
19. Martin Glinz. Very lightweight requirements modeling. In *18th IEEE International Requirements Engineering Conference*, pages 385–386, 2010.
20. V. Shukla and G. Auriol. Methodology for determining stakeholders’ criteria weights in systems engineering. Poster in CSDM 2013, Paris.
21. Requirements Working Group (RWP). *Guide for Writing Requirements*. INCOSE, 2012.

Concurrent System Engineering in Air Traffic Management: Steering the SESAR Program

Alfredo Gomez¹, Benoit Fonck¹, André Ayoun² and Gianni Inzerillo²

¹ SESAR Joint Undertaking
alfredo.gomez@sesarju.eu, benoit.fonck@sesarju.eu

² EADS SESAR Industrial Support
andre.ayoun@cassidian.com, gianni.inzerillo@cassidian.com

Abstract As a “System of Systems” (SoS), Air Traffic Management (ATM) in Europe will be improved by simultaneous and coordinated evolutions of its constituting systems. The SESAR Program aims at reaching an ambitious performance target (for 2020) by developing a large collection of “Operational Improvement Steps” (OIs). This development is achieved by more than 300 projects, themselves involving a number of partners working at their own site throughout Europe.

To face this challenge, the SJU supported by EADS through an “industrial support” contract, has organized the management of the contributing projects on some basic principles:

- The SESAR Program is **Performance-driven**: this principle gives priority to developments that demonstrate significant performance gains within Key Performance Areas (KPA)¹.
- The programs **monitors the maturity** progress of the constituting OIs on a maturity scale (V1, V2, V3 maturity levels introduced by the European Operational Concept Validation Methodology) and revisits their priority in accordance with their maturity status.
- The achievement of each maturity level corresponds to a phase, and the transition from a maturity level to the next is assessed on the basis of **maturity criteria**.
- Each maturity criterion shall be demonstrated through evidences, **relying on validation results**.
- Maturity criteria reflect the confidence that the Requirements attached to OIs will be met. In particular, the confidence in meeting the performance

¹ The Key Performance Areas are: Safety ; Security ; Capacity ; Cost effectiveness; Efficiency ; Environmental sustainability ; Flexibility ; Interoperability ; Participation ; Predictability; Access and Equity.

targets (considered as a particular category of requirements) is a key maturity criterion: therefore, the accuracy of the performance results (based on platform measurements) supports the estimation of the confidence that the performance target will be met. So, the **maturity progress** includes **de-risking the level of performance**.

- For each OIs, a **validation strategy** is defined to ensure that the proper validation activities are planned by the relevant projects and aligned with programme priorities.

The SESAR Research and Development methodology therefore implements a progressive de-risking / Validation approach, considering the performance gains and confidence to support the maturity assessment. This approach permits to efficiently drive the program on Performance, by re-allocating, when necessary, the resources to the most significantly promising performance gains.

To give an example, let us consider two key performance areas: safety (characterized by a number of near-collisions or runway incursions) and capacity (characterized for example by a number of flights in airspace volume or runway movements per hour). Some performance figures, exploring several operational scenarios can be early obtained by fast time simulations and Monte-Carlo analysis. Initial performance results may be sufficient to support trade-offs between performance features, and to feed cost-benefit analysis supporting a decision to proceed or not (or rather to increase/ decrease the priority). Conversely, real flights with representative equipment in all the systems that contribute to the considered OIs may be costly and not suited to explore the solution performance in all relevant scenarios (e.g. nominal and off-nominal situations). So, exercises with real flights in representative operational environment will be mostly suited to assess maturity areas remaining to be validated (such as human factors).

2 Introduction: the SESAR Program

The SESAR program is the technological part of the Single European Sky initiative. The current phase addresses the Research and Development (R&D) activities to define the Operational concept and technical solutions to meet the challenging performance targets for 2020:

- 27% increase in Europe's airspace capacity,
- 40% reduction in accident risk per flight hour despite an increase in air traffic,
- 2.8% reduction per flight in environmental impact (e.g. CO₂ emission),
- 6% reduction in cost per flight.

The SESAR program deals with a collection of pre-identified Operational Improvement steps (OIs) and corresponding Enablers (ENs) that need to be matured in two ways:

- refinement of their definition,
- verification and validation (V&V) aiming at increasing the confidence in their feasibility and ability to achieve the requirements, including allocated performance requirements.

The R&D activities are achieved by a high number of entities (Air National or International Service Providers and Industrials) that have their own methods, interests,

and program of work but share the common goal to integrate the validated improvements in their operational environment and products.

SESAR Features:

- more than 300 projects working in parallel on around 40 Operation Focus Areas
- 3 steps (2013 – 2015- 2017) planned,
- 200 Operational Improvement steps (OIs) already identified

3 Methodological considerations for a R&D program

The classical V-cycle (waterfall) is a valid reference to conduct the proper development and validation of the concepts and solutions. The V-cycle is used as a reference to harmonize (or internally standardize) the development and validation activities and documentation.

Two methodological considerations, meaningful in any System of Systems R&D programme, are discussed hereafter:

- Top-down versus bottom-up design approach,
- "incremental" and "spiral" development methods.

3.1 Top-down versus bottom-up in a R&D program

In the commonly accepted meaning, top-down development refers to the derivation of high-level (user) requirements down to lower level (system / component levels). In the SESAR case, top-down here means that the driver is the performance target.

Bottom-up here reflects the fact that some Operational concepts or Operational Improvement steps and Technological evolutions (of Technical Enablers) are defined and developed by the experts as a result of local needs rather than by a direct derivation of higher level requirements.

In a R&D program, the solutions are often proposed spontaneously by the experts and even their refinement results from the deepening of emerging ideas rather than from a mere problem-solution development.

So this apparent contradiction can be resolved by applying a "selection" process, based on the joint assessment of maturity and performance. If a solution, based on the validation results, is not promising enough in terms of contribution to the global performance targets it could be rejected in favour of a more promising improvement.

3.2 Incremental versus spiral development

In a classical development, resource and risk management lead to develop successive increments. In a R&D program, it is preferable to take into account the results of a validation stage before deciding investment to further develop / mature the considered operational improvement.

The two approaches: incremental and evolutionary or spiral are briefly compared hereafter (reference [1]² can be considered for the definition of these terms).

3.2.1 Incremental development

The incremental build model is a method of development where the solution is designed, implemented and tested incrementally. The product is defined as finished when it satisfies all of its requirements. This allows partial utilization of product and avoids a long development time. This incremental implementation support stakeholders confidence, as incremental improvements progressively introduce partial capabilities.

In the SESAR program, 3 steps have been predefined with corresponding sets of OIs . Their development and validation are planned over several years in high-level roadmaps (release strategy and Validation roadmaps). In a sense, the SESAR program is basically incremental, where "block builds" correspond to the pre-planned content of the 3 steps.

3.2.2 Spiral development

The spiral development model process combines advantages of both top-down and bottom-up approaches. It combines the features of the prototyping and the waterfall model. The spiral model is suited to large, expensive and complicated systems.

In practice, in the SESAR program, the full set of OIs and Enablers was not fully and precisely defined at the beginning. Due to the R&D nature of the Programme, most of them need to be refined or modified according to the results of the ongoing experiments and development activities, supported by prototyping.

² Reference [1] defines Evolutionary in the following way: "Plan, specify, and implement an initial system capability. Gain experience with the initial system and define the next iteration to fix problems and extend capabilities. Refine the Concept of Operations, add and change system requirements, and revise the design as necessary. Continue with successive iterative refinements until the system is complete. This strategy can be shown as a series of "Vs" that are placed end to end since system operation on the right side of the "V" influences the next iteration. ...For particularly complex projects, a *spiral model* may be used, which is an evolutionary approach that is driven by risk management and extensive planning in each iteration. In the spiral model, the initial iterations include prototyping, analyses, and studies that are intended to reduce risk prior to implementation of an operational capability. The products in each iteration are defined to reduce risk as the system's degree of definition and implementation is increased incrementally."

4 Key SESAR System Engineering Management features

4.1 Discrete Operational Improvements steps

The Operational Improvement Steps are the smallest elements of the Operational concept. They have initially been defined during the Definition phase of SESAR and are permanently refined during campaigns. Their implementation into the real ATM system has been planned with Initial Operational Capability (IOC) dates set assuming an initial maturity.

These Operational improvements rely on several Enablers (ENs), including in particular the System Enablers based on technological development.

OIs having strong dependencies and contributing to the solution of the same problem may be grouped into a “SESAR Solution” to be jointly validated. For the sake of simplicity we consider in the sequel that SESAR solutions are OIs.

At the end of the operational concept development activity, all Operational Improvement steps are characterized by operational and performance requirements and all related System Enablers are characterized by technical requirements. In most cases, the performance requirements are initially set in a qualitative way and are more precisely defined during the maturation process.

4.2 Development and validation stages in SESAR

With reference to the classical V-cycle, the development and validation activities of OI steps follow the generic stream:

- Concept definition, Definition of Operational Requirements, along with their Safety and Performance Requirements, Operational Concept development and, simultaneously, Validation plan production,
- Development of System Requirements meeting the Operational Requirements (for all Systems contributing to the corresponding Operational Improvement) , and simultaneously production of the Verification plan,
- System solution development with prototypes and platform integration,
- Verification that each System satisfies its requirements,
- Validation of the operational concept and related performance.

Standard SESAR documentation has been defined to ensure the consistent development of requirements and validation objectives.

4.3 The SESAR performance target

The performance target addresses Key Performance Areas (KPAs), with corresponding Key Performance Indicators (KPIs) for the overarching ATM system of systems.

The political targets are split over each of the 3 program steps.

The KPIs are broken down in a number of Performance Indicators (PIs) with associated metrics. PIs are related to KPIs via modelling techniques called Benefit Mechanisms. PIs are measured during validation Exercises.

Managing the performance targets on PIs as requirements, allow linking the political target and project activity.

So the validation activities allow risk-reduction as regards performance. Indeed, the performance uncertainty decreases and confidence that the performance target will be met increases (as notionally represented in the figure below).

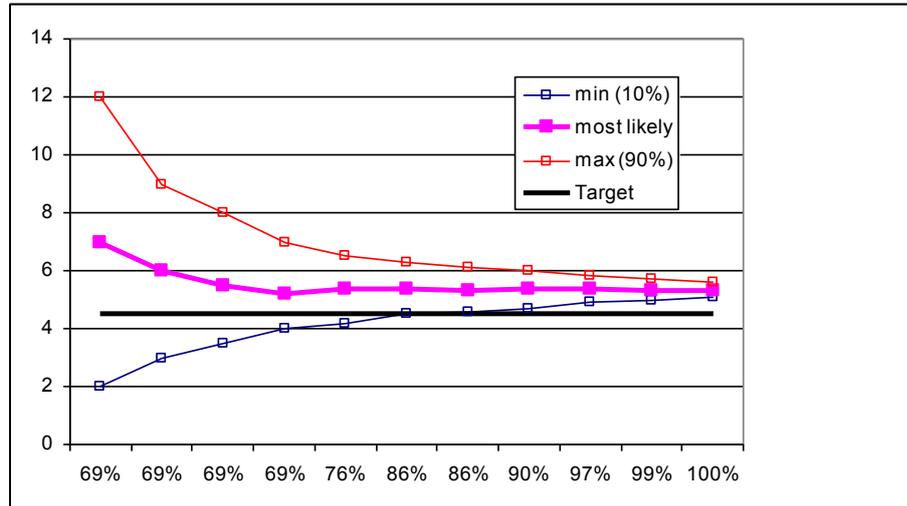


Figure 1: notional representation of performance uncertainty reduction along with validation activities

(In this case, the probability that target is met increases with time)

4.4 Maturation and program steering

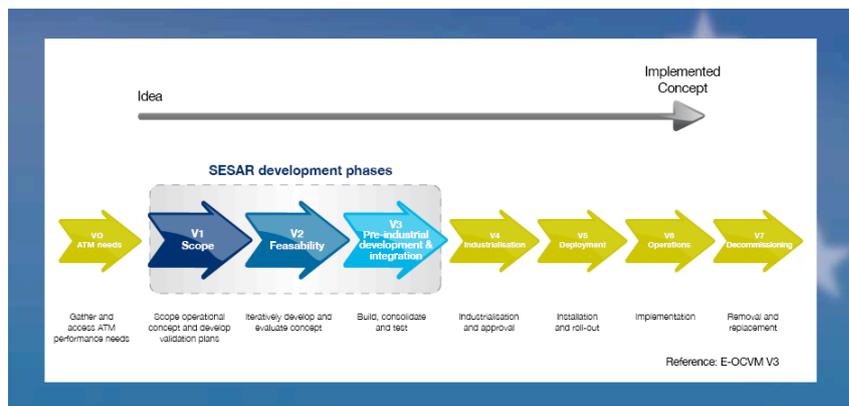


Figure 2: SESAR within the E-OCVM lifecycle

SESAR covers three phases of the E-OCVM lifecycle: V1, V2 and V3. These three phases correspond to 3 development iterations of the development process, ending to increasing level of maturity. The SJU has developed a set of criteria to address the 3 transitions: V1 to V2, V2 to V3, V3 to V4.

For each OIs, a validation strategy is defined to plan sequence of activities that end to a completely V3-mature delivery at a date compatible with the planned IOC date (generally, V3 should be completed at least 2 years before the IOC to let sufficient time for eventual V4 industrialization and certification activities). A "top-down Verification and Validation Roadmap" is regularly updated to refine the planning of validation activities in accordance with the validation strategy.

Every year, the V3 activities for the coming year are planned with a high level of detail to ensure the concepts will be fully V3-validated at the end of the next year: this "Release" approach intends to deliver each year a set of V3-validated OIs.

The management of a Release lays on 3 System Engineering reviews. At the last one, the actual maturity is assessed based on the provided evidence, including Validation-Exercise-Reports. If most V3-to-V4 criteria are satisfied, the corresponding OIs are "released" by the SESAR program.

Monitoring the maturity supports the decision to proceed to the next phase and to continue investing into solutions. This monitoring process takes place from initial V1 to V3 maturity level, with increased attention at the latest stages (especially in Release monitoring). Such a continuous monitoring supports decision to stop, redirect or reallocate resources towards the most beneficial OIs, with consideration of their time-horizon.

5 Conclusion: the SoS concurrent design challenge

The SESAR program addresses concurrent engineering of a large System of Systems. As such, the various developments of all its constituting elements need to be coordinated, taking benefit from both top-down approach and from the use of successive development and validation activities to improve the definition of the Operational concept elements.

Strict monitoring of maturity and steering the program based on the expected performance benefits ensure that the parallel developments and validation activities, achieved by 300 projects working in parallel, are properly synchronized and steered.

This has been permitted, within SESAR, by defining standard levels and standard maturity criteria and by imposing a pace with annual releases and synchronization points. Such an approach demonstrated its efficiency, since 2013 will see the 3rd Release of V3-validated sets of OIs, grouped into SESAR Solutions.

However, consolidating results and feeding back, to properly drive the program from the performance view, has demonstrated to be uneasy, and remains a challenge.

References

- [1] System Engineering for Intelligent Transportation Systems, US Dept of Transportation, 2007
- [2] Systems Engineering Handbook, a guide for system lifecycle processes and activities, International Council on Systems Engineering (INCOSE), version 3.1 August 2007.
- [3] Baldwin, K., June 28, 2007, "Systems of Systems: Challenges for Systems Engineering, INCOSE SoS SE Panel.

- [4] ISO/IEC 15288, 2002, Systems Engineering—System Life Cycle Processes.
- [5] Maier, M., 1998, "Architecting Principles for Systems-of-Systems," Systems Engineering, Vol. 1, No. 4, pp 267-284.
- [6] Dahmann, J. and K. Baldwin, April 7-10, 2008, "Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering," IEEE Systems Conference, Montreal, Canada.
- [7] Office of the Undersecretary of Defense for Acquisition, Technology and Logistics (OUSD AT&L), August 2008, Systems Engineering Guide for Systems of Systems, Washington, DC.

For more information on SESAR, visit www.sesarju.eu or contact communications@sesarju.eu.

Agile Stylized Approach to Manage Complex Project

Mikhail Belov

Deputy Chief Executive Officer, IBS, mbelov@ibs.ru

Abstract The paper is devoted to one of the practical approaches to manage complex projects. A case study of complex project management is described. This concerns transforming a Russian IT-company into an enterprise system and system of systems (ES/SoS) at the same time. Main elements of the transformation – the goal, solution, structure, and relations among constituents of ES/SoS, ES/SoS engineering process, and results – are represented. The analysis of the project –environment, constraints, risks, opportunities, and uncertainty – is considered. It is found that it is exactly uncertainty that causes grave problems in complex project management and that is responsible for the main challenges facing the governing body. An “agile stylized approach” to complex project management which was successfully used in the case study is described. The applicability of this approach to manage other complex projects and further research prospects are discussed.

1. Introduction

The traditional project management domain has been very well developed during the last several decades. The PMBOK@Guide [1] and other handbooks and documents of this area provide good theoretical and practical information background to initiate a project, to schedule project activities, to plan and allocate resources, to monitor work, to manage risks, etc. The traditional approach works perfectly in huge numbers of projects in practically any area of business or social life, thereby demonstrating its applicability and efficiency.

But the absolute insufficiency of this approach has shown up in some cases – in complex projects, those involving enterprises or other complex systems as well as projects dealing with new product development. Global business environment sophistication (enterprises, products, services, their interrelations, etc.) causes the complexity of the projects, so the complex project management theme has become more and more topical in recent years [2].

2. Exemplary complex project

The exemplary project [4] represents the transformation of complex ES/SoS encompassing different types of constituents, interlinked by different and sophisticated relationships, with “soft” and variable boundaries and complex ES/SoS engineering processes. Together these characteristics give rise to really “wicked” problems [5] and make the project truly complex.

In 2001 the top management of the IBS company initiated a fundamental transformation to change the company’s strategy and business model. The company was one of the biggest Russian IT systems integrator at that time, with about 900 employees. Annual revenues of around \$80M were mainly generated by IT infrastructure projects (complex computing systems, multiservice networks, etc.), hardware and software distribution.

IBS management forecasted considerable growth of the Russian IT services and consulting market based on the fast growing Russian economy . The economy was rapidly recovering from the national financial crisis of 1998. The largest corporations started overseas expansion and borrowed from international markets to finance this growth. IBS predicted corresponding growth in the complexity of business processes and their associated software and hardware systems all of which should require more consulting and IT services.

Based on this forecast, IBS established a strategy goal to double the share (in annual revenue) of IT services and consulting from 25% to 50% over one year. Further growth in this business was planned as a long term trend. The consulting and IT services business is very complex technologically and organizationally and dramatically differs from IBS’s former infrastructure focus. Thus, a fundamental transformation was required, and it was executed during 2002.

To achieve this strategy goal, the company’s management defined new capabilities required to sell and execute large, complex, and multi-disciplinary consulting and services projects. They thought sales and execution processes should be treated as absolutely standardized, regular, and routine procedures. Accordingly they defined five major groups of focused capabilities:

1. Deliver consulting and services.
2. Sell complex consulting projects.
3. Execute and deliver complex multi-discipline consulting projects as very effective and highly standardized processes.
4. Manage human resources effectively. (Highly skilled and experienced employees are the key performing engine of the consulting business.)
5. Measure and account for projects’, business units’ (BUs’) and employees’ performance. (The target business model is very dynamic, so on-line measurement and forecasting of key performance indicators is critically important.)

The IBS structure consists of a corporate governing center (CGC) and autonomous business units (BUs), figure 1.

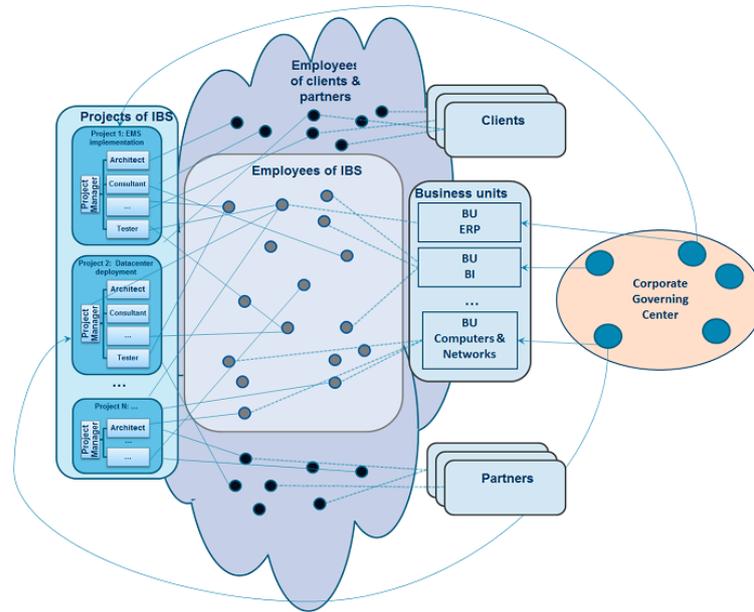


Fig. 1. Key business-agents and the relationships among them.

Different BUs execute the sales function, deliver the products and services through project execution, and provide back-office support for other BUs. In reality BUs serve as the resources pool that form project teams. The same employee may play different roles in different projects. For example, a BU head is often assigned as director of one project and an architect in another project. The BUs are independent to each other. There is no direct relationship between any two BUs (constituents); they are linked via their employees' participation in joint project teams. The CGC consists of the Chief Executive Officer (CEO) and his deputies who run the company; they supervise and coordinate BUs' activities rather than controlling them by directives.

The IBS business structure is more than just "BUs + CGC". Projects, project teams, and employees also play considerable roles. Employees as well as project teams and BUs are key business-agents of the company.

Almost always complex consulting projects are executed by joint teams which include employees of the consulting company (IBS), as well as its partners and customers. Thus the company forms an "extended enterprise" by involving employees of other firms in projects, and the transformation scope covered both IBS and its extended enterprise. Joint project teams are temporal objects (team lifetime equals project duration); so the boundaries of ES/SoS transformed are variable and temporal. The extended enterprise significantly differs from IBS company: the average supplement of "virtual" constituents is around twice or more (estimation of [4]).

In general the relationships inside an extended enterprise form a very complex network like that depicted in Fig. 1 which connects BUs, projects, and employees by different links (administrative or operational; project management; supervising; etc.).

The relationships are realized in practice by means of integrated processes of project management and accounting covering joint project teams formed of employees of different firms. The CGC is not the top root of a control hierarchy; rather it is at the center of a “star” of autonomous BUs.

A systems engineering (SE) task was established to develop the required capabilities for IBS to become an ES/SoS company. The SE process of the transformation consists of:

1. Mission analysis. In the initial analysis the mission was translated to capabilities. The transformation team found that capabilities might not be directly translated to any business-agent: neither BUs (resource pools), nor projects (temporal elements), nor employees might realize necessary capabilities.
2. Key areas definition. Realizing 1 above the transformation team defined several key areas of company’s operations which were supposed to be changed to form new capabilities. Five key areas of operations or activities were defined: core technologies area – consulting and services area; project implementation area; BU growth area (hiring and newcomer integration); motivation of the employees area; and management accounting area.
3. New capabilities support systems development with pilot implementations and roll-outs (for each key area). In each of the areas the engineering and implementation of the systems to support new capabilities was planned. The support systems (procedures, guides, documents, and software systems) were implemented initially in pilot zones and later rolled out to the full extended enterprise. Processes and rules were developed as “end-to-end” and “crossing” ones to integrate all BUs, project teams, and employees.
4. Operational performance assessment.

As is now seen the Russian IT services and consulting market grew by more than a factor of five during 2001-2010, and IBS has been leading this growth, getting the main market share for the years 2009-2012.

- The mission was accomplished: new capabilities of BUs and the company as the whole were formed in areas of sales and delivery of complex projects; the business model and company strategy dramatically changed.
- Specially developed auxiliary and supporting systems serve as the tools to support new capabilities; the systems formed exactly the corporate infrastructure of new business model.
- The Extended enterprise was formed around the company through integration of employees of clients and partners into project teams.

3. Analysis

The main factors (and their origins) which caused wicked problems for the project governing body and project team in an exemplar project are studied and analyzed in this section.

Any project (or even any kind of human activity) might be characterized by the following shortages, limitations, or uncertainties which are absolutely universal ones:

- Recourses (finance, material, human, etc.);
- Time (schedule limits);
- Knowledge/information.

The importance and influence of each of these factors is best illustrated with a concrete case. Resource and time restrictions play key roles in repeatable or typical activities; here uncertainty is insignificant and might even be ignored. On the contrary in the exemplar transformation project as well as in other complex projects uncertainty is the most influencing factor; it should be considered very seriously in project management activities.

External factors affecting complex projects

The IBS company, as an ES/SoS, operates in a very sophisticated external environment that combines policy and law, economy, society and culture, technology, and even nature. ES/SoS's elements and environment interact on several different levels: IBS as the whole; BUs; projects; and employees. And not only does the environment affect ES/SoS but also ES/SoS influences the environment.

Besides generic external factors mentioned above national and industrial factors affected the transformation:

- The Worldwide Internet boom created very high business expectations of IT sector growth and attracted investors; also new technologies appeared very rapidly which expanded the IT market very fast. These factors forced IT companies (IBS as well) to spent resources to try to capture a niche in the growing market. For example, IBS management at that time established several BUs (customer relationship management systems, internet technologies, etc.), and some of them were reformed or closed later.
- The after crisis factor (after national crisis of 1998) – fast economic recovery, devaluated currency, and international expansion of Russian firms, on the one hand, initiated and pushed the transformation, and on the other hand, was embarrassing due to the fast growing labor cost which is the main cost component of a consulting and IT firm.
- The post-Soviet legacy played a considerable role in society (even now). Destruction of entrepreneurship during the Soviet period dramatically lowered business activities for the majority of employees, and this complicated necessary changes.

Specific corporate constraints and challenge

During the transformation period IBS management faced very specific corporate constraints.

The lack of experience in ES/SoS transformation, engineering, and the running of a consulting and IT services company (even the lack of any textbooks or guides in

those areas) were the major challenges which IBS management faced. The task to be solved did not impact organizational changes (a well-developed and described area) but did belong to ES/SoS engineering. In spite of IBS's lack of the experience it was decided to prepare and execute the transformation based on the companies' employees without external consultants' involvement. The following arguments supported the decision:

- The task to be solved was not typical, so there was no widely used and well tested algorithm, and there were few consultants exactly experienced in such things. So only consultants with similar experience (strategy consulting and organizational management) might be hired.
- In 2001-2002 the Russian consulting industry was not developed, so Russian consultants with appropriate experience might not be hired at all. Only foreign professionals were available but they would have needed first of all to study Russian economic specifics. Such study, naturally, would have increased time and cost of the transformation.
- Also, it was evident, that a joint team of IBS management and employees would have to be formed.; management would have to be interviewed and involved in decision making; and all employees would have to participate in change implementations.
- External consultants are "external people"; they are not stakeholders; so their level of interest in success might not be very high, and their output also might not be outstanding.
- Unwillingness to open professional secrets to direct competitors and other intellectual property issues limited external consultants hiring.

The final decision was made based on the comparison of pros and cons: to execute the transformation without involvement of external consulting resources. A special back-office unit (BoU) responsible for business processes development was established, and an "agile-stylized" program management approach was applied to take the challenges, pursue opportunities, and to mitigate risks.

Another challenge dealt with the transformation objective: a very high complexity IBS as an ES/SoS. Management recognized that the company was very complex, with a lot different agents, constituents, and inter-relationships, and that ES/SoS might become even more complex after the transformation. This complexification happened due to the company becoming an extended enterprise, with governing hierarchies weakening, and relationships increasing in sophistication.

One more business challenge was the risk of a mistaken forecast of IT market development: expected growth of the consulting and services market might have not happened, and in this case the transformation would have been senseless, this challenge generated additional emotional stress for management.

These specific corporate constraints engendered the shortage of knowledge – the uncertainty in the project, which created main challenges for the governing body. Time and resource limitations also existed: as with requirements to move as soon as

possible, do not interrupt on-going business, and avoid external expenses, but time/resources restrictions did not play key role.

Sources of uncertainty, opportunities and risks

The ES/SoS opportunities were attractive but the lack of experience and knowledge were very serious concerns that induced considerable risks that needed to be managed. The ambiguity of the exemplar transformation might be expressed by whether:

- The implementation of auxiliary supporting systems and the completion of project activities would really create required capabilities;
- New capabilities would ensure the capturing of a considerable consulting market share;
- The prediction of dramatic growth of consulting and IT services demand would be correct.

To synthesize these issues we may conclude, that the main opportunities and risks were whether the:

1. System (ES/SoS) of interest being created (its architecture, design, properties, etc.) would get the required, functions, capabilities, etc.;
2. Technologies and approaches would be appropriate to create the system of interest;
3. System would be efficient in interaction with the environment;
4. Prediction of the status of the environment would turn to be correct.

The first and second aspects above are biased or specific ones – both of them might be avoided (theoretically, at least) by hiring external consultants. But the third and fourth aspects are inevitable ones – nobody knows the future.

Further, the fourth aspect relates to the lack of knowledge about the environment and its variability that engenders changeability of the requirements. This is a very natural characteristic of the complex projects focusing on ES/SoS development, transformation, or modernization.

Analytical summary

The following conclusions based on the analysis of the exemplar complex project are more or less common for the whole complex project domain:

- Generic, industrial, and national external factors intensify time and resources shortages and also reinforce uncertainty.
- Specific project constraints mainly (and very heavily) engender the shortage of knowledge – the uncertainty in the project.
- Rampant uncertainty causes wicked problems in complex project management and creates main challenges for the governing body. The four fundamental opportunities and risks listed above exemplify the core uncertainties of complex projects. All deal with the system of interest, but not with the project activities.

4. Agile stylized approach

The transformation team developed and used an approach which is very similar to the Agile Development approach of [3] to address the uncertainties.

The PMI defines a project as a “temporary group activity designed to produce a unique product, service or result” [6]. Previous analysis demonstrates, that main risks might not be studied and explained based only on “activities”, there are other entities which cause main risks and which should be seriously considered in complex project management scope. The traditional approach is appropriate to manage the time and resources constraints, but it is not focused on the system of interest’s uncertainty and the shortage of knowledge. So traditional SE’s usability in complex domain is quite limited. Fig. 2 represents core entities of a complex project domain.

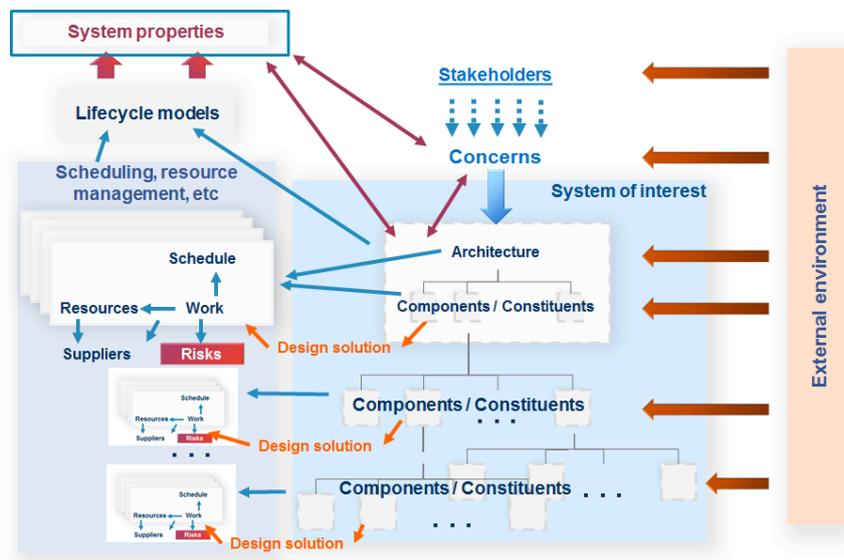


Fig. 2 The domain of complex project management.

The left-hand portion of the figure represents the traditional project management area – which is activities based: work, resources, teams, etc. All these entities are well defined and described in traditional project management documents like PMBOK [1].

The right-hand portion of the figure demonstrates the system of interest and external environment, which really cause the uncertainties described above.

The system (ES/SoS) of interest is represented by the system’s architecture and the components (or constituents) which might be also complex entities or systems. The external environment affects the complex project by the following threads:

1. Direct influence on the system of interest;

2. Influence on the efficiency of the system operating in the environment;
3. Changes of the environment which causes changes of the requirement to the system of interest.

In this manner the left part corresponds to the resources and time constraints, and right part, to the uncertainty constraint; both parts cover all complex constraints and better (than traditional approach) represent complex project domain.

The transformation was, in reality, a quite compact program of projects. The project teams consisted of company's employees, and the program was governed by the CGC according to a single program plan. The transformation task did not look complicated from a project or program management point of view. Organizational change implementation was a quite standard activity with a well-known personnel resistance problem and tested approaches to overcome this resistance. Deep involvement of top management, headed by the CEO, guaranteed effective project management and execution as well as organizational management implementation.

The set of activities in all previously described key areas had to be executed to effect the transformation. All those activities were conducted in parallel to save time and resources. Integration and interoperability of the new capabilities' support systems (developed in key areas) required a thorough integration of parallel development tasks. So joint workgroups of employees were formed at levels below the officers. CGC played the role of integrating the workgroups at the management level. In effect, a multi-level integrated workgroup was formed.

The major complexities and/or problems derived from the four uncertainties described in Section 3. These uncertainties could not be controlled by means of additional research and study – the ES/SoS team did not have time for that. Experiments and tests also would not help – there was no testing or training area to check solutions before implementation: all solutions were piloted within a working company.

The quick and effective creation of solutions and their practical testing is a very natural and rational approach to manage such uncertainties: the main idea was to accelerate the circle or loop, “define requirement–design–implement–validate”, when there is no other way. Initial conditions and the approach mentioned led to plenty of changes in the implementation process, so it's necessary to manage them fast and effectively.

Based on the understanding of insufficiency traditional activity-based project management, the management team formed a “project kernel” including the description of core elements of the left- and right-hand portion of Fig. 2: <work and resource plan> and <architecture and core component solutions>.

Such a project kernel covers the whole complex project domain and enables the management of uncertainties. Both elements of the kernel were used as an aggregate: not only the plan but also the architecture description was used to monitor the progress, to make changes, etc.

The following principles were used to manage the portfolio of projects in case of lack of experience and ready-to-use algorithms and methods:

- Form solution as fast as possible (without regard to pure quality) and quickly test it in practice;
- Failures are unavoidable, perceive them easily and react rationally;
- In case of failure analyze the situation, find a new solution, generate changes, and update the plan;
- Work in parallel, verifying and coordinating intermediate results;
- The schedule might be corrected and updated but should not be violated due to improper execution;
- Formulate and test the most critical and questionable solutions first;
- Start with pilots and then (if they seem to be working) roll them out to cover the whole scope;
- Use high level and high qualified management to control the piloting a developed solution (but not additional aspects) to avoid waste of the resources.

Following those principles, a quite strong executing discipline, a high level of the sponsorship, and the involvement of all employees enabled the transformation to be completed in time and without hiring consultants while keeping and developing on-going business.

5. Conclusion

The agile stylized approach represented in the paper made it possible to complete complex project in quite short time period; complicated transformation of ES/SoS was kept under full control during the execution period. And very importantly, this transformation had good business results – the IBS company played the leading role in the Russian consulting market during 2009-2012.

The analysis showed that the majority of properties or characteristics of the exemplar project are quite common to other complex projects dealing with ES/SoS. So the main ideas of the approach (the information project kernel and the acceleration of the loop “define requirement–design–implement–validate”) might be recommended for practical usage in any complex project domain.

Further research might be focused on the development of the agile complex project methodologies or frameworks including models of project processes, manuals and guidebooks, reporting templates, etc.

References

1. PMI (2008) A Guide to the Project Management Body of Knowledge PMBOK® Guide. 4th Edition, Project Management Institute, USA.
2. Ireland V., Gorod A., White B. E., Gandhi S. J., Sauser B. (2013) A Contribution to Developing a Complex Project Management BOK. In Project Perspectives. The annual publication of International Project Management Association
3. INCOSE (2010) Systems Engineering Handbook v. 3.2 INCOSE, January 2010

Agile stylized approach to manage complex project

4. Belov M. (2013) IBS Group, Eastern European IT Services: Capability-Based Development for Business Transformation. In Case Studies in System of Systems, Enterprises, and Complex Systems Engineering Gorod A., White B. E. Ireland V., Gandhi S. J., Sauser B. (eds.) New York, NY: CRC Press, Taylor & Francis, New York (scheduled for Fall, 2013)
5. Conklin J. Chapter 1 of Dialogue Mapping: Building Shared Understanding of Wicked Problems. (2005) "Wicked Problems and Social Complexity," CogNexus Institute, http://cognexus.org/wpf/wicked_problems.pdf (accessed 15 February 2013) t
6. PMI (2013) What is Project Management? <http://www.pmi.org/About-Us/About-Us-What-is-Project-Management.aspx> Accessed 5 April 2013.

A Model-Based Assessment for the Solution Space of a Cognitive Coffee Robot Waiter

Thierry Sop Njindam, Torsten Metzler,
Udo Lindemann and Kristin Paetzold

Abstract Cognitive products are tangible and durable things with cognitive capabilities that meet and exceed user expectations by using cognitive functions, e.g. to perceive, to learn, to reason, etc., to reduce the need for human input. This paper presents a model-based assessment of the solution space for cognitive products. So far, the design of cognitive products has been based on prototype-oriented approaches, which mainly focus on cognitive algorithms, relying too much on designer's experience, beliefs or ad hoc arbitrated processes and following as a consequence the "design it now and fix it later!"-philosophy. A model-based assessment of the solution space would enable a better and early estimation of design alternatives that meet not only software requirements but also hardware requirements from the very early stages down to system structural and behavioural aspects in highly dynamic and uncertain environments. The conventional MBSE approach has been adapted to cognitive products and is demonstrated using a cognitive coffee robot waiter.

1. Introduction

Cognitive products are tangible and durable things with cognitive capabilities such as perceiving the environment, learning and reasoning from knowledge models that are created through tight integration between a physical carrier system with embodied mechanics, electronics, microprocessors and advanced software algorithms [8]. A typical cognitive product basically perceives its environment as well as the actions performed by the user with whom it interacts through its embedded sensors, then stores acquired information in its knowledge base, reorganizes and enlarges its prior knowledge and skills through learning and then plans its actions either on the basis of processes and sequences of operations stored in its knowledge base or from logical reasoning mechanisms.

The design of cognitive products requires a collaborative effort between engineering sciences, information processing, cognitive and life sciences and artificial intelligence. A holistic view of how the entire system fits together is required with regards to the number and diversity of interconnected elements, the tight integration between hardware and software elements, the close interaction with the surrounding environ-

ment and the cognitive behavior over time. To date, there is no holistic approach to support the development process of cognitive products. From the engineering design point of view, systematic approaches (VDI 2221; VDI 2206; Axiomatic Design; Gero's FBS-Model), even though they provide fundamental aspects of the design as a problem solving activity from the conceptual design and embodiment design to detail design, have several shortcomings since they do not adequately consider the system as a whole as well as the various involved disciplines (information processing, cognitive sciences, etc.) and refer to disconnected simulation models in different design stages.

With regards to the development of cognitive products, traditional long-lasting prototype-oriented approaches with disintegrated hardware and software processes are highly iterative, inefficient, time consuming, error-prone and do not fully comprehend the system under consideration, especially during the early design phases.

The goal of this contribution is to improve the design process of cognitive products and provide a generic model-based approach by addressing the following problems:

- Incoherent and non-holistic representation of the system with its cognitive functions, especially during the early design phases.
- Insufficient traceability between core aspects of cognitive products such as the flexibility of their requirements, functions including cognitive functions, structure, behavior, performance and operational scenarios processed during their lifetime.
- Arbitrary, experience-based or a priori selection of design parameters without analysis and evaluation of system requirements, design options, uncertainties during the product lifecycle, etc.
- Limited re-use of specifications, system models, and design artifacts to support the development of complex embedded systems such as cognitive products.

The analysis and visualization of the solution space in the design process of cognitive products will support decisions to be made in the selection of system design parameters. A cognitive coffee robot waiter is used as an illustrative example.

Section 2 introduces cognitive products and how they are modeled from a functional perspective. Section 3 describes a model-based systems engineering approach to assess the solution space of systems in general. This approach is then applied to partially assess the solution space of the coffee robot waiter in Section 4. Section 5 discusses the results and section 6 concludes this contribution.

2. Cognitive Products

Cognitive products are tangible consist of a physical carrier system with embodied mechanics, electronics, microprocessors and software. The surplus value is created through cognitive functions enabled by flexible control loops and cognitive algorithms, e. g. stemming from AI. Cognitive functions, like *to perceive*, *to learn*, *to reason*, etc., allow cognitive products to act in an increasingly intelligent and human-like manner. They can adapt to dynamic environments as well as to the changing product state and can be integrated in human living environments easily. They interact and cooperate with humans, have a better performance than non-cognitive products and are able to maintain multiple goals and make appropriate decisions and thus exceed current user expectations [8, 11].

To support the interdisciplinary development of cognitive products a taxonomy of cognitive functions and flows is presented in [9]. The taxonomy enables and fosters a model-based development of formal functional models in the conceptual design phase of cognitive products. Functional architectures, combining a functional model with a structural model, make the reuse of the allocation from function to structure possible as well as the identification of patterns. Another method, addressing how cognitive functions can be identified in activity diagrams and integrated in cognitive product concepts, has been published in [10].

3. Model-Based Assessment of Solution Space in the early Design Phase

This section describes a general model-based systems engineering (MBSE) assessment of the solution space using systems engineering and extends it for cognitive products by including the flexibility needed to handle the cognitive behaviour. Generally, the earlier a new technology is adopted in complex systems development, the more likely it is to create an inconsistent and error-prone design, at least before adequate design methodologies are developed. Model-based systems engineering has been widely recognized as an effective means to manage the complexity of systems by using descriptive and simulation models to support the specification, design, analysis and verification of systems consisting of both hardware and software components [4]. The conventional framework as depicted in fig. 1 has been adapted to emphasize cognitive functions as well as environmental conditions among the core characteristics of cognitive products. This top-down approach maintains consistency between the system views and activities within the design process such as requirements specification, functional analysis, functional-structural allocation, architecture definition, evaluation and optimization of design alternatives, verification and validation. New challenges faced in designing cognitive products emerge on the one side from the flexibility of requirements and related operational scenarios in a cognitive context and on the other side from unpredictable and dynamic environmental conditions

The adapted MBSE-Workflow begins with the well-known typical early design tasks which focus on the identification of user needs as a basis for the technical requirements specification. This stage defines and at the same time consequently constrains the design space [6].

Next, the identified user needs and system requirements are turned into functions. Functions are a solution-neutral description of what the system does and can be represented conveniently in blocks with interfaces between them. The emphasis in generating functional architectures of cognitive products is placed on the identification of flows of information and energy among cognitive functions and between cognitive functions and other non-cognitive functions. A deep understanding of the interactions between the system and its surrounding dynamic environment, by means of inputs and outputs, is crucial to determine the system boundary [12]. It is usually necessary for complex systems, to decompose their primary functions into sub-functions. This increases the level of detail of the model and provides a good overview about the flows (information, energy or matter) on which the functions operate [3]. As a result, this functional model, on the one side, provides a link between the system's specifications and the subsequent physical embodiment. The resulting functional model is an ab-

abstract and static view of “what the system should do” and illustrates the internal relations between the functions.

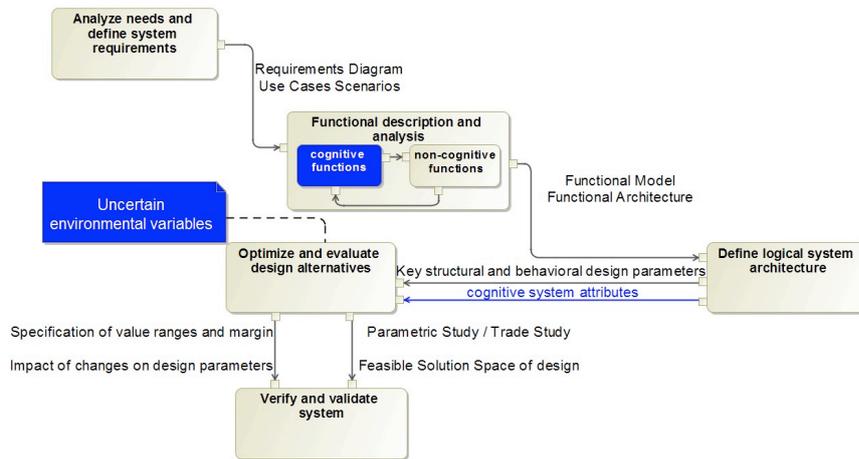


Fig. 1 Activities within the MBSE Process, according to [1] and [12]

On the other side, functional models strategically guide further allocation of system functions to physical components even though there exists no direct or objective mapping from functional elements to physical elements [5], [15]. This implies that more than one design may ensue from the mapping between the functional domain and the physical domain. Defining the system architecture, which further reduces the solution space of design, includes the specification of structural design parameters such as geometric attributes of parts and physical relationships between the parts. However, the cognitive system behaviour is implemented to a great extent in the system software elements. Even though related cognitive system attributes can not be estimated at these design stages, it is crucial to set critical system parameters, limit values and boundary conditions within which the cognitive system behaviour is assumed to be performed. The context-dependent solution space of the design is then tremendously influenced by these cognitive system variables and is the result of the optimization of possible design alternatives in a well-defined context with a wide range of possible scenarios. It is assessed by trading off structural and performance design parameters, based on equations of the system dynamics and technical constraints with regard to previously defined performance requirements, and by coupling them with environmental variables and cognitive system attributes. A relatively high level of imprecision of the environmental and system design parameters is assumed in the early design phase. Several methods such as fuzzy arithmetic, interval mathematics or probability-box have been introduced to cope with such uncertainties and imprecision issues to estimate value ranges and margin of design parameters [2],[13]. The final verification and validation activities are intended to make sure that the selected design alternatives satisfy the previously defined system requirements in the specified context.

4. Assessing the Solution Space of a Cognitive Product using an Application Example

In this section is shown how the assessment of the solution space in cognitive product development is accomplished using a coffee robot waiter as an example. The coffee robot waiter (see fig.2 left) is a cognitive product serving coffee autonomously in a known environment [9]. It was developed by students and assistants with the goal to implement and test cognitive functions in a physical product. The robot is able to serve coffee based on orders placed on a website. This is possible because the robot knows its working environment that it learned prior to the use-case when serving coffee. If more than one order is placed at the same time it calculates the optimal route according to an online traveling salesman algorithm which depicts some aspects of the cognitive system behavior by planning the delivery route and then moves to the target positions (compare tours in fig. 2 right). In addition, it checks if enough coffee and energy is available to satisfy user requests. On its way it avoids static and dynamic obstacles. It remembers reoccurring obstacles at certain locations and adds them to the map to consider them in the next path calculation. Based on the robot's experience, it estimates the time till coffee is delivered for every target and sends a message to the user screen.

The focus of this paper regarding the assessment of the solution space of the coffee robot waiter is on the top-level functional requirement "cognition" with its derived sub-requirements "autonomy", as illustrated in fig. 3. Other sub-requirements are not relevant for this work. The objective herein is limited to the specifications of "WHAT" the system should do in terms of its cognitive functions. Given this problem and assumed requirements specifications, we identify the following core cognitive functions:

- Perceive working environment
- Learn working environment
- Decide best route
- Think about orders
- Act in environment

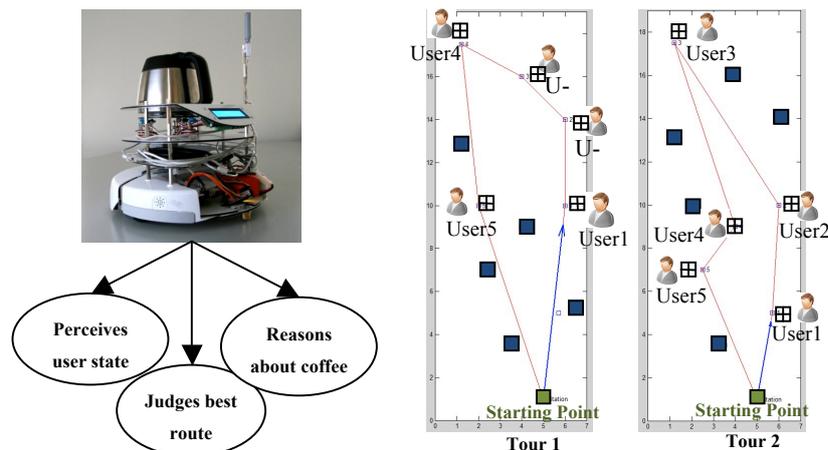


Fig. 2 Overview of the cognitive coffee robot waiter

Next, a functional architecture, as an essential element of the conceptual design of the system is developed and serves as basis for the derivation of the system architecture. In the application example the Systems Modeling Language (SysML) in combination with the taxonomy of cognitive functions and flows is used. Cole Jr. underlines in [3] the importance of this integrated functional view in the design process even though things are fuzzy at this stage of the design process. Hierarchical functional identification diagrams and functional flow diagram are typical diagrams belonging to a front-end functional analysis.

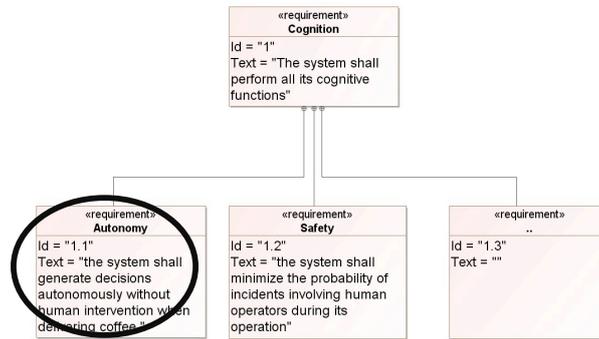


Fig. 3 Top-level requirements of the cognitive coffee robot waiter

4.1 Defining the system cognitive functions with the functional identification diagram

Functional analysis, as viewed in the MBSE process (fig. 1), includes a top-down view, from the highest to the lowest abstraction level which is usually required to hierarchically decompose high level functions into sub-functions and illustrated by functional identification diagrams. Fig. 4 illustrates the function hierarchy of the coffee robot waiter.

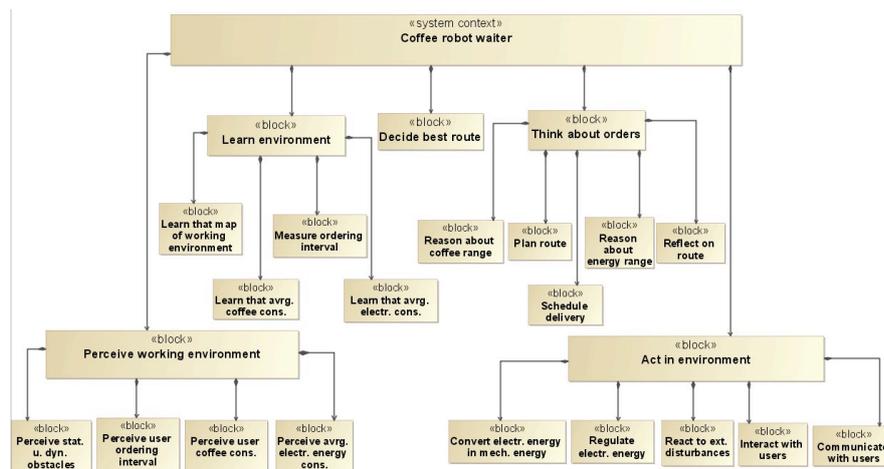


Fig. 4 Functional identification diagram of the coffee robot waiter

4.2 Representing the system functional model with functional flow diagram

As functions are more detailed, functional flow diagrams show the linkage between these functions and also provide valuable information on the arrangement of functional elements, their sequence of actions and the interaction amongst the system functions. The lines connecting the functions illustrate the functional flows by means of information, data and energy flows. Figure 5 illustrates the functional flows of the coffee robot waiter.

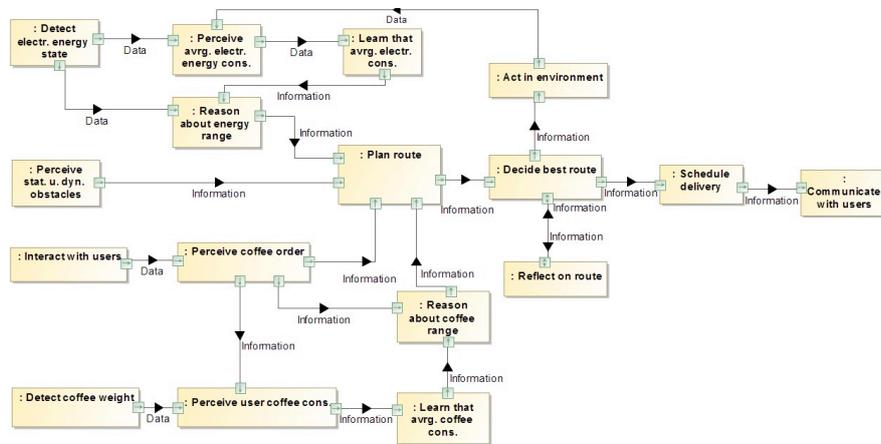


Fig.5: Functional Flow Diagram of the coffee robot waiter

4.3 Allocating functional to structural parts

Allocating functional elements to structural elements is a common aspect in the design process called system architecture which provides an overview about the concrete relation between cognitive functions and the structural elements they need to be realized in the physical world. The complexity of cognitive products is reflected in this stage with the number and diversity of interrelated system elements. Linking cognitive functions with physical elements is basically essential to identify the necessary hardware modules and generate the system physical architecture. An example of the functional-structural allocation for the cognitive function “Act in environment” is illustrated in fig. 6. The complete functional-structural allocation of the system is illustrated in a functional-structural allocation matrix in fig. 7.

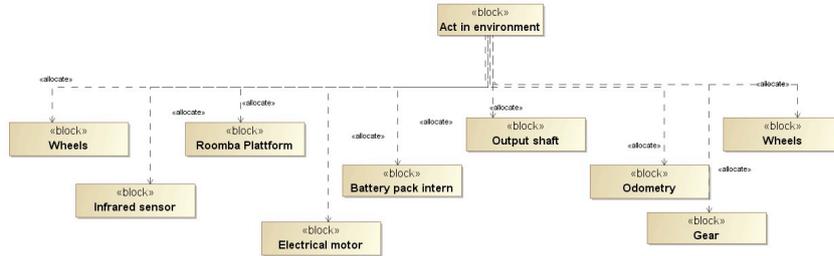


Fig. 6 functional-structural allocation of “Act in the environment”

The complete hardware configuration of the coffee robot waiter with characteristic design parameters is shown in fig. 8. The associated design parameters related to the hardware components are displayed as values and serve as basis for the subsequent value-oriented exploration of the solution space of design. The linkage between the models of the coffee robot waiter and external numerical solver for the computation of the solution space of design is done with the SysML-Parametrics diagrams.

	Additional wheel [Strabug...]	Additional wires [Strabug...]	Battery controls [Strabug...]	Battery pack ext. [Strabug...]	Battery pack inte... [Strabug...]	Coffee pot [Strabug...]	Current sensor [Strabug...]	D-Link USB-Stick [Strabug...]	Electrical motor [Strabug...]	Gear [Strabug...]	Glass plate [Strabug...]	Infrared sensor [Strabug...]	Internal watch [Strabug...]	Laptop-computer [Strabug...]	Laser range scan... [Strabug...]	LCD test display [Strabug...]	Micro-PC [Strabug...]	Odometry [Strabug...]	Output shaft [Strabug...]	Roomba Platform... [Strabug...]	User computer [Strabug...]	Weight sensors [Strabug...]	Wheels [Strabug...]
Functions	2	3	2	3	5	1	3	10	3	2	4	1	7	26	4	3	12	5	1	3	3	5	1
Act in environment	↗																						
Communicate with users																							
Convert electr. energy in m...																							
Decide best route																							
Detect coffee weight																							
Detect electr. energy state																							
Detect obstacle distance																							
Interact with users																							
Learn environment																							
Learn that avg. coffee cons.																							
Learn that avg. electr. cons.																							
Learn that map of working ...																							
Learn user ordering interval																							
Measure ordering interval																							
Perceive avg. coffee cons.																							
Perceive avg. electr. ener...																							
Perceive coffee order																							
Perceive electr. energy state																							
Perceive stat. u. dyn. obst...																							
Perceive user coffee cons.																							
Perceive user ordering inter...																							
Perceive working environment																							
Plan route																							
React to ext. disturbances																							
Reason about coffee range																							
Reason about energy range																							
Reflect on route																							
Regulate electr. energy																							
Schedule delivery																							
Think about orders																							

Fig. 7: Functional-Structural Allocation Matrix of the coffee robot waiter

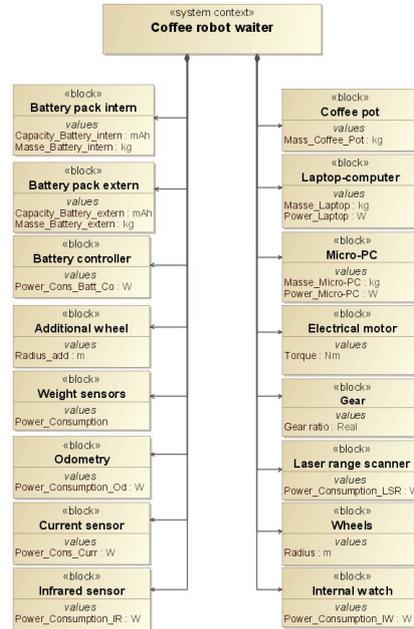


Fig. 8: Hardware modules composition of the coffee robot waiter

4.4 Use of causal loops to optimize design alternatives

The main task to assess the solution space of structural elements of the coffee robot waiter is to trade-off performance and structural design parameters with a view to requirements specification and technical constraints. At the same time, designers must make sure the coffee robot waiter has enough energy left to perform its cognitive functions while delivering coffee orders as fast as possible. As already stated in the description of the cognitive coffee waiter, a map of the environment with environmental variables such as the estimated position of the users is incorporated into its knowledge module. The coffee robot waiter is equipped with a coffee pot having a capacity of five cups and being able to deliver coffee at maximum to five users out of ten potential users in one tour after which it automatically returns to its starting point to refill the coffee pot and recharge its batteries. Fig. 2 illustrates the map of the environment with two tours we reproduced in an external numerical computing environment. Possible scenarios with boundary conditions are hereby defined with these assumptions.

Variable Parameters	
Voltage	[24:3:96] V
Electrical motor rotational speed	[2000:750:10000] 1/min
Power consumption of electronic components	[24:0.4:64] W
Gear Ratio	6:1:16
Speed range	0.1:0.05:0.4 m/s
Mass of the hardware components	[5.243:0.036:7.171] kg

Fixed Parameters	
Acceleration of gravity	9.81 m/s ²
Desired Acceleration during the delivery	0.2 m/s ²
Coefficient of friction	0.01
Wheel radius	0.035 m
Transmission efficiency	0.8
Mass of the coffee pot	0.728 kg
Mass of one coffee cup	0.3 kg

Table 1: Parameters employed for the optimization

From this, the coffee robot waiter chooses up to five users (represented in fig. 2) from the ten assumed available users (boxes on the map; blue boxes represent the locations of the unselected users during the delivery tour) and drives back to the starting point (colored in green, fig. 2). The traveling salesman algorithm has been computed to calculate the optimal route (see fig. 2) for the delivery. We did not include static and dynamic obstacles for this work. The simulation of the environment with the assumed user locations was numerically solved with the well-known Traveling Salesman Problem and has the objective to estimate the distance covered by the cognitive coffee robot waiter during the coffee delivery which is the basis for the energy consumption while moving. However, one of the most difficult problems encountered at this design stage when optimizing complex systems is, as explained above, the suitable estimation of their component design parameters whose values cannot be predicted with certainty. To cope with this issue, interval analysis has proven useful in bounding the values, by means of their minimum and maximum, of uncertain design parameters [2]. The system design parameters employed for this case study can be selected either on the basis of the designer's experience or on empirical values and are to be varied as shown in Table 1. Common parameters such as coefficients of friction, mass of the coffee pot can be assumed as fixed.

Based on these assumptions, a trade-off between the design parameters is done, the constraints related to the system's dynamic behavior and the optimization objectives. Fig. 9 shows the results of the performed simulation of the solution space. For a better understanding of the use case scenario, the distance travelled by the cognitive coffee waiter was divided in five different sub-distances, corresponding to the delivery of one coffee cup to a user. It is assumed that no obstacle disturbs during the delivery. It is also assumed, due to the significant energy consumption of activities requiring high level computation such as cognitive processing, that the energy consumption of the motion of the robot accounts only for half of the total energy consumption [7]. The feasible solution space of design shows that the results of the trade-off analysis are not obtained from the maximization or minimization of the assumed design parameters. We used a variance coefficient ($\text{var} = 0.2$) to express the deviation from these extreme values (maximum and minimum). This reflects the fact that the global optimum does not fulfill the previously defined requirements. Based on these results, designers are able to support their decision making process concerning the mass of the structural components and the energy consumption, by means of the battery capacity

the cognitive coffee waiter needs to perform its cognitive tasks, thus satisfying the optimization objectives.

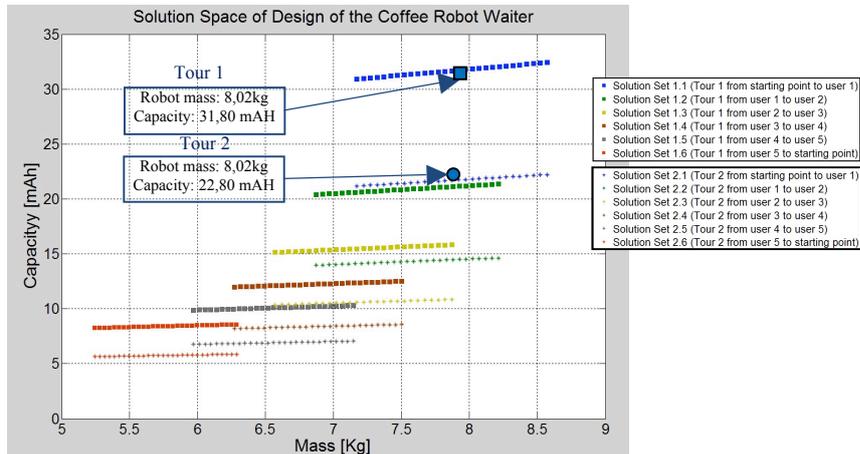


Fig. 9 Context-dependent feasible Solution Space of design of the coffee robot waiter

5. Discussion

The context-dependent assessment of the solution space of the coffee robot waiter, while considering two delivering tours (fig. 2 left), is illustrated in figure 9. The required capacity of the battery throughout the delivering from one user to the next and back to the starting point is illustrated. For example, with an assumed robot total mass of 8,02 kg, the battery must have at least 31.8 mAh in the first delivery simulation (tour 1, see fig. 2) from the starting point to user 1 and 22.8 mAh in the second delivery simulation (tour 2, see fig. 2) from the starting point to user 1. As expected, the mass of the robot as well as the distance between the users play a huge role in power consumption. The estimation of the driving distance with the TSP algorithm (see delivery tour 1 and 2 in fig. 2) has proven to be necessary for the approximation of the delivery distance. On a broader scale, simulating as many as possible scenarios and delivery tours is appropriate to consider many use cases before building a physical prototype. The assessment of the solution space is also possible regarding other design parameters from Table. 1. However, designers must be aware of the unpredictable delivering sequence of orders in the sense that users ordering can not be fully predictable. After simulation, it is possible to estimate depending on the mass and the delivering state how much energy the cognitive coffee waiter requires. Further work is needed to reasonably estimate the power consumption of electronic components, especially during high computation tasks such as the cognitive processing. This cannot be achieved without several testing procedures.

6. Conclusion

In this contribution, an approach is proposed to analyze and visualize the solution space of cognitive products using the cognitive coffee waiter as an example. The feed forward approach starts with the requirements specification up to the optimization and evaluation of design alternatives which are represented in the design solution space. On this basis, designers can computationally generate and verify several use cases and analyze the solution space to support their decisions concerning the choice of system design parameters.

7. References

- [1] Abdul Rahman, M. A., Mayama, K., Takasu, T., Yasuda, A. and Mizukawa, M. "Model-Driven Development of Intelligent Mobile Robot Using Systems Modeling Language", In "Mobile Robots - Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator", 2011
- [2] Antonsson, E.K., and Otto, K.N. (1995), 'Imprecision in Engineering Design', *ASME Journal of Mechanical Design* 117, pp. 25-32.
- [3] Cole, E.L., Jr., "Functional analysis: a system conceptual design tool [and application to ATC system], in IEEE Transactions on Aerospace and Electronic Systems, Vol. 34, Issue 2, pp. 354-365, 1998
- [4] Estefan, J. A. : Survey of Model-Based Systems Engineering (MBSE) Methodologies, Jet Propulsion Laboratory, California Institute of Technology, Tech. Rep., May 2007
- [5] Gero, J.S. (1990) 'Design prototypes: a knowledge representation schema for design', *AI Magazine* 11(4), pp. 26-36.
- [6] Kordon, M., Wall, S., Stone, H., Blume, W., Skipper, J., Ingham, M., Neelon, J., Chase, J., Baalke, R., Hanks, D., Salcedo, J., Solisch, B., Postma, M. and Machuzak, R. (2007) 'Model-Based Engineering Design Pilots at JPL' IEEE Aerospace Conference Proceedings, 3rd -10th March, 2007
- [7] Mei, Y.; Lu, Y.-H.; Hu, Y.C.; Lee, C.S.G. A case study of mobile robot's energy consumption and conservation techniques, Proceedings of the 12th International Conference on Advanced Robotics, 2005. ICAR '05, pp. 492-497
- [8] Metzler, T., Shea, K.: Cognitive Products: Definition and Framework, In: Proceedings of International Design Conference (DESIGN2010), May 17-20, Dubrovnik, Croatia, 2010, pp. 865-874
- [9] Metzler, T., Shea, K.: Taxonomy of Cognitive Functions, In: Proceedings of 18th International Conference on Engineering Design (ICED11), August 15 – 18, Copenhagen, Denmark, 2011
- [10] Metzler, T., Jowers, I., Kain, A., Lindemann, U.: Development of Cognitive Products via Interpretation of System Boundaries, In Proceedings of 4th Conference on Research into Design (ICoRD' 13), January 7th – 9th, Chennai, India
- [11] Paetzold, K., Ethische Aspekte bei der Entwicklung kognitiver technischer Systeme für die Unterstützung bei demenziellen Erkrankungen, In International Conference on Engineering Design, ICED'07, 28 – 31 august 2007, Cite des Sciences et de l'Industrie, Paris, France
- [12] Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H., "Engineering Design: A Systematic Approach", 3rd ed., Springer, 2007.

- [13] Rekuć S.J., Aughenbaugh J.M., Bruns M., Paredis C.J.J., “Eliminating design alternatives based on imprecise information.” In Society of Automotive Engineering World Congress. Detroit, MI, 2006.
- [14] Sop Njindam, T., Platen, E. and Paetzold, K. (2012) „Modellbasiertes Systems Engineering zur frühzeitigen Absicherung komplexer multidisziplinärer Systeme“, in Maurer, M. and Schulze, S.-O., (eds.) (2012), *Tag des Systems Engineering*, Paderborn, Germany, November 7th-9th, München, Hanser.
- [15] Suh, N.P. (1990) “The Principles of Design”, Cambridge, Massachusetts, Oxford Series on Advanced Manufacturing.

The Missing Link - between Requirements and Design

Armin Haße¹, Cees Michielsen²

¹Siemens Industry Software GmbH & Co. KG, Stuttgart, Germany
armin.hasse@siemens.com

²R&D Studio b.v., Maarheeze, The Netherlands
cees@rnd-studio.com

Abstract The combination of complex system design and management today requires a profound understanding of the relationships between the requirements and design processes throughout the product life cycle. To practice Systems Engineering merely as a technical matter is a serious misjudgment. Engineering is an interdisciplinary approach for all those involved in the Product Creation Process (PCP). The systematic design and creation of the product, based on intensive interaction between stakeholders, systems and subsystems, is performed on more than only at, what is usually called, the technical level of the PCP. This awareness inspired the authors to describe "Product Abstraction Levels" and align it with for instance the V-Model. It is shown how the information transformation path, starting from the customer, through marketing, engineering to purchasing and back to sales can be paved systematically without losing product-related information. It is shown how a common understanding about the product to be developed can be kept complete and consistent, despite a highly fragmented engineering activities and increasing parallelization of PCP sub-processes. The method how to achieve this, is shown in the way in which requirements are in actually interconnected. At the same time it becomes clear that most of today's existing requirements management tools are not able to support integration with the design process. "The Missing Link" refers to the world behind the various relations between bits of information and also what kind of support we really need for the development of complex systems.

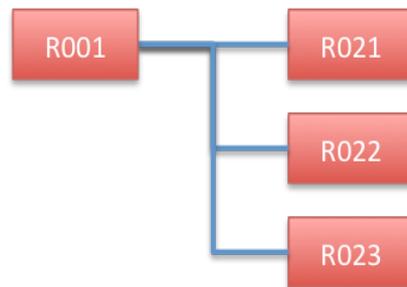
Keywords Requirements Management, Product Lifecycle Management, SITIO, Product Abstraction Levels, Design, Design Decisions

Visualization of relations between requirements

Despite new insights, a persistent image in many books, guidelines, user manuals and tutorials on the subject of requirements management is the *Requirements tree*. This article identifies a number of shortcomings of this type of visualization and suggests improvements. Especially when we look to the overall picture (which we will call the *Product Abstraction Level* landscape later, Fig. 8) it becomes obvious that crucial parts of knowledge are missing in the tree representation. As we go through the product abstraction levels and through the process phases in the Product Creation Process (PCP) we make it clear that the tree representation is not capable of visualizing the actual relationships between the necessary information elements, and in a way forces the users of the tree representation to model their product in an unnatural and often illogical manner.

To clarify the problem, we will use a simple example from the automotive practice: A high-level statement by management is the start of product engineering. As long as just this single statement or requirement is in focus, or even when this would be the only requirement, little seems to go wrong. Only when a second requirement is added and both need to be considered to reflect project priorities, the trouble begins. As we know, even at the highest product abstraction levels (business level, or operational use level with people trying to capture the customer needs) many requirements are identified and specified that need to be considered in conjunction with one another. That's where the problem starts.

Fig. 1 A simple requirements tree - How to interpret the relationships between the requirements as depicted in the Figure? Is it a parent-child-like relation or more an Abstract versus Content of an article? Does the picture (syntax) help the viewer to understand what the meaning (semantics) is?



In the following example it becomes clear that understanding the meaning of relations between requirements needs more than what is suggested in Figure 1. Suppose requirement R001 is a high-level need from top-management of a car manufacturer, saying: “*We need a better car than our previous model X*”. This is an unclear, not verifiable statement. There are several ways to interpret the relationship with requirement R001.

Example A: “By ‘better’ we mean an improved fuel consumption improvement (R021), improved driving performance (R022) and less emission of NOx (R023).” In this way (a bit) more detail expresses what is meant by R001. One could also say for this example A that $R001 = R021 + R022 + R023$. The requirements are at the same

abstraction level, where R001 can be seen as an *umbrella statement*, summarizing the other three.

Example B: “*The powertrain shall be optimized for fuel consumption (R021), the stability and brake performance shall be improved (R022), and the exhaust treatment system shall be improved (R023)*”. In this case requirements R021, R022 and R023 are derived from a design- and decision-making process and are allocated to systems at a lower abstraction level. Requirement R001 has led to the other three requirements: $R001 \rightarrow R021, R022, R023$.

In both examples it is not possible to determine if requirements R021, R022 and R023 represent a complete set of requirements to satisfy R001. In example B the design and the following decision-making process is not visible, which in practice will lead to problems when requirements must be met under specific constraints. These constraints are often resource-related: financial, time, personnel, material, capacity et cetera. Typically, these constraints lead to different options during the design process, and therefore also lead to different decisions.

Suppose we add requirement R002 to our structure: “*The sales price of the new car shall not exceed € 30.000 for the European market.*”

For example A this requirement could be an addition to R021, R022 and R023 as shown in Figure 2. In that case the set of product requirements is $P_{Reqs} = \{R021, R022, R023, R002\}$. For example B, requirement R002 is at the same abstraction level as R001. Both are requirements that need to be satisfied by the complete product.

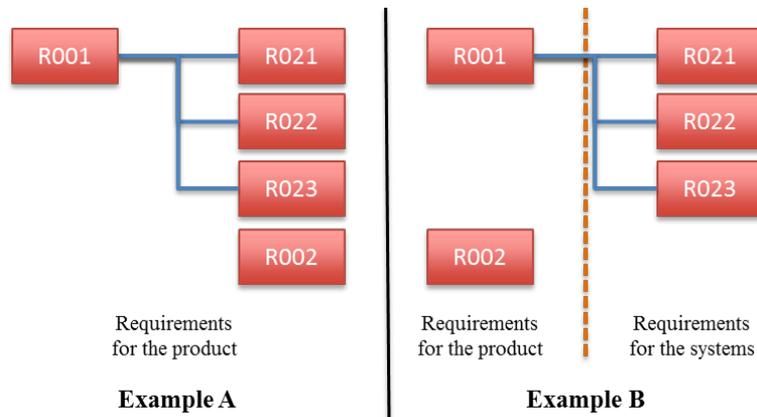
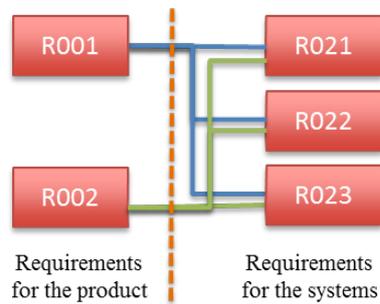


Fig. 2 a new high-level requirement R002 added.

Example B in Figure 2 shows the positioning of R002 at the same level as R001. No relations have been made, i.e. no requirements for the lower levels have been derived. In case R021 (concerning the fuel consumption) would not only have been derived from R001 but also from R002 then this could visually be shown by linking the two requirements. The content of R021 would be changed to reflect this relation

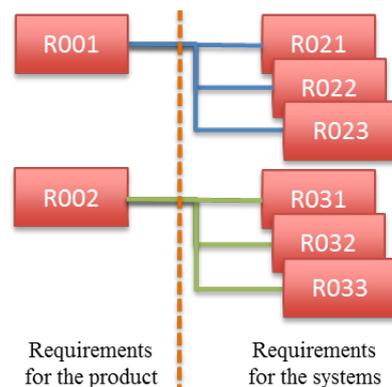
“The powertrain shall be optimized for fuel consumption, whereby the BOM cost for the powertrain shall not exceed € 2.500“. And although this relation seen from R021 seems sufficient, it remains unclear how the BOM cost for the powertrain was derived (this information is not stored as part of R002). In addition, how R002 is managed across the systems remains fully unclear.

Fig. 3 integrating requirement R002 - In case the cost for each system has been budgeted in coherence with the other product requirements (R001), this relation can be depicted as in the figure. In this case requirements R021, R022 and R023 are rewritten to reflect the consequences for each of the systems (powertrain, stability & brake and exhaust system).



System requirements R021, R022 and R023 are all linked to both product requirements R001 and R002. And although this would be more in line with what has been decided for both product requirements, the actual trade-off is not visible. Systems requirements cannot be justified purely based on their relation with product requirement without an explicit rationale about the available design options, their pro's and cons (trade-off) and the final decision on which the derived requirements are based.

Fig. 4 two independent requirements trees - In case R002 has not been integrally analyzed and budgeted in coherence with R001, then the cost-related requirements for the systems have no apparent relation with the existing requirements for the “better car”.



The structure in Figure 4 suggests that requirements R001 and R002 are independent. Summarizing we conclude that relationship diagrams or requirements trees that express relations between requirements only,

- Fail to show the difference between requirements that specify a customer need in more detail at the same abstraction level (Figure 1);
- Fail to visualize the multi-objective design and decision-making process results;
- Fail to justify the values in the derived requirements (they are merely capable of showing the result of an untraced decision);
- Fail to help the assessment of completeness for the derivation of requirements (the information of requirements R002, R031, R032, R033 together do not give a decisive answer);
- Fail to support the design and decision-making process at every abstraction level (even when it is made clear that two or more product requirements have been used to build a trade-off and to decide hereupon e.g. in Figure 3, no reference is made to these designs or decisions).

Visualization of the missing link between requirements

When we look at the development of requirements throughout the lifecycle of a product, we see that requirements are fed to a design process (1). Which requirements are addressed by a design is often implicit, i.e. not or poorly documented. During the design process one or more possible solutions/implementations for one or more requirements are identified (2). For each design option several assumptions or preconditions will be made about the systems/components that are part of the design option (3). During the design process the assumptions and preconditions are verified for feasibility with each system/component responsible (4).

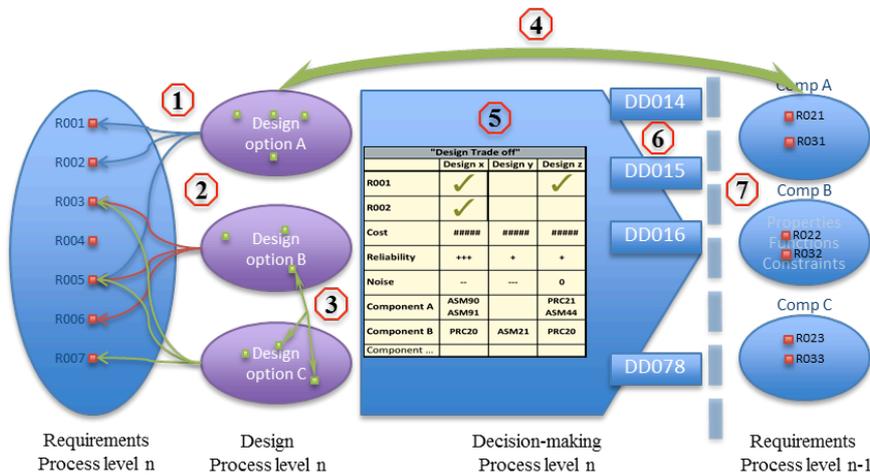


Fig. 5 the requirements derivation steps.

A trade-off table is made to enable the decision-making process (5). This table not only shows the relationships of the design options with the affected requirements, but also the required resources (financial, time, personnel, side-effects, ...) and the assumptions and preconditions for the systems and/or components that are part of a

design option. Once the decision has been made for a particular design option (6), the requirements for the next abstraction level (i.e. systems/components) can be derived from the assumptions and preconditions (7).

These process steps are similar for each product abstraction level (Figure10). It must be said however, that at each level different techniques and tooling can and should be used to accommodate the different types of stakeholders and their communication needs. An important aspect is to identify, specify and maintain the relationships that are made when decisions have been taken. The Design Decision (DD) objects are signposts that point to design options (*references*) and to requirements that are addressed by these design options (*trace links*). In turn, when requirements at lower levels are derived these requirements will point to the Design Decisions.

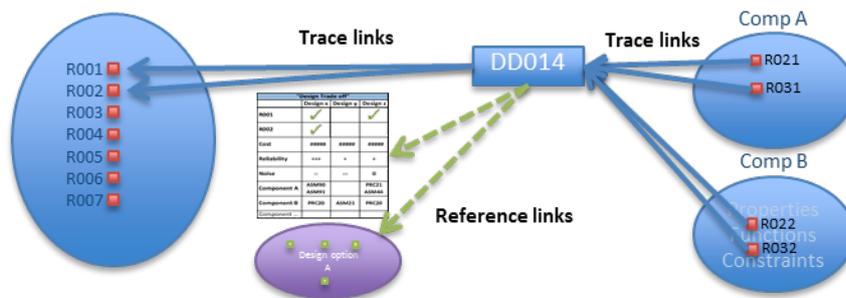


Fig. 6 the Design Decisions (DD) as signpost

The structure of the DDs themselves is simple: apart from the unique ID and a concise title, the DD points to one or more parent or upstream requirements using *trace links* and also points to zero or more design options, trade-off matrices, reports, and more using *reference links*. When we use the term traceability in the domain of requirements engineering we mean the paths that are created by the trace links. Trace links symbolize the path to *where the requirement originates*, is therefore unidirectional and always point upstream to its parent(s). In our requirements lifecycle model requirements are always linked upstream to one or more design decisions (unless it is the start of a trace also called a *demand*). In figure 6 requirements R001 to R007 are *demands* and have no parents to trace to).

Design Decisions can point to zero or more upstream requirements. There is one important restriction: an upstream requirement can be linked to by no more than **one** design decision. The reason is, that this DD represents the complete fulfillment for the requirement. When a requirement would have more than one DD pointing to it, it would be unclear which one fulfills what part of the requirement. This would bring us back to the problem statement at the beginning of this article.

In case of our example the structure could look like:

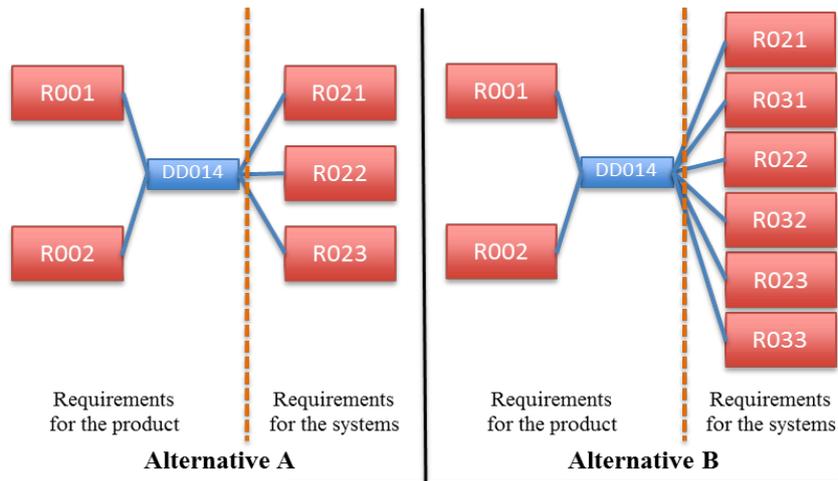


Fig. 7 Example B with alternative Design Decision

For both alternatives Design Decision DD014 holds the trace links to requirements R001 and R002, and all the reference links (as shown in Figure 6) to design options, trade-off tables, and (if not stated in the descriptive text of the Design Decision itself) to actual decision statements.

The difference between the two alternatives lies at the next abstraction level, where the systems/components either start with a requirement that is derived based on a design decision and that has been rewritten to reflect the priorities in the one requirement (e.g. R021: “The powertrain shall be optimized for fuel consumption, whereby the BOM cost for the powertrain shall not exceed € 2.500.”).

Or, the systems/components start with two separate requirements (e.g. R021 “The powertrain shall be optimized for fuel consumption” and R031 “the BOM cost for the powertrain shall not exceed € 2.500.”).

Often the allocation of requirements to lower levels is documented as tables. Initially, these tables show the product requirements in the first column, followed by the requirements allocated to the systems and components in the next columns.

Incomplete Req table	Systems/Components		
	S1	S2	S3
Req's (as in Figure 3)			
R001	R021	R022	R023
R002			

Incomplete Req table	<i>Systems/Components</i>		
<i>Req's (as in Figure 4)</i>	S1	S2	S3
R001	R021	R022	R023
R002	R031	R032	R033

In table format the Design Decisions clearly indicate which product requirements are covered, and also which lower level requirements are derived from it. In this way the DDs can be used to measure the requirements coverage: which requirements are covered by a design? The DDs are also used to verify if the set of derived requirements is complete represent all assumptions and preconditions in the design.

Alternative A		<i>Systems/Components</i>		
<i>Req's</i>	<i>Design Decisions</i>	S1	S2	S3
R001	DD014	R021	R022	R023
R002				

Alternative B		<i>Systems/Components</i>		
<i>Req's</i>	<i>Design Decisions</i>	S1	S2	S3
R001	DD014	R021, R031	R022, R032	R023, R033
R002				

The tables above represent the same information as in Figure 7. In addition, the last three columns also show the Systems/Components to which the requirements are allocated.

Product Abstraction Levels

Examples from industry show combinations of a V-model or pyramids with abstraction levels in it. The basis for creating levels often is a common goal to manage and control information that has a logical coherence. Examples of these goals are: to manage risks [2] or to manage the product requirements transition [3]. In practice not only the information is structured at specific levels, also the roles, responsibilities and processes that are relevant to manage the goals are part of the level definition. In this section the product requirements transition is the central theme for defining product abstraction levels (PALs).



Fig. 8 Product Abstraction Levels: Operational use level (1), Functional level (2), Technical level (3), Implementation level (4)

A typical enterprise objective is to provide products (goods or services) to sell to customers. In turn, customers often need bespoke solutions to meet their needs. At the *operational use level* (business or solution level) the business analysts capture these customer needs as the basis for finding a solution for them.

The first translation is made at the business level where the customer needs are placed in context with the operational use of the product, including the legal, cultural, environmental and political aspects. The results of the analyses are the *Stakeholder Requirements*.

The solution (i.e. the fulfillment of the Stakeholder Requirements) can be made either from available products (i.e. the cars in the showroom) or from products to be developed. The search for a solution at the business level can also be seen as a design process: in what way can we meet the stakeholder requirements in the most effective and efficient manner?

When the solution is not available (yet) at the business level, the requirements must be specified for the product to be developed. At the business level the *product* is seen as a black box defined by its functions and properties and limited by the product's constraints. The translation at the business level can be described as a process where the stakeholder requirements are translated into functional and quality requirements for the product to be developed:

- What are the required properties of the product?

- *Drivability, comfort, reliability, ergonomics, styling, safety, security, ...*
- What are the required functions of the product, and how well should these functions perform under which conditions?
 - *Central door locking, park assistance, cruise control, ...*
- What are the limitations or constraints for the product design?
 - *Total Bill of Material cost, external interfaces, design envelope, ...*

When a complete translation is made for all Stakeholder requirements into Product requirements, and the (relevant) stakeholders agree to this translation, the product development can go to the next phase and can start designing the product. This is one of the crucial decisions made within product development and is symbolic for the responsibility at the business level. It means that the people responsible at business level for ensuring that the stakeholder requirements are understood and met as well as the people responsible for *running the business*, and the people responsible for developing the product, have a common understanding (represented by the functional product requirements) about the outcome of the development process.

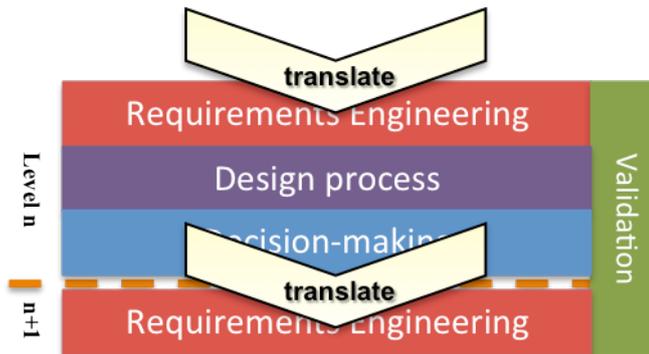


Fig. 9 Information translations in between the product abstraction levels. The Requirements Engineering process is at the receiving end of the translation. In each level the same three processes are involved in the translation process.

The idea behind the PALs model is to acknowledge the need on each level to express the requirements and design process in such a way that the goals of the level are achieved. The number of levels is determined by the necessity of each level to find a solution outside its own scope. In case we were to determine the number of different levels for a car shop that buys and sells cars, one level would suffice. In the model we use in this article our reference is a car manufacturer; In this case the number of levels is 4. The nature and content of each level differs. You will find a lot of decompositions at the Technical level, right up to the disciplines (mechanics, electronics, software, ...) that generate the requirements specifications for the Implementation level.

From a Model-Based Systems Engineering perspective often only the technical (3) and implementation levels (4) are documented and supported by methods and tools. We have experienced that it is worth the trouble to extend the model with the business (1) where the customer solution is delivered and with the functional level (2) as an intermediate between the business and technical level to describe the required product properties and functions.

The *Operational use* level is focused on

- Capturing the customer needs;
- Analyzing the operational use terms and conditions;
- Determining the product roadmaps;
- Determining the product's scope;
- Determining the product development process conditions;
- Finding solutions;
- Determining the Functional architecture, including the interaction and dependencies between the properties and functions;
- Transforming customer demands into a deliverable solution.
- Deliver the solution to the customer;

Normally, the creation of solutions follows a business interest to sell available products that fulfill the solution requirements and constraints (sufficiently for the customer). Sometimes the constraints are more decisive for customer satisfaction than the product requirements (“*I need a new car this week!*”), in which case the customer buys a car from the show room that comes closest to his needs, rather than waiting for the product to be developed.

The *Functional* level functional level focuses on

- Capturing the product's Properties, Functions and Constraints;
- Simulating and validating the overall behavior and feasibility;
- Finding integral solutions (design options);
- Determining the Systems architecture, including the interfaces between the systems;
- Develop trade-offs in which the design options are compared to each other;
- Transforming functional, quality and limiting product requirements into requirements for the specific systems that play a role in the solution, where assumptions made in the product's design are allocated to these systems and requirements are derived from it (see Fig. 5);
- Agree on acceptance criteria and process with people responsible for the systems or subsystems at the Technical level;
- Integrate the systems into functionally complete product;
- Test and release the product, including the product specification (documented results of the tests and measurements to describe the precise behavior and properties of the product);

Just as all the other levels, the functional level defines the **complete** product in its own way. All functional, quality and limiting requirements must be allocated to specific systems at the Technical level.

The *Technical* level focuses on:

- Capturing the Property, Function and Limiting requirements for each system (each system must be fully specified in relation to the overall product);
- Decomposition of each system, including the intra-system interfaces;
- Design for each system;
- Deriving requirements a result of the systems and subsystems design process to the various disciplines that play a role in the development of the product;
- Developing discipline-based architectures and designs;
- Specifying the requirements for the components or elements at the Implementation level;
- Agree on acceptance criteria and process with people responsible for the components/elements at the Implementation level;
- Integrate the components into functionally complete systems;
- Test and release each system, including the system specification (documented results of the tests and measurements of each system to describe the precise behavior and properties of each system);

The *Implementation* level focuses on

- Using the requirements and detailed design to construct, make, manufacture, or program the components/elements.
- Perform unit tests;
- Release the components (including a component specification describing the precise behavior and properties of the component as found during test and measurement);

It is important to understand that the levels described above represent the product dimension by means of a specification structure in which the dependencies and decompositions of all the product elements are shown in relation to each other.

When we look at the process dimension, we see all the processes that are required for the transformation in the product dimension, are performed in parallel to each other. Typical milestones and toll-gates, phase descriptions are part of the process dimension and are not discussed in this article. Besides the process dimension we also see the need for the people dimension to complete the whole picture: *who* does *what*, *why*, *when*, and for *how much*. The people dimension is crucial to have good quality transformation of information between the levels. Using the SITIO [1] method, a high level of confidence can be achieved about the conformance to expectation, the fitness-for-purpose and the intrinsic quality of requirements.

A Product Abstraction Level in a Nutshell

For each level two major *inputs* are known: (I_1) needs/decisions/trends of the level above and (I_2) known properties & functions/deliveries/outputs of the level below. For each level two major *outputs* are known: (O_1) requests & assumptions/decisions and (O_2) deliveries. This means that within the scope of each level, processes are used to produce the outputs, based on both inputs for that level. It also means that the responsibilities can be directly coupled to the roles within these processes. As an example, the figure below shows 7 major processes that either support the information transition downstream or support the solution delivery upstream.

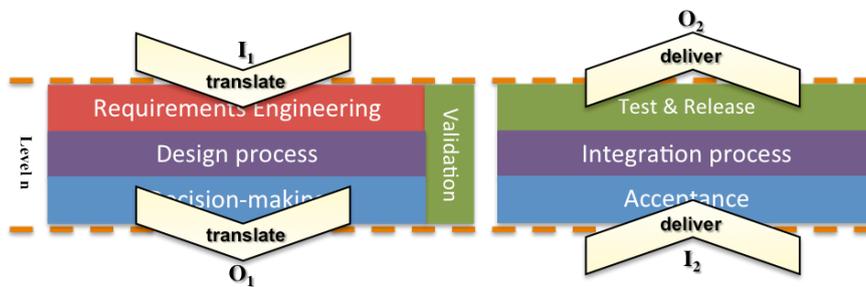


Fig. 9 the seven main processes and inputs and outputs of each level

The four levels describe four basic views of the product. These views are in essentially determined by the active roles at the levels (people) and their objectives. Each level in turn can operate independently, providing results (outputs, O_x) and processing requirements (inputs, I_x). The arrows *translate* and *deliver* symbolize the direction of the information/delivery and also the source of this information/delivery.

In practice these two movements are a common process of continuously interaction of roles at these levels. It also indicates that the quality of the translation results depends on the persons that have specific responsibilities at these two levels. E.g. the customer can raise the quality of the requirements by being more specific about his/her needs, and at the same time the marketing representative who is responsible for gathering the product requirements can raise the quality by applying the skills to gather, analyze, specify and validate the requirements. For more see SITIO (Secure Information Transformation from Input to Output) [1] methodology.

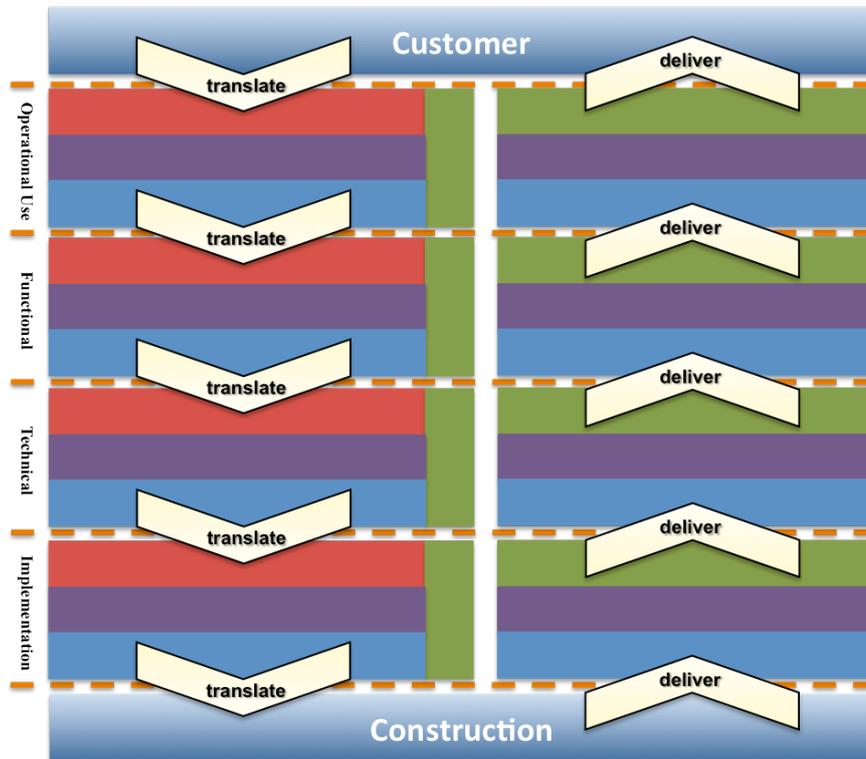


Fig. 10 The seven major processes are found at each level.

Before going into more detail about the processes in the abstraction levels it is important to understand the function of translate and deliver.

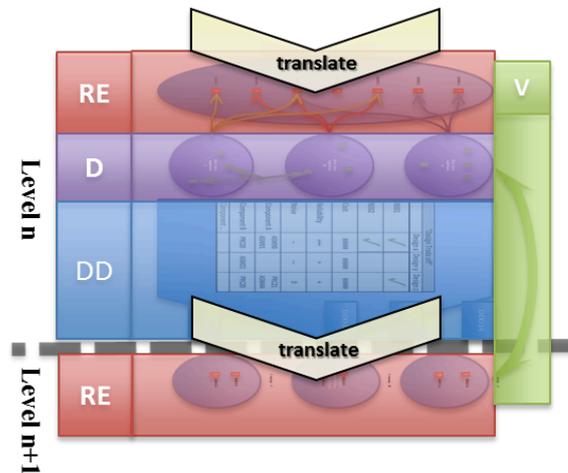
Translate – at the *Business level* the incoming *Translate process* (I_1) is often triggered by customer needs or demands. Characterized by pushing (customer) and pulling (marketing) activities. The translate process itself encompasses all the steps and means to communicate these demands to the requirements responsible (Business Analyst or Requirements Manager roles typically in marketing departments). The communication means differ at every level. Whilst e-mail or meeting minutes can be used to transmit the demands between customers and the business, other means (e.g. user stories or use cases) are more appropriate at lower abstraction levels, e.g. at the Business level for the outgoing translate process (O_1).

Deliver – at the *Solution level* the incoming *Deliver process* (I_2) delivers the already produced/available/released products. These products will then combined and configured (assembled) to the desired individual customer solution. This is often done by agents (e.g. shop assistant or sales persons). Then the individual solution will delivered to the customer via the outgoing delivery process (O_2). A typical ex-

ample could be a car dealer. He has usually not only the product “Car” available, but he also has banking products (like lease or loan agreements) as well as various service options/products. The combination of these products allows him to respond to individual customer needs and assemble a customized solution.

About the processes – at each level the design process is fed with the decisions and the derived requirements from the level above. In addition, the design also has knowledge of the deliveries done in the past by the level below. Design options are created that fulfill the requirements. Design options are validated, weighed and decided upon. Requirements for the next level are derived from these decisions. This sequence is repeated until the implementation is completed. Then the integration and test processes start to go upwards in the product dimension.

Fig. 11 Figure 5 turned 90° labeled with the requirements & design processes. Requirements Engineering (RE), Design (D), Decision Making (DD), Validation (V)



By themselves these processes are not new, the model presented in this article merely emphasizes the application of these processes in the transformation of information in the product dimension. The model also makes clear that the seven main processes are used at every abstraction level throughout the entire life cycle of the product. When one would describe each process individually, one must take care that the nature of the deliverables differs at each abstraction level. I.e. requirements do not have the same format at each abstraction level. Designs use different communication means and formats to ensure correct interpretation of the requirements. Where SysML can work very well on the Technical level to communicate between the systems, it may be fully inadequate to communicate at business level. The authors wish to stress that this awareness is a huge advantage when complex products are developed in large organizations, where the decision-making process from a political or social viewpoint are troubling the view on the product’s structure and dependencies. In other words, understanding the product’s structure helps to get a grip on the other two dimensions: process and people.

Conclusion

In summary, we conclude that the traditional (*tree structured*) way of visualizing requirements does not do justice to the inherent complex nature of current products and product development processes. The structured and systematic method described in this article makes the information transformation process transparent and explicit throughout the life cycle of the product.

By adding the *missing links* (Design Decisions) to the traces, a complete, traceable, verifiable and justifiable structure is made throughout all product abstraction layers. Information is structured and described by means that are appropriate to the direct stakeholders for each level. No information gets lost or gets degraded by deploying the product abstraction levels, the design decisions and SITIO.

The awareness of professionals to know where the parts they work on fit in the overall structure; the constant awareness of what is *up* and what is *down* in the product structure helps to determine how complete and how good the specifications are and what the dependencies are. By using the product abstraction levels as a map, everyone involved in the product development process is better equipped to help others and at the same time can be helped by others more effectively.

In this way complex systems can be developed and managed in a disciplined and structured manner.

References

1. Rebel, Martin; Fischer, JörgW; Haße, Armin; Michielsen, Cees (2013): Enhancing Interpretation-Quality of Requirements Using PLM Integrated Requirements-Communication in Cross Company Development Processes. In: Michael Abramovici und Rainer Stark (Hg.): Smart Product Engineering: Springer Berlin Heidelberg (Lecture Notes in Production Engineering), S. 43–50.
2. H. Negele, S. Wenzel, T. Pfletschinger and G. Getto: Successful Implementation and Application of Continuous Risk Management to Complex Systems Development in the Automotive Industry. In: Published and used by INCOSE 2005, S.4
3. Prof. A. Katzenbach: Informationstechnik und Wissensverarbeitung in der Produktentwicklung. Vorlesung (Handout) 2009/2010 Universität Stuttgart 2009, S.28(14)

Global Safety Management Method in Complex System Engineering

Romaric Guillerm^{1,2}, Hamid Demmou^{1,2}, and Nabil Sadou³

¹ CNRS, LAAS, 7 avenue du Colonel Roche, F-31400, Toulouse, France

² Université de Toulouse, UPS, LAAS, F-31400, Toulouse, France

³ SUPELEC / IETR, avenue de la Boulais, F-35511, Cesson-Sevigne, France
guillerm@laas.fr ; demmou@laas.fr ; nabil.sadou@supelec.fr

Abstract. In System Engineering, one of the most critical process is the requirement management, particularly when it deals with the safety requirements. These one are non-functional requirements and are related to emergent properties, which come from the integration of the different system components. They must be identified as soon as possible, because they are guards to validate or not the system, which can require changes in system architecture. Moreover, they are formulated at system level and need to be declined at sub-system level.

The objective of this paper is to propose a global safety management method based on well-known safety methods, in order to organize the different tasks to make the system safe. The method focuses mainly on the definition of the system safety requirements following risk and hazard analysis, and also on their declination according to a top-down approach. It is based on the famous Failure Mode, Effects, and Criticality Analysis (FMECA) and the use of Fault Trees and Event Trees.

Keywords: Safety requirement ; Requirement engineering ; Complex system.

1 Introduction

Modern systems are increasingly complex [1]. Indeed, they integrate more and more different technologies, offer more functions, and have complex interactions between their components. The processes and the design methods must evolve to reflect this growing complexity [2], [3]. In particular, for our purposes, the management of properties such as reliability or security [4] must evolve accordingly, to ensure and enable the necessary level of confidence [5]. For an effective consideration of safety in the design process, it is necessary to consider safety in overall studies by the engineering system processes. The safety properties must be defined globally ; that is to say elicited [6]. Once these safety properties are identified, they must be declined locally to be actually realized by the system. The local properties associated with subsystems must be satisfied to ensure the global properties, reaching issues of traceability [7], [8] and requirements engineering [9].

Requirements Engineering (RE) is one of the System Engineering (SE) processes. RE is a crucial process within the development of complex system. Safety requirements are classified as non-functional requirements and are related to emergent system

properties. They cannot be attributed to a single system component. Furthermore, non-functional requirements are fundamental to determine the success of a system. Two activities are defined in RE. The first one concerns requirements development including the processes of elicitation, documentation, analysis and validation of requirements. The second one concerns requirement management which includes the processes of maintainability management, changes management and requirements traceability.

The work presented in this paper concerns a part of our approach for the integration of safety in system engineering processes [10]. It is an improvement and extension of the method presented in [11], that was inspired from [12] with an engineering process and requirements point of view. The approach allows taking into account the safety requirements in system engineering process to facilitate traceability of these requirements throughout the life cycle of the system. It concerns the two activities of RE: the development and the management activities. The paper presents a method that allows to define, derive and decline system safety requirements, with the combination of several FMECA (Failure Mode, Effects, and Criticality Analysis) [13], Fault Trees analysis [14] and Event Trees analysis [15]. This paper contains four parts. The second one presents the system engineering framework of the method. The third one exposes the method for safety requirement definition and declination, with its different steps. Finally, the last section concludes the paper and presents some perspectives.

2 Context

In this part, the context of our work is exposed. The first section presents the System Engineering notion. Then, the standard that we adopt is presented with its useful concept of building block that devises the design in different system layers. To finish, a focus is done on the safety requirement management.

2.1 System Engineering

System Engineering (SE) is an interdisciplinary approach, whose objective is to assist the development of new systems. It contains collaborative and interdisciplinary processes of resolution of problems, supporting knowledge, methods and techniques resulting from the sciences and experiment to define a system, which satisfies an identified need, and is acceptable for the environment, while seeking to balance the total economy of the solution, on all the aspects of the problem in all the phases of the development and the life of the system. SE concepts are adequate specifically for complex problems [16].

SE is the application of scientific and engineering efforts to:

- Transform an operational need into a description of system performance parameters and a system configuration, through an iterative process of definition, synthesis, analysis, design, test and evaluation.
- Integrate reliability, safety, maintainability, expandability, survivability, human engineering and other factors into the total engineering effort to meet cost, schedule, supportability and technical performance objectives.

SE is the global framework of the approach proposed in this paper.

2.2 EIA-632 Standard

A standard currently used in the industrial and military fields is the EIA-632 standard [17]. Our work is also based on it.

Briefly, this standard covers the product life cycle from the needs capture to the transfer to the user. It is constituted by 13 processes grouped into 5 sets (see Figure 1):

1. Technical management processes (three processes): these processes monitor the whole process ranging from the initial idea to building a system until the delivery of the system.
2. Acquisition and supply processes (two processes): these processes ensure the supply and acquisition (and are very close to logistics).
3. System design processes (two processes): these processes deal with the elicitation and the acquisition of requirements and their modelling, the definition of the logical design and its physical solution.
4. Product realization processes (two processes): these processes deal with the implementation is-sues of the system design and its use.
5. Technical evaluation processes (four processes): these processes deal with verification, validation and testing issues.

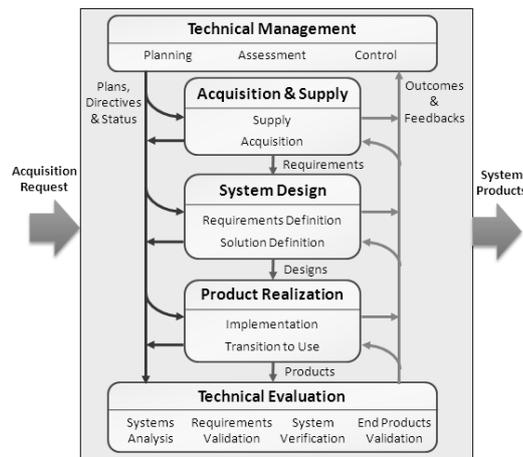


Fig. 1. EIA-632 Standard System Engineering Processes

2.3 Building Block Concept

The EIA-632 standard adopts an original and interesting system decomposition based on the concept of "building block". A building block is the association between one (or several) final product and a set of enabling products, as shown in Figure 2.

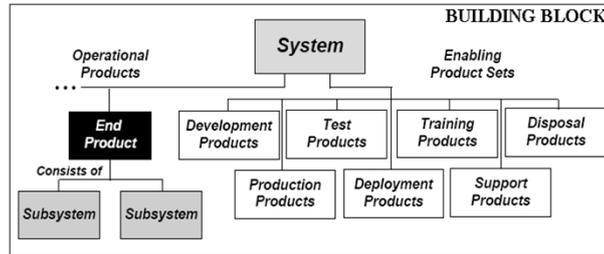


Fig. 2. One Building Block

In fact, the system is seen as a hierarchy of building blocks. The solutions defined in the upper layer (level) blocks, described by a set of specified requirements, are allocated as input requirements for the lower layer blocks (see figure 3). Finally, the building block decomposition is stopped when blocks correspond to on-the-shelf components or when their realization can be subtracted. With this description, we identified the need of deriving the safety requirements through the hierarchical decomposition.

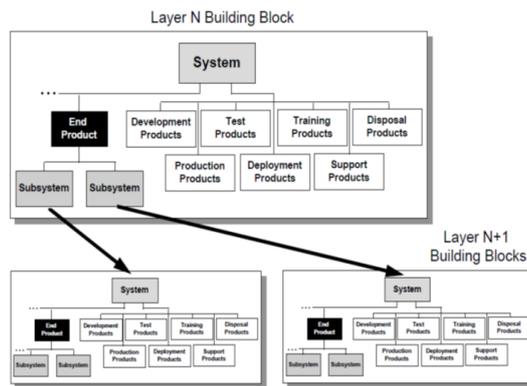


Fig. 3. Multilayer building block

2.4 Safety Requirement Management

To clearly situate the position of the method for deriving safety requirements, the Figure 4 gives an overview of the involved EIA-632 system engineering processes.

Among the different possible sources of safety requirement we can find the requirement provided by some dependability analysis as shown in the Figure 4. In this paper we consider this source of requirements. The proposed approach is used to define, derive and decline safety requirement with different safety analysis.

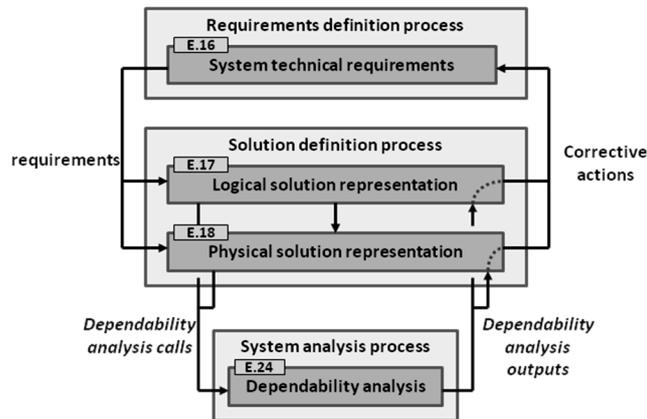


Fig. 4. Dependability analysis as a source of requirements

3 Global Safety Management Method

In this second part, the global safety management method is presented. First, an overview of the method is given, following by an explanation about the different kinds of safety requirements that are taken into account in the current version of the method. Afterwards, the 9 steps of the method are explained in details.

3.1 Overview

The method assumes that a complex system is composed of some subsystems (the principle of Building Block of the EIA-632 standard). It combines FMECA, fault trees and event trees, and has the objective to define all safety requirements at system level and to decline them locally at subsystems level with a goal of traceability. The Figure 5 summarizes the process associated to the method and illustrates how the different steps are integrated together.

3.2 Classification of the considered Safety Requirements

The method enables to identify and deals with several kinds of *safety requirements*. We have classify these requirements into subcategories, which are :

- *Reliability requirement*, that claims a quantitative objective in term of reliability properties.
- *Architectural requirement*, that defines an architectural design to deal with safety (like redundancies).
- *Active functional security requirement*, that is related to an additional security equipment (protective barrier) that can participate to reduce the probability of an accident.
- *Passive functional security requirement*, that is related to an additional protective or mitigation equipment that can reduce the severity of an accident.

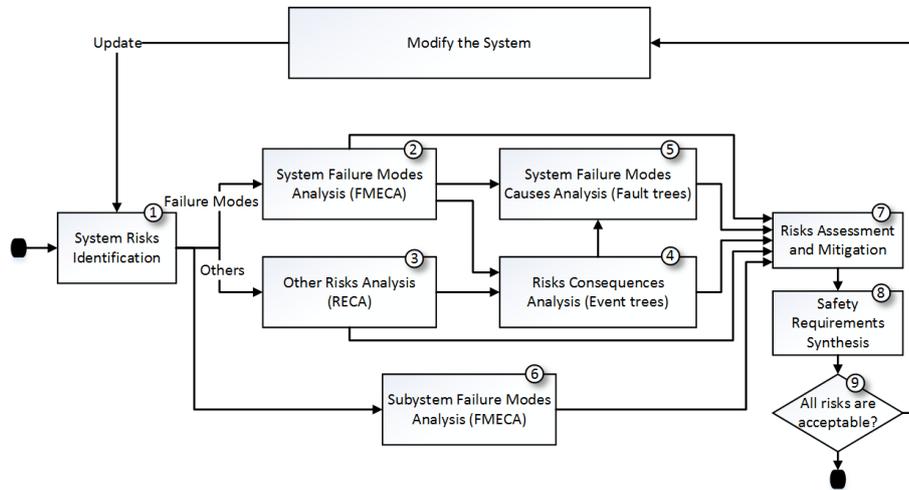


Fig. 5. Overview of the global safety management method

3.3 Step 1: Risks Identification

The first step is to identify and classify all the system risks. These can be human actions, external failures, internal failures (those of the system) or environmental conditions. The classification must be done in two groups: the risks representing internal failure modes and the other risks.

3.4 Step 2: System Failure Modes Analysis

The second step is to begin the analysis of risks that correspond to system failure modes. The recommended method is the FMECA [13], that is a technique used to identify, prioritize, and eliminate potential failures from a system, a design or a process. Concretely, this step is to complete few columns of the FMECA table (others than severity, probability, criticality and corrective action) (see Table 1). For each system function, we identify failure modes, causes of these modes and effects on the system (possibly depending on the phase, state or mode). For the identification of failure modes, lists of generic modes have been defined in some standards like CEI 60812: 1985 [18]. The effects are here the potential accidents.

In fact, we also propose some changes in the classical FMECA to clarify the method, visible in the Table 1. A distinction is made between the probability and the detectability of the failure modes and those of the effects. Indeed, between a failure mode and an effect (accident), there is a set of involved cofactors (protection barrier, environmental condition ...), recorded in the "condition" column of the table. These conditions will be identified during the step of consequences analysis.

The assessment of the probability of the risk and the assessment of the severity, probability and criticality of the effect will be done during the step of risk assessment. The corrective actions will be proposed at the risk mitigation step.

Table 1. FMECA table

Function	Failure Mode (FM)	Probability (of the FM)	Detectability (of the FM)	Causes	Effect (Accident)	Condition	Probability (of the Effect)	Severity (of the Effect)	Detectability (of the Effect)	Criticality (of the Effect)	Corrective Actions	Acceptable
::	::	::	::	::	::	::	::	::	::	::	::	YES/ NO

3.5 Step 3: Other Risks Analysis

This step is similar to the previous step of system failure modes analysis, but focuses on the other risks (external). The recommended method is to use the principle of an FMECA, that we can call here RECA (Risks, Effects, and Criticality Analysis) (see Table 2). This step is to complete few columns of the RECA table (others than severity, probability, criticality and corrective action). The effects are also the potential accidents.

Table 2. RECA table

Actors	Risk	Probability (of the Risk)	Detectability (of the Risk)	Reason	Effect (Accident)	Condition	Probability (of the Effect)	Severity (of the Effect)	Detectability (of the Effect)	Criticality (of the Effect)	Corrective Actions	Acceptable
::	::	::	::	::	::	::	::	::	::	::	::	YES/ NO

The same remarks as for the FMECA remain true concerning the probability, the detectability and the severity of the failure modes and the effects, and the conditions.

3.6 Step 4: Consequences Analysis

In this step, the consequences of all the identified risks (system failure modes and others) must be analysed. This step is to identify how the risks contribute to an accident. It can be done using event trees [15] to visualize the possible chains of events that led

from the risk to the accident, through branching points representing protective measures or interventions (cofactors) (see Figure 6). The minimal cuts associated with the various accidents are also identified.

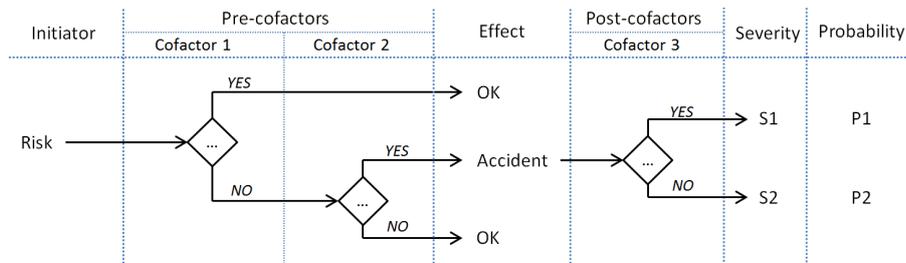


Fig. 6. Event tree

A generic example is given in Figure 6. The effects correspond to accidents, whereas the term consequences refers to events or factors involved in the causes to consequences relationship, starting from the analysed risk.

3.7 Step 5: Causes Analysis

The fifth step is to conduct an internal analysis of the system by identifying the causes of system failure modes. These causes analysis must lead to subsystems failure modes. For this step, the use of fault trees [14] is recommended. Indeed, a fault tree provides a simple modelling way to represent the interactions between components from the point of view of reliability. Static fault trees use traditional Boolean logic functions to represent the combination of component failures (events) that cause system failure.

So, the top event of each tree corresponds to a system cause. The objective is to determine the causes of the top event (using logical operators such as AND and OR) in the sub-systems. The leaves of the fault tree correspond to sub-systems failure modes (see Figure 7).

In fact, the system failure modes analysed correspond either directly to a system risk (defined in the first step), or to a cofactor of an event tree which is a system failure mode.

3.8 Step 6: Sub-systems Failure Modes Analysis

An analysis of the subsystems failure modes should be led in parallel, using FMECA. The subsystems failure modes used in step 5 re-appears (the principle of the FMECA analysis). This FMECA will define the corrective actions at the sub-systems level that are representative of subsystem reliability requirements.

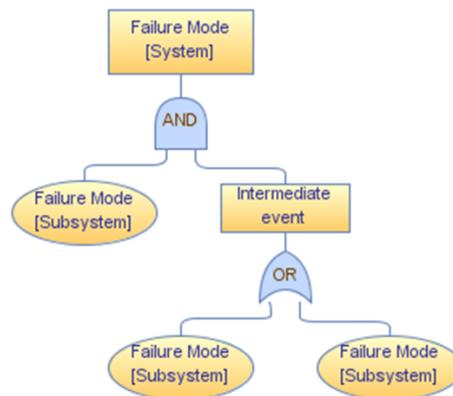


Fig. 7. Fault tree

3.9 Step 7: Risks Assessment and Mitigation

The seventh step is the central one. It deals with risks assessment and risks mitigation with definitions of corrective actions.

Assessment The risk assessment consists in defining the severity and the probability of the identified accidents, in order to evaluate the criticality. This information must be recorded in the various FMECA and the RECA tables. Concerning the probability, this one must be evaluated based on the fault trees and the event trees. Finally, it must be decided whether the risks are acceptable or not.

Mitigation The risk mitigation consists in advocating corrective actions (to be filled in the FMECA and the RECA) for the risks qualified as "non-acceptable" during the risk assessment step, in order to make them become "acceptable". The corrective actions can:

- Reduce the *probability* of the accident, by:
 - Fixing an *objective of reliability* with a *reliability requirement* (at system or subsystem level).
 - Modifying the system architecture for a better reliability (with redundancies for example) with an *architectural requirement*, that derives from the *reliability requirement* of the *objective*.
 - Adding an additional security equipment (protective barrier) with a *active functional security requirement*, that derives from the *reliability requirement* of the *objective*. During the next iteration of the method, reliability requirements will be defined for this security equipment based on the analysis of the failure modes in which it participates.
- Try to satisfy a *criterion*, for example:

- A *single failure criterion*, adding a security equipment (barrier) to increase the number of failures before the occurrence of an accident, with an *active functional security requirement*.
 - A *spatial dispersion criterion*, with an *architectural requirement*.
 - A *redundancy with separate development criterion*, with an *architectural requirement*.
- Reduce the *severity* of the accident
- Adding a protection or mitigation equipment, with an *passive functional security requirement*.

Note: This is not the only possible corrective actions (preventive maintenance for example). Other types of corrective actions will be incorporated in future work to improve the process.

3.10 Step 8: Safety Requirements Synthesis

Before eventually transferring the change requests to modify the system, this step will summarize the results in terms of requirements, declination of requirements, and traceability links between requirements and accidents, or requirements and requirements. As in the first version of the method [11], the declination part is based on the following 3 types of relations:

- System causes and system corrective actions,
- System causes and sub-systems failure modes,
- Sub-systems failures modes and sub-systems corrective actions.

A generic example of this synthesis is given in Figure 8.

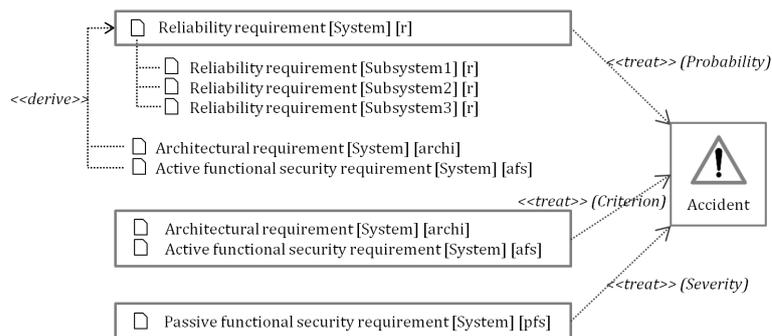


Fig. 8. Requirements traceability synthesis

3.11 Step 9: Stop Criterion

The process is finished once all the risks are considered as "acceptable". If this is not the case, a change request must be ordered to modify the system and the method must be reapplied from the beginning by updating the different analysis.

4 Conclusion

The method provides a support framework to define system safety requirements with an objective of traceability and requirements declination and derivation. The interest is multiple for the safety field: the method deals with the safety elements (failure modes, safety requirements...) and it is done with a comprehensive system engineering (with traceability and requirements declination) which is a factor contributing to safer systems. This method is compatible with the standard EIA-632 [17], and it extends the principle and strengthens the links between failure modes researches and analysis (FMECA), causes analysis and effects analysis.

In this work, several safety attributes are taken into account, like reliability, passive security and active security. They correspond to the given classification of safety requirements, which are themselves defined from the corrective actions. Other requirements concerning maintainability or availability should also be considered in further study. The probability, the severity and the criticality was treated through the FMECA. However, the work still doesn't consider the detectability aspect. We also should update the tool that implements the first version of the method presented in [11].

References

1. Chavalarias, D., Bourguin, P., Perrier, E., Amblard, F., Arlabosse, F., Auger, P., Baillon, J.B., Barreteau, O., Baudot, P., Bouchaud, E.: French roadmap for complex systems 2008-2009. French National Network for Complex Systems (RNSC), Paris Ile-de-France Complex Systems Institute (ISC-PIF) and IXXI, Entretiens de CargÃlse (2008)
2. Juristo, N., Moreno, A.M., Silva, A.: Is the european industry moving toward solving requirements engineering problems? *IEEE Software* **19** (2002) 70–77
3. Komi-Sirvio, S., Tihinen, M.: Great challenges and opportunities of distributed software development - an industrial survey. Fifteenth International Conference on Software Engineering and Knowledge Engineering (2003) 489–496
4. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* **1** (2004) 11–33
5. Rasmussen, J.: Risk management in a dynamic society: A modelling problem. *Safety Science*, Elsevier Science Ltd. **27** (1997) 183–213
6. Goguen, J., Linde, C.: Techniques for requirements elicitation. 1st IEEE International Symposium on Requirements Engineering, San Diego (1993) 152–164
7. Gotel, O.C.Z., Finkelstein, C.W.: An analysis of the requirements traceability problem. *International Conference on Requirements Engineering* (1994) 94–101
8. Sahraoui, A.E.K.: Requirements traceability issues: Generic model, methodology and formal basis. *International Journal of Information Technology and Decision Making* **4**(1) (2005) 59–80

9. Sommerville, I.: Software engineering (update) (8th edition). International Computer Science, Boston, MA, USA **1** (2006) 11–33
10. Guillermin, R., Demmou, H., Sadou, N.: System engineering approach for safety management of complex systems. Proceedings of European Modeling and simulation (ESM), Leicester, United Kingdom (2009)
11. Guillermin, R., Demmou, H., Sadou, N.: Combining fmeca and fault trees for declining safety requirements of complex systems. European Safety and Reliability Conference (ESREL), Troyes (France) (2011)
12. Lindsay, P.A., McDermid, J.A.: Derivation of safety requirements for an embedded control system. Engineering, Test and Evaluation Conference, Sydney (2002) 29–30
13. Buzzatto, J.: Failure mode, effects and criticality analysis (fmeca) use in the federal aviation administration (faa) reusable launch vehicle (rlv) licensing process. **2** (1999) 7.A.2–1 – 7.A.2–7 vol.2
14. Lee, W.S., Grosh, D.L., Tillman, F.A., Lie, C.H.: Fault tree analysis, methods, and applications - a review. IEEE Transactions on Reliability **1** (1985) 194–203
15. Villemeur, A.: Sûreté de fonctionnement des systèmes industriels. Paris : Edition Eyrolles (1988) 785
16. Sahraoui, A.E.K., Buede, D., Sage, A.: Issues in systems engineering research. INCOSE congress, Toulouse (2004)
17. EIA-632: Processes for engineering systems. Electronic Industries Alliance standard, January 7 (1999)
18. CEI-60812: Techniques d'analyse de la fiabilité des systèmes. (1995)

Proposition of a guide for investigating, modeling and analyzing system operating modes: OMAG

Vincent Chapurlat, Nicolas Daclin

Laboratoire de Génie Informatique et d'Ingénierie de Production (LGI2P) – ENS Mines d'Alès - Site de l'Ecole des Mines d'Alès - Parc scientifique Georges Besse, f30035 Nîmes cedex 5, France – tél. : (+33) 466 387 066 – {surname.name}@mines-ales.fr

Abstract - This paper presents and illustrates an approach that allows designers exploring and reasoning, checking and then arguing the consistency of the operating modes of a system. The goal is to help designers to build system's functional architecture by linking operating modes, allowed configurations and operational scenarios *i.e.* the set of functions displayed by the system in each mode in order to fulfill its mission taking into account its current configuration, requirements and environment. This approach is implemented as a guide called OMAG and is here illustrated on a vehicle design.

Keywords - System Engineering, System modeling, Operating Mode, Operational scenario, Verification

Introduction

System Engineering (SE) (INCOSE 2011) (SeBOK 2012) (Fiorèse *et al.* 2012) is a design approach approved and largely used in industry. Based on concepts and principles coming from system sciences *e.g.* (Félot 2007), SE promotes simultaneously a model based approach (INCOSE 2008) and a process oriented approach (ISO 2008) covering the whole cycle of a system design project. We consider here only technical activities related to architecture design (Sharman *et al.* 2004) (Blanchard *et al.* 2011). The goal is here to help designers' to make emerge potential alternative solutions of functional architecture. We propose for this to cover some of designer's modeling and verification needs:

- To find what are the relevant operating modes of the system considering its mission and the moving environment in which this mission has to be fulfilled.
- To become able to imagine how the system can evolve from an operating mode to another one when considering various events (external coming from environment as internal coming from the system itself *e.g.* dysfunctions) and system configurations.

- To model the expected behavior of the system when considered in each retained operating mode. It is here question to model various operational scenarios for each mode, each operational scenarios showing what are the requested

functions of the system in this mode and how these functions are then dynamically processed.

- To precise then what are the links between modes, configurations and scenarios allowing then to improve the coherence of the entire model of the system.

- To analyse the resulting behavior of the system as proposed in (Chapurlat 2012) by 1) checking modeling expectations in order to detect modeling errors or mistakes (*e.g.* unwanted deadlock or model consistence), and 2) cheking some functional requirements as far as possible detecting then some potential omissions.

Considering modeling needs, designers often use their experience, know-how, sometimes approaches based on creativity *e.g.* brain storming or mental representation. They can also use, when they are formalized, best practices, design patterns (Schindel 2005) or some guide such as GEMMA (French acronym of Guide d'Etude des Modes de Marches et d'Arrêts (ADEPA 1981)). This guide helps manufacturing systems designers to determine the control part. So it is proposed here to develop an approach and to implemnet it in a guide called OMAG (Oper- ating Modes Analysis Guide). OMAG promotes a graphical formalism facilitating its use by designers *e.g.* allowing them to select, decompose or refine an operating mode, a configuration or a scenario.

Considering analysis aspect, (Monin 2003) and (Grady 2007) propose using some formal approaches and particularly those focusing on properties proof (Da- guspta 2010) (Yahoda 2012). So a formal operational semantic¹ is proposed for OMAG and a technique based on property formalization and proof (Mallek et al. 2011) is used as described in (Chapurlat 2013a) (Chapurlat 2013b).

Last, this guide has to be fully interoperable with existing modeling languages and tools currently used in system engineering domain. This article presents the OMAG concepts and principles briefly illustrated on the case of a vehicle named VERECINT. Second, it presents the basics of the verification approach before concluding about perspectives and developments.

Modeling aspect

VERECINT is a vehicle allowing firemen and experts in the field of chemical, biological, radiological, and nuclear (CBRN) crisis to explore, to evaluate the different elements characterizing a crisis situation (data *e.g.* temperature or radiation level, information and expertise *e.g.* that follows the observation and the expertise of a phenomenon on the site), and to communicate them to crisis managers hav- ing to decide actions to take. VERECINT has then to be able to act accordingly to these actions. At this stage, we assume that VERECINT mission, purpose and ob- jectives as environment (other systems in interaction all along its life cycle and enabling systems), requirements, life cycle and all or part of requirements have been defined. These element cannot be detailed in this article.

¹ The set of principles and rules allowing a model (defined here as an instance of a modeling language) execution

OMAG principles and elements

OMAG (Operating Modes Analysis Guide) is a graphical guide proposing a set of pre-defined Operating Modes and of pre-defined Transitions between Operating Modes considered as generally relevant and helpful for various kinds of systems. Applied in VERECINT case, OMAG is diagrammed in Appendix. Let's notice for instance that VERECINT as any a system to be designed must 1) fulfill its mission for some nominal Operating Modes, 2) have to ensure either the continuity of service of this mission as much as possible in non-nominal Operating Modes, and 3) assume security of goods, people and its immediate environment. Then, before presenting briefly the requested elements of OMAG, let's reformulate its goals as follows:

- To allow a designer to select and choose what are the relevant Operating Modes and Transitions (evolution conditions and events) by taking into account a list of Operating Modes and Transitions having generally to be taken into account. This choice has to be made accordingly to the available knowledge about system definition, the current state of the set of requirements and about system environment.
- To determine gradually what are the possible, unavoidable or interesting configurations of the system and having then to be considered.
- To facilitate the modeling of operational scenarios relevant in each mode *i.e.* to guide the research of the requested functions of the system and the description of how they have to be dynamically associated to describe the expected behavior of the system.
- To allow designers to trace these choices, to change or modify a choice during design activities and to check, evaluate and compare alternatives solutions of functional architectures induced by these choices.

Considering a system S , OMAG (see Figure 1) requires to define first a set A of data from time, shape or space nature. These ones are chosen and can be evaluated or estimated in order to characterize the temporal aspect (time), the structural aspect (shape) and the situational aspect (space) of S and of its environment. They do not specify any candidate for architectural solution of S . They aim only to take into account as far as possible, for instance, dimensional constraints, specific attributes, expected delays, speed *i.e.* non functional requirements. A is initialized at the beginning of the design process and enriched bit by bit. Second, OMAG highlights three main Phases named respectively Deployment, Exploitation and End of life. A Phase is a set of Operating Modes of S that are logically linked or dependent. The Exploitation Phase is itself divided into Operating, Maintenance and Default sub phases.

An Operating Mode of S is a state reachable by S during its life cycle exhibiting then particular behaviors. By hypothesis, S is in one and only one Operating Mode called active Operating Mode (conversely, disabled) at each moment of its evolution. Each Operating Mode O of S is characterized by (E, C, OS_O, T) where:

- E is defined by a name and a set $S_E \subset A$. It aims to describe the environment of S , even partial or simplistic *i.e.* the context in which S has to be able to execute various functions when O is active. For instance VERECINT may evolve the night, under snow or rain, or on roads that may be damaged due to the crisis.

- The set C contains one or several Configurations relevant for S in O . A configuration c determines and refines the description of the state of S when it is in O . VERECINT can be for example described by configurations named ‘*exercice*’, ‘*operation on SEVESO site (site containing large quantities of dangerous substances)*’ or ‘*operation on a city*’. A configuration c is defined by a name and a set S_c A defining the requested data from A to describe the configuration. At least one Configuration is required for S , then considered as *default configuration* c_0 . Last, S can evolve from a Configuration c to another Configuration c' crossing a Transition as explained below.

Phases		Operational Modes
System deployment		D1 System is ready and waiting for deployment
		D2 Operational retirement
		D3 System functions for tests, maintenance, or training out of operational site
System exploitation	Operational	O1 System is deployed and operational on site, waiting
		O2 Preparing the system to assume its mission in nominal mode
		O3 System functions in nominal mode
		O4 Preparing the system to end normally its mission
		O5 System functions for tests, maintenance, or training on operational site
	Dysfunctioning / security	DS1 Stop after a default or dysfunction
		DS2 Diagnosis for default detection
		DS3 System functions in non-nominal mode
	Maintainance	M1 Diagnosis and Corrective Maintenance
		M2 Diagnosis and Preventive Maintenance
		M3 Diagnosis and Adaptative Maintenance
		M4 Diagnosis and Evolutive Maintenance
Cessation and dismantling		C1 Retract
		C2 Dismantling

Figure 1: Phases and Operating Modes of a system in OMAG

- The set OS_O contains one or several Operational Scenarios describing expected behavior of S when S is characterized by the current configuration. An Operational Scenario is defined by a functional model compose of a set of functions dynamically linked *i.e.* a part of S functional architecture describing how S must fulfill its mission. The behavior of VERECINT can be for instance specified by a scenario ‘*To observe and to measure specific values from crisis area*’.

- The set T contains one to several input and output Transitions. A Transition links a source object here an Operating Mode O (or a configuration C) to a target object *i.e.* an Operating Mode O' (or configuration C'). A input Transition describes how O' (resp. C') can be reached (O' or C' are then activated). Conversely, an output Transition describes how O or C are deactivated. A Transition is then formalized by a 6-uplet ($source, destination, c, e, d, [op]$) where $source$ and $destination$ describes respectively source object and destination object, c is the triggering condition (computed taking into account data from A), e is the initiating event (internal to S or coming from the environment) and d is the delay (null by default, but allowing to describe for instance duration of a requested reconfiguration time) under which S can evolve from the source to the target (from the same Phase or not). Last, a Operational Scenario op describing the behavior of S when it is reconfigured can be associated to the Transition.

OMAG: modeling principles

Establishing an OMAG model for a system S begins by defining a first version of set A i.e. by defining the set of space, shape and time data with approximate values. Then designer selects what are the appropriate Phases and Operating Modes. Indeed, those proposed by default (see Figure 1) can be selected by the designer or conversely may be rejected considering they are irrelevant regarding the mission, objectives, requirements, and context of S . If an Operating Mode O is selected:

- The set of Transitions allowing to activate O (input transitions T for which the source Operating Mode has been also selected) and to deactivate O (output transitions T' for which the destination Operating Mode has been also selected) are selected and defined by giving the 6-uplet (*source, destination, c,e,d,[op]*) which characterizes T .

- Configurations of S reachable in O and Operational Scenarios authored by these configurations can be determined. Then, transitions T' between Configurations have to be determined by determining the 6-uplet (*source, destination, c,e,d,[op]*) which characterizes T' where the firing event e can be linked to one of the proposed Operational Scenarios that can induce a modification of the current configuration.

Obviously, as OMAG, all the possible Configurations that can appear in the Operating Modes and the set of Transitions between Configurations forms a new state model that allows to decompose S from a different point of view, here by refining its possible Configurations whatever may be the Operating Mode on which S is.

VERECINT application

Applied to VERECINT system, selected Phases are:

- **Deployment:** the system is subjected to operations to ensure its storage and deployment onto a site on which its mission can begin. This phase is relevant for VERECINT which is involved in activities such as waiting, preparation, adaptation, training, exercise, or regulatory maintenance.

- **Exploitation:** the system is ready and deployed in operational conditions. This phase is relevant for VERECINT and means that it can be used by stakeholders, here firemen and experts having to explore and evaluate a crisis site. In order to avoid any interpretation, this phase is split up into three Families as follows:

Operating (O): the system is in nominal condition being able to fulfil its mission in coherence with specified requirements, hypothesis, and planned resources. In this Family, VERECINT fulfils efficiently its mission and exhibits nominal behaviours and configurations.

Maintenance (M): the system undergoes operations to restore the operating conditions due to anticipation, failure diagnosis, request for modification, adaptation, evolution, etc. In this Family, we choose to consider VERECINT in predictive or curative maintenance that may be done eventually on the operational site where the crisis occurs.

Default (DS): the system is in a safe state or degraded operation following order, failure, damage, or more generally to an internal or external interference that may cause damage. In this Family, VERECINT has to check default and to decide what maintenance is requested considering it must continue as possible to fulfil its mission eventually with loss of performance but a loss of security of users cannot be acceptable.

- **End of life:** the system is removed from service being concerned by decommissioning, partial reuse or reprocessing of part of all of its components and subsystems for possible future use. The feedback, uses and special cases of operational scenarios 'lived' by the system are finalized, indexed and stored to feed the design of possible future releases. This phase is also selected for VERECINT.

The possible Operating Modes of S and those that are retained by a designer for VERECINT are then the followings:

- **D1 - System is ready and waiting for deployment:** VERECINT is available but stopped and possibly stored out of operational site, ready and packed to be deployed on site and then exploitable by stakeholders. By hypothesis, D1 is defined as the initial Operating Mode of the studied system (graphically denoted by a box with large borders in Appendix) here VERECINT.

- **D2 - Operational retirement:** VERECINT has to be removed from the site and repackaged or prepared to be redeployed afresh on another site.

- **D3 - System functions for tests, maintenance, or training out of operational site:** VERECINT, although not deployed, is operating, possibly in a degraded or reduced testing environment for functional tests, training, or regulatory service beyond operational site.

- **O1 - S is deployed and operational on site:** VERECINT is operational on site, ready to fulfill its mission in identified operating environments.

- **O2 - Preparing the system to assume its mission in nominal mode:** VERECINT requires preparation before it can fully perform its operational mission on site (*e.g.* preheat, audit checklist usage, etc.).

- **O3 - S functions in nominal mode:** VERECINT fulfill its mission maximizing its performance on specified operating environment.

- **O4 - Preparing S to end normally its mission:** VERECINT requires to be prepared before stopping its mission normally so various Operational Scenarios can be expected in O4 for VERECINT *e.g.* cleaning, decontaminating, etc.

- **O5 - S functions for tests, regulatory maintenance, or training on operational site:** VERECINT functions, possibly in a degraded or reduced functional coverage for testing, training, regulatory maintenance on the operational site where VERECINT is currently deployed.

- **DS1 - Stop after a default or a dysfunction:** VERECINT has to be put in safety due to an internal dysfunction or a default detected which threatens its own integrity and safety or the integrity and safety of the environment.

- **DS2 - Diagnosis for default or failure detection:** VERECINT is submitted to tests and procedures (led by itself or by one or more contributors systems) of assessment and diagnosis of failures of its functions and components.

- **DS3 - S functions in non-nominal mode:** VERECINT suffers the consequences of an internal failure or external events affecting its operational capabilities but continues to fulfill its mission staying in a range of acceptable values in terms of risk, performance or respect of some chosen non-functional characteristics - safety, security, survivability, maintainability, interoperability, ... called "*ilities*" (Weck *et al.* 2012.).

- **M1 - Diagnosis and corrective maintenance:** VERECINT has to undergo operations permitting to restore a specified configuration so that it is able to ensure its operational mission again.

- **M2 - Diagnosis and Preventive Maintenance:** VERECINT has to undergo operations for the replacement, revision or repair of one or more of its components before the dreaded occurrence of a default respecting a maintenance plan.

- **M3 - Diagnosis and adaptive maintenance:** The system has to undergo operations to adapt, for example due to the possibility of using new technologies to better fulfill its initial operational mission. This includes reengineering activities. This Operating Mode is not relevant for VERECINT so M3 and each input and output Transition of M3 are the graphically identified by a red line has diagrammed in Appendix.

- **M4 - Diagnosis and evolutionary maintenance:** The system has to undergo operations to make it evolve, for example due to the possibility to extent its functional coverage to respond to new requirements or modify its initial operational mission. This includes also reengineering activities. As for M3, this Operating Mode is not relevant for VERECINT.

- **F1 - Retirement:** VERECINT has to be taken out of service permanently.

- **F2 - Dismantling:** VERECINT has to be dismantled and its various components and subsystems may be stored, packaged or stored for reuse, conversion, re-processing.

A set of generic Transitions between Operating Modes is given in Figure 2. These transitions are quoted as follows: Classic Transitions (T_i , $i = 1$ to 20), Stop (S_i , $i = 1$ to 3), Application Maintenance (AM_j , $j = 1$ to 4) End of Maintenance (EM_j , $j = 1$ or 2) or Fault Detection Security (FDS). Only input and output Transitions of selected Operating Modes have to be specified by the designers. For more information, the reader can find all conditions and events having to be specified for each selectable Transition in OMAG in (Chapurlat *et al.* 2013c).

	D1	D2	D3	O1	O2	O3	O4	O5	DS1	DS2	DS3	M1	M2	M3	M4	C1	C2
D1		S2	T2	T4					S1			AM1	AM2	AM3	AM4	S3	
D2	T1								S1			AM1	AM2	AM3	AM4	S3	
D3	T3	S2							S1							S3	
O1		S2			T7	T5		T11	S1	FDS						S3	
O2		S2				T8			S1	FDS						S3	
O3		S2		T6			T9		S1	FDS						S3	
O4		S2		T10					S1	FDS						S3	
O5		S2		T12					S1	FDS						S3	
DS1		S2								T15						S3	
DS2		S2							S1			T16	T17	T18	T19	S3	
DS3		S2							T14	T13						S3	
M1	EM2	S2		EM1					S1							S3	
M2	EM2	S2		EM1					S1							S3	
M3	EM2	S2		EM1					S1							S3	
M4	EM2	S2		EM1					S1							S3	
C1																	T20
C2																	

Figure 2: Selectable transitions between Operating Modes in OMAG

Verification aspect: OMAG semantic

The operational semantic of OMAG is formalized for two reasons. First, it allows describing without ambiguity how a model OMAG can be interpreted and executed and then to define and implement OMAG simulation mechanisms. Second, it allows to formalize what are then expected modeling properties (Chapurlat 2013b) that must be satisfied in order to help designers to improve the quality of OMAG *e.g.* absence of modeling errors, but not its relevance or adequation with the modeled system. In this case, OMAG transformation rules are proposed in order to transform an OMAG model into a formalism authorizing proof of a-temporal and temporal properties as proposed for instance in (Mallek *et al.* 2012). In this case, techniques are based on the use of Conceptual Graphs for a-temporal properties and on Model Checking techniques for temporal properties. The reader interested by these two complementary techniques can find definitions and illustrations in the referenced articles.

By definition, OMAG is conform to the Interpreted Sequential Machine (ISM) described in (Larnac *et al.* 1999) and is then an extension of a State Machine. Operating Modes and Configurations are formalized by *states* and Transitions are described as conditioned *transitions* between *states*. By hypothesis, an Operational Mode or a Configuration can be decomposed giving then a new ISM. The ISM operational semantic is then enriched by model decomposition rules as proposed by (Harel 1987) for Statecharts. This semantic can be summarized in the next. There always exists an initial Operating Mode *i.e.* initial state for each level of decomposition and the next hypothesis of behavioral determinism are required:

- For each moment in the evolution of the OMAG, the same input vector applied to the same active state S induce always the same resulting output vector and the same next reached state S' .
- A transition T is triggered at a null time *i.e.* there is no potential event e (internal as well as external) that can be omitted during triggering of T .
- For a given state S , the evolution condition associated to output transitions of S are exclusive *i.e.* only one transition T can be triggered at each moment.

Thus, the transition triggering is done in two stages. If the condition c is true and the trigger event e appears (always occurring by default if it is not specified), then the Operating Mode or the Configuration is deactivated. This induces eventually to be able to stop current Operational Scenarios that are associated with this one. It can also require the execution of the so called *reconfiguration operational scenario* op which describes what the required functions are allowing S going from the current Operating Mode or the current Configuration to the next one. After the delay d (equal to null by default), the targeted Operating Mode or Configuration is activated. In the case of an Operating Mode, a default configuration is defined and then, is activated. In the case of the activation of a Configuration, authorized Operational Scenarios are launched and executed. Conversely, any Operational Scenario may induce a modification of Configuration, possibly causing the trigger a new transition between this configuration and the next one.

Conclusion

An abstract and a concrete syntax are under development by using DIAGRAPH tool box (Pfister *et al.* 2012) under Eclipse modeling framework. It provides a graphical user interface allowing to handle operating modes, configurations and operational scenarios. By hypothesis, eFFBD (enhanced Functional Flows Block Diagram) (DoD 2001) modeling language is here adopted to describe operational scenarios. So, a first perspective consists to use operational semantic of eFFBD proposed by (Seidner 2006) for synchronizing OMAG and Operational Scenarios evolution. Last, the modeling tool will aim to be interoperable with various SE tools. For this, transformation rules and mechanisms are studied by using ATL (ATL 2006).

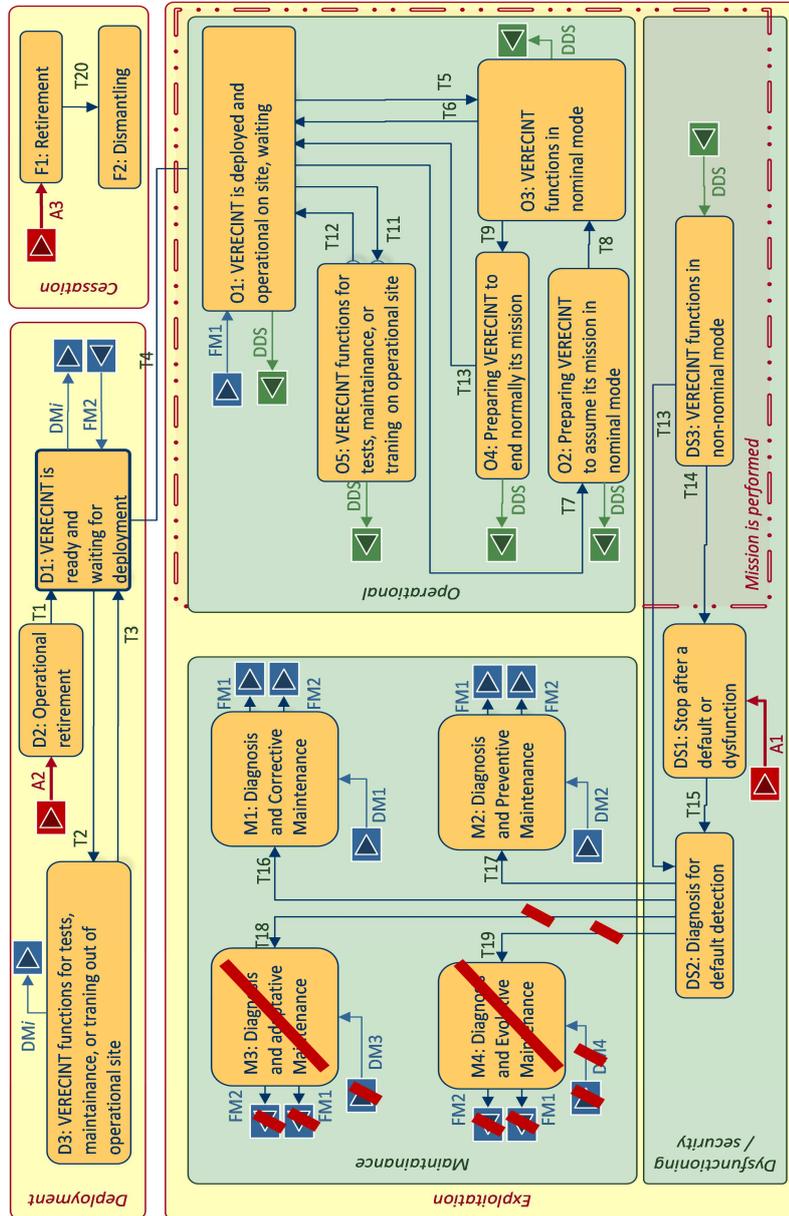
Verification techniques have now to be adapted and tested on complex examples. The perspective is to enrich these two techniques (Conceptual Graphs and model checking) by considering extensions proposed by (Thierry-Mieg *et al.* 2004) allowing to gain in performance and relevance when facing problematic of growing up models' size and complexity (*e.g.* due to number of refinements levels) to be analyzed during verification activities.

References

- ISO/IEC 15288:2008(E) / IEEE Standards 15288.2008 – Systems engineering – System life cycle processes (2nd edition), February 2008
- INCOSE, System Engineering (SE) Handbook Working Group, System Engineering Handbook, A Guide For System Life Cycle Processes And Activities Version 3.2.1, INCOSE TP 2003 002 03.2., 2011
- BKCASE Project, System Engineering Book of Knowledge, SEBoK v1.0, <http://www.sebokwiki.org/> (last visit 2013-04), 2012
- S.Fiorèse, J.P.Meinadier, Découvrir et Comprendre l'Ingénierie Système, CEPADUES Editions, ISBN 978.2.36493.005.6, May 2012
- C.Félot, Toward a formal theory of systems, Colloque d'Automne du LIX 2007 - CAL07 Complex Systems: Modelling, Verification and Optimization, Paris, Carré des Sciences, <http://www.lix.polytechnique.fr/~liberti/cal07/presentations/> (last visit 2012-01)
- INCOSE, Survey of Model-Based Systems Engineering (MBSE) Methodologies, Model Based Systems Engineering (MBSE) Initiative from International Council on Systems Engineering (INCOSE), 10 June 2008
- B.S.Blanchard, W.J.Fabricky, Systems Engineering and analysis, Prentice Hall International Series in industrial and systems engineering, fifth Edition, Pearson Coll., 2011
- D.Sharman, A.Yassine, Characterizing Complex Products Architectures, Systems Engineering, 7(1), pp. 35-60, 2004
- B. Schindel, Pattern-Based Systems Engineering: An Extension of Model-Based SE, INCOSE 2005, TIES 4

- ADEPA, Guide d'Etude des Modes de Marches et d'Arrêts, <http://www.technologuepro.com/cours-automate-programmable-industriel/GEMMA/Gemma-original.pdf> (last visit 2012-12), 1981 [*in French*]
- J-F. Monin, Understanding Formal Methods, Springer-Verlag, ISBN: 1-85233-247-6, 2003
- J.O.Grady, System Verification: proving the design solutions satisfies the requirements, Academic Press, Elsevier editor, ISBN: 978-0-12-374014-4, 2007
- P.Dasgupta, A roadmap for formal property verification, Springer, ISBN: 978-90-481-7185-9, 2010
- Yahoda, formal verification tools overview web site, <http://http://anna.fi.muni.cz/yahoda/> (last visit 2012-01)
- S.Mallek, N.Daclin N., V.Chapurlat, An Approach for Interoperability Requirements Specification and Verification. The International IFIP Working Conference on Enterprise Interoperability, Stockholm, Sweden, 2011
- V.Chapurlat, (2013a) UPSL-SE: A Model Verification Framework for Systems Engineering, Computers in Industry, Computers in Industry 64 (2013), pp. 581–597
- V.Chapurlat (2013b), Property concept and modeling languages for Model-Based Systems Engineering (MBSE) context, Internal Research Report (access on demand),
- M. Larnac, J. Magnier, V. Chapurlat, B. Chenot, Extended Temporal Proof of Properties using the Interpreted Sequential Machine Model, proc. of International Congress IEEE SMC'99, Tokyo, Japan, 1999, pp I-974/I-979
- V.Chapurlat (2013c), Proposition d'un Guide d'Etude des Modes Opérationnels des Systèmes (GEMOS) pour l'Ingénierie Système, Conférence Francophone de Génie Industriel, La Rochelle, June 12st -14st, 2013 [*in French*]
- D.Harel, Statecharts: a visual formalism for complex systems, Science of computer programming, 8, 1987, pp 231-274
- C.Seidner, Vérification des EFFBDs: Model checking en Ingénierie Système, Thèse de Doctorat Université de Nantes, 3 Novembre 2009 [*in French*]
- O.L. de Weck, A.M.Ross, D.H. Rhodes, Investigating Relationships and Semantic Sets amongst System Lifecycle Properties (-ilities), third International Engineering Systems Symposium CESUN 2012, Delft University of Technology, 18-20 June 2012
- F.Pfister, V.Chapurlat, M Huchard, C.Nebut, A proposed tool and process to design domain specific modeling languages (pp. 31-35), http://www.lgi2p.ema.fr/~urtado/Slides/RR12_Lab_002.pdf (last visit 2012-09)
- DoD, Systems Engineering Fundamentals. Defense Acquisition University Press, http://www.dau.mil/pubscats/PubsCats/SEFGuide_2001-01.pdf (last visit 07/07/2012), 2001
- ATL: Atlas Transformation Language - ATL User Manual, version 0.7, ATLAS group LINA & INRIA, Nantes, 2006
- S.Mallek, N.Daclin, V.Chapurlat, The application of interoperability requirement specification and verification to collaborative processes in industry Computers in Industry, n°63 (2012) 643–658, 2012
- Y. Thierry-Mieg, S. Barrir, A. Duret-Lutz, F. Kordon, Nouvelles techniques de Model Checking pour la vérification de systèmes complexes, Génie Logiciel, 69, pages 17-23, 2004 [*in French*]

Appendix: OMAG graphical representation for VERECINExample



Systems engineering and modeling: some epistemological remarks

Fabio Roda

LIX, École Polytechnique, 91128 Palaiseau, France.
roda@lix.polytechnique.fr

Abstract. In this paper we provide some epistemological and historical remarks that concern systems engineering and modeling.

1 Introduction: Philosophy of Engineering

In this paper we provide some epistemological and historical remarks that concern systems engineering and modeling. This is consistent with the idea that science and co-science (i.e. philosophy of science) can cooperate fruitfully. However, we think that it is confusing to mix them vaguely and that we have to separate real applications and the consideration about their philosophical implications. Thus, using a terminology that philosophers love, this work belongs to the meta-level.

2 Philosophy of Engineering

The match between philosophy and engineering is quite unusual and merits further clarifications. The basic issue in the philosophy of science can be introduced as follows: scientists study the world, philosophers of science study how they do that (and sometimes they also study scientists themselves).

Borrowing from Lipton [1],

“I am a philosopher of science: what do I do? Here is the short version: astronomers study the galaxies; I study the Astronomers.”

There has been a certain disregard by philosophers of science towards technology, which they consider a straightforward application of pure sciences.

“The method and the theories of science can be applied either to increasing our knowledge of the external and the internal reality or to enhancing our welfare and power. If the goal is purely cognitive, pure science is obtained; if primarily practical, applied science. Thus, whereas cytology is a branch of pure science, cancer research is one of applied research.” [2]

This idea, i.e. “technology is applied science”, hides the fact that technology and engineering disciplines have some features needing a special epistemological investigation. Recently, a new branch of epistemology called *philosophy of engineering* has attracted increasing interest: it is concerned with the clarification of the epistemological role

of technology and engineering within science and human knowledge. To continue the analogy introduced above, engineers study how to design systems, philosophers of engineering study how they do that. The crucial difference between engineers and scientists is that the former *decide* how to manufacture or produce working artifacts and systems, while the latter *analyze* nature and formulate theories to explain how natural systems work. Decision-making naturally brings engineers to think about objectives much more than natural scientists need to do. This profiles a different kind of rationality. On the one hand, we might believe that models are simplified versions of reality that exists independently from our ideas about it, and that the task of science is to describe this reality. On the other hand we might think that models do not describe reality, but they actually create it (indeed most of the modern epistemology tells us that all observation is theory-laden, for example see Hanson [3] on this point and, more philosophically, think of Kant and his *Copernican revolution*). Likewise, we might feel a need to simply explain systems, as opposed to endowing them with aims. If truth is not discovered, but it is invented, the hierarchy between science and technology is inverted.

“Despite the more than two millennia that separate Aristotle’s thinking from ours, Aristotle’s conception [sets] the agenda for almost all subsequent thinking about explanation. [...] The rivalry had been between those who thought that all causal explanation must proceed in terms of efficient causation and those who (following closely on Aristotle’s footsteps) thought that there is room (and need for) teleological explanation (that is, for explanation that cites final causes). [...] Aristotle saw goals and purposes in nature, mechanical philosophers either excised all purpose from nature (Hobbes, Hume) or placed it firmly in the hands of God (Descartes)”. [4]

This debate fails to have a clear outcome within epistemology, but if the target system is artificial rather than natural, then it must have a goal, and the issue becomes clearer. What we might call the “pure problem” of scientists is “is it true?”, while that of engineers might either be “does it work?”, or, perhaps more appropriately, “does it do what the stakeholders want?” It is clear that there is a relationship between being able to verify a statement and making a choice. However, decision making and systems design have some features that make them a special case from an epistemological point of view.

“In engineering the ultimate purpose of modeling is to realize reliable artifacts or technical processes. This contrasts substantially with the natural sciences where, conceptually at least, the aim underlying the modeling activities is to gain knowledge for knowledge’s sake.” [5]

Epistemologists, in the first half of the '900, usually made reference to natural sciences as chemistry, biology and, most of all, physics (probably due to its resounding success). In this case, the observer is in front of a system that is given and he/she has to describe and understand it. However, in engineering the system is actually *built* by the observer (or one of his/her fellow humans). Thus, the *demarcation criterion* of natural science may be not perfectly suitable. Systems designers still have to do verifications and observations (as natural scientists) but most of all they have to make choices. They

are interested in the truth of statements as much as in the effectiveness of choices. From the point of view of systems design, good models are the ones that help to split properly the domain of possible choices in good and bad ones. Some epistemologists underline the problem-solving aspect of science, for example Laudan.

“Science is essentially a problem-solving activity. [...] The approach taken here is not meant to imply that science is “nothing but” a problem-solving activity. Science has a wide variety of aims [...] My approach, however, contends that a view of science as a problem-solving system holds out more hope of capturing what is most characteristic about science than any alternative framework has.” [6]

Considering science as problem-solving corresponds to a change of perspective since we are more interested in getting local solutions rather than global theories. In particular, Khun suggested Operations Research as a good example of the problem-solving approach to science.

“For Kuhn, science is problem-solving rather than truth-seeking activity And what would be a more striking example of problem-solving than OR! . . . As a problem-solving activity OR is oriented towards practice: it tries to use the methods of science to find optimal solutions to problems concerned with alternative courses of actions. As the solutions are its primary aim, it is clear in which sense OR is not a truth-seeking activity: it is not a knowledge-seeking enterprise.”[7]

Philosophy of engineering focus of these special aspects of applied science. We adopt the same perspective. It has been said that “*Philosophy of science is about as useful to scientists as ornithology is to birds*”¹, namely that it is not very useful in practice, but we try to show that some epistemological issues arise anyhow. In our opinion, they require a consideration. At least, epistemology is useful for an external analysis of the scientific method. A scientific analysis of the scientific method would be self-referential.

Nevertheless, no-one is better placed than a scientist or an engineer to understand and analyze his or her own way of working. With the words of Schlick,

“A philosopher, therefore, who knew nothing except philosophy would be a knife without blade and handle. Nowadays a professor of philosophy very often is a man who is not able to make anything clearer, that means he does not really philosophize at all, he just talks about philosophy or writes a book about it. This will be impossible in the future. The result of philosophizing will be that no more book will be written about philosophy, but all books will be written in a philosophical manner.” [8]

2.1 Epistemic vs non-epistemic values

McMullin [9] introduces a distinction between *epistemic* and *non-epistemic* values, that is relevant in epistemology. He proposes that a value is epistemic if it helps to “*promote*

¹ Richard P. Feynman

the truth-like character of science". Otherwise, it is non-epistemic. Dorato [10] confirms that we can use the term *epistemic* for values "*regarded as capable of furthering our knowledge*" and *non-epistemic* to refer essentially to values that are ideological, economical, political, ethical, environmental, esthetic or religious. Non-epistemic values can influence science, indirectly. They influence, for example, the choice of the destination of economic endorsement of research projects. Nevertheless, there is a strong agreement, in the scientific community, on the idea that non-epistemic values have no role in determining scientific truth. Non-epistemic values influence the use of the results of pure science, but are never (or hardly never) integrated in the content of scientific theories.

However, for engineering and technology disciplines the role of non-epistemic values appears to be less clear. Safety, equity and economical sustainability are examples of non-epistemic values (since they do not produce knowledge) that have an important role in the decision making process concerning real systems. Engineers, who have to choose between two or more alternative models, in some cases, have to consider non-epistemic values, and integrate them in their models. It is the case of the systems we have considered in this work.

This leads us to think about the way a model can gain a justification when it does not rely upon pure epistemic values. In fact, a first possibility is the experimental approach. We "try and observe". This is not unusual. A second possibility is the collaborative methodology. Another one deals with ethical values. This is not totally common. Thus, in the next sections, we present the tradition that is behind each one of these approaches. However, preliminarily, we present some remarks about the concept of *model*.

3 Models

The root of the term *model* can be traced back to the Latin term *modus* which in turn would derive from the Indo-European root "med-". Its meaning is measure [11]. *Modus* has two diminutives *modellus* and *modulus* which we find in different contexts linked to engineering related disciplines. The roman architect Vitruvius uses *modulus* to mean architectural standard, which is a surprisingly modern use of the term. Tertullianus uses *modulus* to indicate basis for a marble sculpture. In the period which spans from the Roman Empire to the Middle Ages, terms derived from *modulus* spread across Europe and we detect the terms *modle*, *mole* and *moule*, which came into English as *mould*. Modern English also introduced directly the term *module* from Latin. During the italian renaissance *modelo* and *modello* are employed by important architects, such as Brunelleschi, who uses it while building the cupola of the dome of Firenze, and Alberti:

"Be sure to have a complete Model of the Whole, by which examine every minute Part of your future Structure eight, nine, ten Times over, and again, after different Intermissions of Times". [12]

From the Italian *modello* derives the French *modèle* and the English *model* and *modell*. Shakespeare uses *model* both with reference to buildings, thus in the architectural sense, and in a more general sense as "kind of behavior" and Bacon indicates with *modulus* a mental copy of the real world, which is quite close to the modern use. Nowadays these

terms are intensively diffused. For example, during the decade 1990-1999 there have been 17,000 publications including them in the title.

The remarkable point is that along centuries there is an interesting feature which characterizes models: they appear to be tools which help to design artifacts. Models are visions of a target system constructed respecting constraints drawn from its environment, which help the system designer/architect to conceive it. In engineering disciplines, modeling is first of all an activity that is close to design. The designing of systems and services requires both analytical and synthetic processes, because designers invent and create new *artificial* systems to fulfill a need. This is different from describing and understanding a given *natural* system. From this point of view modeling assumes a meaning which is much more practical with reference to other scientific disciplines as natural sciences, formal logic and mathematics. Modeling is a set of activities, tools, heuristics (in the broad sense of the term), capabilities which lead a designer to build system-answer to a problem-question which he/she is confronted to.

3.1 Model validation

“The mathematical models that are used in OR are representations of the system under study. These models may be imperfect and idealized, but still the quality of the solutions that they yield crucially depends upon their closeness to reality in the relevant respects.” [7]

Engineers separate the “judgment” of a system into two distinct phases, *verification* and *validation*. The verification process guarantees that the system has been realized correctly, respecting all the specifications documented during the phase of requirements engineering. The validation process ensures that the system functions as expected. Notice that, from an end-user perspective, a system which performs perfectly a wrong task is not a good outcome. This issue is very important in systems design.

“Simply put, the Product Verification Process answers the critical question - Was the end product realized right? The Product Validation Process addresses the equally critical question - Was the right end product realized?” [13]

This issue concerns also the method of OR, which typically includes two phases: in the first phase a problem is formalized into a model; in the second phase efficient techniques are searched in order to solve the model. Model verification deals with questions about the capacity of providing correct solutions with a limited amount of computational resources and time. We refer to this issue as the problem of *efficiency*. Model validation assesses that the model really addresses the right problem. We refer to this issue as the problem of *effectiveness*. For example, a model for the shortest path problem and a fast and correct algorithm that finds its solutions would not be a good answer for someone who is looking for paths that go through the “top n ” interesting cities starting from Milan and arriving in Paris. It would be efficient but not effective. In OR several important problems are already accurately identified and classified, therefore the focus is most of all on the capacity of solving them, i.e. efficiency. The problem of efficiency is well defined. Computational complexity theory deals with it and provides a stable framework.

However, engineers and system designers are often puzzled by the problem of writing the right model. In systems design effectiveness is a major issue.

“- What is a valid model? - has been one of the least discussed topics in the OR literature. [...] Thinking about model construction and model validation is basically to raise the issue of different ways of producing knowledge and deciding about the acceptability of the knowledge thus produced”.[14]

The problem of effectiveness encompasses several approaches and has blurred boundaries. Validation tests can be based on comparing model predictions to real world results. However this kind of validation is not always possible because repeated tests can be expensive, time-consuming or simply impossible. Thus, alternatively, models can be validated using historical events and inter-subjective arguments. In our opinion, the problem of model validation in OR can not be separated from general issues about the approach to scientific knowledge. We believe that philosophy of science and in particular philosophy of engineering are good frameworks for the problem of effectiveness. A few authors share this opinion with us.

“Whether Operational Researchers are aware of it or not does not make any difference: to take an option in the debate on model validation in OR is, explicitly or not, to actualize epistemological choices”. [15]

4 Experimental approach to model validation

Modern science is empirical. Experimentation has a role in science which can not be underestimated. According to R.P. Feynman:

“The principle of science, the definition, almost, is the following: The test of all knowledge is experiment. Experiment is the sole judge of scientific truth”
[16]

Nevertheless, in this section we provide some arguments to remind that the debates that emerged in contemporary epistemology show that the role of experimentation is (sometimes) considered as troublesome. There are a bright and a dark side of the coin. We start from the bright side.

First of all, experiments are used to produce a *confirmation*, as they can give us strong arguments to trust a hypothesis. Secondly they can favor the *discovery* of new theories showing new unknown phenomena which call for an explication. As representatives of these two uses of experiment, we can cite, among others, G. Galileo and F. Bacon. Both of them championed a more empirical attitude in natural philosophy and both of them supported a new vision of knowledge based on observations that had to be performed without prejudice or preconception. However, we consider Galileo to exhibit an example of the use of experimentation to confirm a theory and Bacon as an example of use of experimentation to favour the discovery of new theories.

Observations can endorse a theory. With the telescope, Galileo discovered the four large moons of Jupiter, which, since they do not orbit Earth, provide an argument against

the Ptolemaic theory that fixed it at the center of the universe. In this case, facts obtained through experimental work (repeated observation) confirm a theory (Copernican system).

Observations can foster new, general ideas, as explained by Bacon. In fact, Bacon was a convinced inductivist. His *Novum Organum* (1620) can be considered as the first modern work on inductive logic. In particular, it analyses the methods that can be used to produce theoretical inductive inferences, namely from particular to general, which had been relegated to a minor role during the previous centuries.

“The syllogism consists of propositions, propositions consist of words, and words are tokens for notions. Hence if the notions themselves (this is the basis of the matter) are confused and abstracted from things without care, there is nothing sound in what is built on them. The only hope is true induction.”

More recently, the more radical defense of empiricism is reasserted by the logical empiricists of Vienna Circle²: who stated, in their *Manifesto*, that true knowledge is totally empirical because the scientific enterprise is characterized

“essentially by two features. First it is empiricist and positivist: there is knowledge only from experience [...] Second, the scientific world-conception is marked by the application of a certain method, namely logical analysis.”

One of their most famous thesis is the *verification criterion of meaning*: the meaning of a proposition consists in its method of verification, and a proposition which cannot be verified is meaningless. Thus, the role of experimental verification is even stronger than in the vision of Galileo and Bacon, since it is at the basis of meaning.

We now take a look at the dark side of the experimentation coin. Duhem [17] proposes that it is not possible to test experimentally a single hypothesis because complex theories includes many hypotheses and it is really hard to establish which statements are contradicted by a test (systems engineers would call this a traceability problem). Moreover, an observation that refutes a model can be compatible with many other ones. For example, the observation of Galileo was consistent with both the models proposed by Copernicus and the one proposed by Tycho Brahe. This position is known, nowadays, as Duhem-Thesis³.

A second difficulty concerns the trustworthiness of what we are used to consider *objective facts*. Starting from the platonic *allegory of the cave* up to now, several philosophers have warned about the possibility that facts could be illusory. Many times in the history of philosophy evidence has been called into question. However, in this case, the target is not knowledge in general, it is the exactly the scientific method which is questioned. In the context of modern science a common reference, from this point of view, is the work of Hanson, as mentioned above. Hanson believes that there is not unconditioned observation of facts and, moreover, there is not a neutral language to

² The Vienna Circle was an association of philosophers centered at the University of Vienna in 1922. Among its members there were Moritz Schlick, Rudolf Carnap, Richard von Mises, Otto Neurath, Herbert Feigl.

³ We remark the often the terms Duhem-Thesis and *Duhem-Quine Thesis* are used as equivalent, but, in reality they refer to quite different thesis.

express them. Observational terms are “full of theory”. Thus the idea that theories are confronted to pure facts is wrong, in his opinion.

“There is a sense, then, in which seeing is a ‘theory-laden’ undertaking. Observation of x is shaped by prior knowledge of x . Another influence on observations rests in the language or notation used to express what we know, and without which there would be little we could recognize as knowledge.” [3]

What we observe is influenced, from the beginning, by our system of reference, our opinions, our background knowledge and, in general, our theory.

A third difficulty is explained by Hempel. He proposed the so-called paradox of confirmation, which he explains through the example of the ravens. We normally admit that the observation of a black raven confirms the hypothesis that “all ravens are black”. On the other hand, a white raven is a clear counterexample. However if we also admit (and in general we do) the *equivalence condition*, then we get strange results. The *equivalence condition* states that if two hypothesis are logically equivalent, then certain evidence that confirms the first one confirms also the second (equivalent) one. A logical equivalent of “all ravens are black” is “all non-black objects are non-ravens”. This last is confirmed by a non-black non-raven, e.g. a white tie. It follows that a white tie also confirms “all ravens are black”. This is logically correct, but it sounds strange.

We know that Popper proposes a fundamental improvement to the verification principle of Vienna Circle. He believes that inductive inferences have no justification, since no matter how many singular facts you have observed, you are never sure that a different singular phenomenon could occur, making your general conclusion wrong. Thus verification is, in practice, not feasible. He introduces a different criterion to defend the possibility of empirical justification of a theory. A theory has to divide the world into two distinct classes of phenomena: the ones that are compatible with it and the ones that contradict it. Thus, we should not look for facts that confirm a theory, but for the ones that could make it false. The longest a theory resists to these assaults, the better. It is trusted, or, using his terminology, *corroborated*. This is a considerable progress with reference to the positions of Vienna Circle. Problems caused by induction are reduced.

Nevertheless, according to his opponents, the falsification method proposed by Popper does not escape to the issues of theories underdetermination. During the sixties, authors like Kuhn and Lakatos promoted the idea that science progresses through many different ways, making our comprehension of its method more encompassing. Their focus was no more on one single theory against facts. Scientific research started to be considered as a complex system that comprehends many heterogeneous elements. The terms *paradigm* proposed by Kuhn and *research program* proposed by Lakatos gained a remarkable success and entered the terminology of philosophy of science, becoming quite common. In particular (following [18, 19]) there are 4 types of basic research programs: *descriptive*, *explanatory*, *design*, *explicative*. Descriptive research programs aim “simply” to describe of a set of phenomena, while explanatory programs try to provide an explanation and a framework to predict similar phenomena. These first two types concern empirical sciences. Design research programs deal with the realization of artifacts that fulfill certain previously chosen needs. This type concerns engineering and related disciplines. Explicative research programs are meant to provide precise, possibly formal explication of interesting, but unclear concepts. This last type regards

mathematics and analytic philosophy. Thus, there are at least four different approaches to science, and not all of them are purely based on experimentation. The “lesson” of these philosophers of science is that we should consider the method of science simply as “what scientists do”, without limitations. Feyerabend, most of all, strongly endorses this point of view.

From our point of view, we notice that, actually, system designers and decision makers (sometime) have to make choices that can not be based on experimental evidence. Therefore, in the following sections, we consider different possible approaches.

5 Collaborative approach to model validation

In this section we trace historical and conceptual roots of this kind of method, namely the search of truth (only) through an open discussion.

There are approaches to the scientific knowledge that skip most of the issues about the capacity of science of catching the ultimate truth about reality. For example, *instrumentalism*.

“Instrumentalism can be formulated as the thesis that scientific theories, the theories of the so-called “pure” sciences, are nothing but computational rules (or inference rules)”. [20]

Ontological⁴ problems about the effective existence of an immutable “being”, that has to be described by a conclusive explanation, are totally left out. Instrumentalism does not focus on the distinction between truthfulness and falseness of scientific theories. On the contrary it considers, by choice, “only” their practical utility. Important representatives of this approach are, among others, E. Mach, H. Poincarè, P. Duhem, E. Le Roy. For example, Poincarè proposes that we can consider the axioms of the geometry as simple *conventions*. Similarly, Le Roy thinks that science has a pure instrumental value and that scientific laws are only convenient synthesis of sets of facts. The position of Duhem is more variegated, but not very different.

“A physical theory is not an explanation. It is a system of mathematical propositions which can be derived from a small number of principles that serve to precisely depict a coherent group of experimental laws in a both simple and complete way”. [21]

The “second”⁵ Wittgenstein (see.[22]) believes that a general formal study of the language is not viable. No theory can provide general rules that are valid in all cases. On the contrary, we can establish only local norms since human language is elaborated in local contexts. He thinks that these norms emerge from behaviors and cultures based on what he calls *language games*, i.e. specific sets of linguistic rules. A perfect language does not exist and in particular there is not a perfect scientific language. Moreover, in his opinion, this reflects the absence of a common underlying structure, namely the

⁴ Ontology is the branch of metaphysics that studies the nature of existence or being as such

⁵ We remark that the “second” Wittgenstein is almost different from the “first” one, whose positions are represented most of all by the *Tractatus logico-philosophicus*.

absence of a common logic. We should drop the idea that there is one single “Logic” at the basis of human rationality and accept the fact that we act and think according to particular *practices* which are functional to particular aims and can not be generalized.

Instrumentalism, conventionalism and the “second” Wittgenstein open the door to the entrance in the field of philosophy of science of elements that, in the first decades of the 20th century, had been kept out. Social components are introduced as a fundamental part of scientific knowledge. The separation between external and internal components of scientific enterprise starts weakening, so that context and content begin running into one and knowledge is no more *justified true belief*, but, more weakly, *locally accepted belief*. Physics loses its supremacy as model of all scientific disciplines, and the nineteenth-century idea, renewed by the project of unity of science of Vienna Circle, that all branches of science could be reduced to mathematical explanation, is replaced by a more encompassing approach that admits final causes, interpretations, narrative explanations. From the point of view of these authors, the study of nature is similar to the study of social institutions, myths, political groups. In other words, these epistemologists think that knowledge is only a social construction, namely that truth does not exist in itself and it is only agreed consensus (often, of experts). This current of thought suggests that what we consider true is composed by simple beliefs that someone, who has the power, prestige or status to do it, has legitimated.

Bloor and Barnes and other researchers of the University Edinburgh funded in the '60 the *Strong program of sociology of knowledge* (*Strong Program*, for short) endorsing these ideas. This stream of research fits in with the tradition of sociology of science of Merton (cf. [23]) but has stronger objectives. Traditional sociology of science wants to explain the influence of social factors on the process that leads to a discovery, but does not believe that they influence also its content. We could say that it focuses more on scientists than on scientific theories. Basically, the contribution of sociology is considered useful to explain scientific failures. Correct theories do not need sociological explanations. Wrong ones can be object of a sociological analysis. On the contrary the *Strong Program* states that truth is a social product, thus all statements, even correct ones, have a sociological justification. For example, Bloor thinks that the psychologist approach to mathematics proposed by J.S. Mill still had full plausibility. Mill thinks that to understand mathematics is equivalent to understand the psychological processes that are carried out by mathematicians. Frege contrasted this idea, asking for an objective substrate of mathematics. Starting from Frege's objections, Bloor states that this substrate is provided by the inter-subjective layer of psychological processes, namely the social one. Mathematics, from this point of view, becomes essentially a social practice.

We remark that, among others, Popper was absolutely opposed to this approach and he believed that sociology and psychology cannot be used to ground science.

“... to me the idea of turning for enlightenment concerning the aims of science, and its possible progress, to sociology or to psychology ... is surprising and disappointing. In fact, compared with physics, sociology and psychology are riddled with fashions and uncontrolled dogmas ... This is why I regard the idea of turning to sociology or psychology as surprising.” [24]

However, independently from the question of establishing which one of these opposed approaches to knowledge is correct (which is not our task) we can retain that there

is an approach to scientific knowledge that tells us that a decision can be legitimately supported by a deal stipulated by all the people in charge of the choice.

Coming back to the point of view of our work, we can observe that collaborative decision making has its own tradition and, thus, indirectly, a kind of legitimation. We do not believe that this is the best method, neither that this is the only method, as strong program sociologists tell us. Nevertheless, in practice, when no other options are available, or empirical evidence is missing, decisions are taken by means of stakeholders' agreement. We concur that this is not inadmissible. In practice, it happens, quite often. In our experience, this is not unusual in projects management and systems design.

6 Ethical approach to model validation

In this section, we look in literature for relationships between ethics and science (OR and management sciences in particular).

Churchman [25] warns about the possible immorality of OR which, in his opinion, could not respect the Kant's moral law "*make only those decisions which treat humanity as an end, never as a means only*" since, in some occasions, OR treats people only as means, in order to achieve an optimum. Nevertheless, the relationships between ethics and OR are recurrent. Wenstøp [26] offers us a comprehensive overview of the last four decades, indicating the work of Boulding [27] as a divide. Boulding proposes OR as an instrument for ethics due to its capability of optimizing consequences of a decision and maximizing utility, which is the goal of some kinds of moral approaches, for example utilitarianism.

Ackoff observes that OR should take care of the interest of the stakeholders (an idea that is consistent with the approach we have adopted in this work).

"Decisions should be made by consensus of all who are directly affected by the decisions, the stakeholders." [28]

Wallace's edited book, *Ethics in Modeling* [29], covers several arguments related to the role of ethics in design disciplines and endorses an attentive care for stakeholders and ethical issues. Brans [30, 31] indicates Multi Criteria Decision Analysis as the OR tool that can "*take the interests of the stakeholders and nature into account, and calls for a multifaceted concept of ethics, consisting of respect, multi criteria management and happiness*" [26]. Gallo [32] underlines that the research should care about both the consequences of a decision and the respect of fundamental principles. He identifies the two that should ground OR. The *responsability* principle, based on the thought of Jonas [33], and the *sharing and cooperation* principle. Brans and Gallo [34] provide another historical account of the relationships between OR and ethics, indicating Churchman as one of the main initiators of this "match". They observe that:

"Unlike natural sciences, OR/MS⁶ [...] has as its object not natural reality but rather a man-made reality, the reality of man-machine complex systems [...] Hardly any area in OR/MS can be considered far enough from the real world to escape from ethical considerations".

⁶ Operations Research / Management Science (OR/MS)

Mingers [35] analyses the relationships between OR and *Discourse ethics* (DE), a moral framework developed by Habermas [36, 37]. According to Mingers, this theory fits well with the science of decision-making. Habermas thinks that we can, through the analysis of communicative structures, identify the conditions for the acceptability of a valid argument and that these conditions are common to a valid moral theory.

“How then should we apply DE to OR? [...] DE does not put itself forward as a panacea but it does provide a processual template against which proposals and decisions can be tested for ethical legitimacy, and, if followed, should lead to actions that are better in the long run for both organizations and civil society as a whole.” [35]

Le Menestrel and Van Wassenhove focus on the trade-off between

“scientific legitimacy of OR models (ethics outside OR models) and the integration of ethics within models (ethics within OR models)” [38].

This argument recalls the opposition of epistemic and non-epistemic values introduced previously. They identify three possible attitudes towards the relationships between OR and ethics. The first one corresponds to a sharp separation between them. It ensures objectivity of OR, but, in their opinion, is incomplete. The second one integrates ethics in OR. This approach is more complete, but has the flaw of accepting a certain amount of subjectivity. The third approach is based on a distinction between OR model and OR process. Ethics should be integrated with OR process, and not in the models. The OR process can operate as a connector between OR models and the real world and can include ethical matters without compromising the objectivity of OR models. Thus, they refer to this approach as *ethics beyond the model*.

“We present three methodological approaches to combine ethics with Operational Research. The first one is ethics outside OR models [...] The second approach is ethics within OR models [...] The third approach is ethics beyond OR models”

7 Teleological approach to model validation

In this section we focus on the concepts of goals and objectives, which pervade systems engineering. In particular, we dare a possible (audacious) link. The concepts of goal and requirement, used in systems design, have their conceptual “ancestors” in the Aristotelian *final causes*.

For empiricists, the concept itself of teleological explanation of phenomena, namely the existence of purposes and objectives in nature for the sake of which things are done, is inadmissible. This would confer to nature something like a “free will”, which is incompatible with the idea of nature as mechanism. However, Aristotle advanced aims as one of his famous four causes: *material, formal, efficient and final*.

“Aristotle was deeply committed to investigating and explaining natural phenomena, which is reflected all through the surviving treatises on natural philosophy [...] What unites the questions explored in these natural treatises, [...]”

is that they are predominantly questions asking for the purpose of things, or, as Aristotle puts it, questions asking for - that for the sake of which -. According to Aristotle's understanding of scientific knowledge, the answers to these specific why questions constitute teleological explanations [...] [39]

Final causes (or *telos*) differ from other ones from many points of view. The most evident difference is that “normally” causes happen before effects while in teleological explanations are the effects which occur first. In a causal explanation a first event E_1 happens at time t_1 and a second one E_2 at time t_2 . This is not a sufficient condition to state that E_1 causes E_2 , but it is a necessary one. In teleological explanation this temporal sequence is inverted. The E_1 happens at time t_1 to serve the second one E_2 at time t_2 , which is the cause.

“Whereas in a typical causal explanation the earlier-in-time cause explains the later-in-time effect, in teleological explanations, as traditionally understood, the later-in-time effect (that is, the aim or purpose for which something happened) explains the earlier-in-time cause (that is, why something happened). The typical locution of a teleological explanation is: this happened in order that that should occur.” [40]

Bacon recommended a limited use of final causes:

“Bacon... quotes with approval the Aristotelien maxim - Vere scire est per causas scire - and the Aristotelien distinction of four causes, Materia, Forma, Efficiens, et Finis [but proposes ...] his famous condemnation of final causes [...] He blames their use in Physics; he approves their use in Metaphysics”. [41]

Nevertheless, this kind of causes was admitted by authors such as Leibniz and Kant (among others).

“Leibniz did admit teleological explanations alongside mechanical ones. Apart from the need of teleological explanations (in terms of God's purposes) in metaphysics, he argued that physical phenomena can be explained by mechanical as well as teleological principles. ... Indeed, Leibniz wholeheartedly accepted the Aristotelian final causes alongside efficient causes”. [24]

The question is if science should admit or refuse final causes. We propose a compromise solution. In our opinion, the answer is that, anyway, they are actually used in everyday activity by engineers, during systems design, but are hidden by the use of a different terminology. Of course we do not claim the “airplanes want to fly” or “ships want to swim”. It would be an evident nonsense. However, stakeholders and systems have objectives, thus we simply suggest that the term “final causes” can have a (smooth) interpretation that is not incompatible with our *standard view* of science: the term “goal” is a (safe) synonym of the term “final cause”. From this point of view, we might say (quite provocatively), that requirements engineering and operations research are applied philosophy.

References

1. Lipton, P.: Engineering and truth, philosophy of engineering, vol.1
2. Bunge, M.: Technology as Applied Science. (1966)
3. Hanson, N.: Patterns of Discovery: An Inquiry Into the Conceptual Foundations of Science. University Press (1958)
4. Psillos, S.: Causation and explanation. Central problems of philosophy. *Acumen* (2002)
5. Anthonie Meijers. General editors: Dov M. Gabbay, P.T., Woods, J., eds.: Handbook of the Philosophy of Science. Volume 9: Philosophy of Technology and Engineering Sciences. Elsevier BV (2009)
6. Laudan, L.: Progress and Its Problems: Towards a Theory of Scientific Growth. Philosophy of science. University of California Press (1978)
7. Niiniluoto, I.: Is Science Progressive? *Synthese Library*. Springer (1984)
8. Schlick, M.: The future of philosophy. In: Proceedings of the Seventh International Congress of Philosophy, London (1931) 112–116
9. McMullin, E.: Values in science. PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association **1982** (1982) 3–28
10. Dorato, M.: Epistemic and nonepistemic values in science. In: Science Values and Objectivity. Pittsburgh-Konstanz Series in the Philosophy and History of Science. University of Pittsburgh Press (2004) 52–77
11. Shipley, J.T.: The Origins of English Words: A Discursive Dictionary of Indo-European Roots. Johns Hopkins University Press (1984)
12. Alberti, L.B.: De Re Aedificatoria Libri X. Alamani, 1485; English translation by James Leoni of the Italian translation by Cosimo Bartoli, Firenze, 1550. (The) Ten Books of Architecture. Reprint of the edition of London 1755; Tiranti, 1955. (1485)
13. Kapurch, S.: NASA Systems Engineering Handbook. DIANE Publishing Company (2010)
14. Landry, M., Oral, M.: In search of a valid view of model validation for operations research. *European Journal of Operational Research* **66**(2) (1993) 161 – 167
15. Dry, R., Landry, M., Banville, C.: Revisiting the issue of model validation in OR: An epistemological view. *European Journal of Operational Research* **66**(2) (1993) 168 – 183
16. R. P. Feynman, R.B.L., Sands, M.: The Feynman Lectures on Physics. Reading. MA: Addison-Wesley Publishing Company (1963)
17. Duhem, P.: *Sōzein ta phainomena: essai sur la notion de théorie physique de Platon à Galilée*. A. Hermann (1908)
18. Kuipers, T.: Laws, theories and research programmes. In Kuipers, T.A., ed.: General Philosophy of Science: Focal Issues. Handbook of the Philosophy of Science. Elsevier/North Holland
19. Kuipers, T.A.: General Philosophy of Science: Focal Issues. Handbook of the Philosophy of Science. Elsevier/North Holland (2007)
20. Popper, K.: Conjectures and Refutations: The Growth of Scientific Knowledge. Routledge Classics. Taylor & Francis (2002)
21. Duhem, P.: La théorie physique: son objet, et sa structure. Bibliothèque de philosophie expérimentale. M. Riviere & Cie. (1906)
22. Wittgenstein, L.: Philosophical Investigations. Basil Blackwell, Oxford (1953)
23. Merton, R.: Science, technology & society in seventeenth century England. H. Fertig (1970)
24. Popper, K.: Normal Science and its Dangers. In Lakatos, I., Musgrave, A., eds.: Criticism and the Growth of Knowledge. Cambridge University Press, New York (2004) 51–58
25. Churchman, C.W.: Operations research as a profession. *Management Science* **17** (1970)
26. Wenstp, F.: Operations research and ethics: development trends 1966-2009. *International Transactions in Operational Research* **17**(4) (2010) 413–426

27. Boulding, K.E.: The ethics of rational decision. *Management Science* **12** (1966)
28. Ackoff, R.L.: Business ethics and the entrepreneur. *Journal of Business Venturing* **2**(3) (1987) 185–191
29. Wallace, W.A.: *Ethics in Modeling*. Pergamon (1994)
30. Brans, J.: Ethics and decision. *European Journal of Operational Research* **136**(2) (2002) 340–352
31. Brans, J.: The management of the future: Ethics in OR: Respect, multicriteria management, happiness. *European Journal of Operational Research* **153**(2) (2004) 466–467
32. Gallo, G.: Operations research and ethics: Responsibility, sharing and cooperation. *European Journal of Operational Research* **153**(2) (2004) 468–476
33. Jonas, H.: *Technik, Medizin und Ethik. Praxis des Prinzips Verantwortung*. Suhrkamp, Frankfurt am Main (1996)
34. Brans, J.P., Gallo, G.: Ethics in OR/MS: past, present and future. *Annals OR* **153**(1) (2007) 165–178
35. Mingers, J.: Ethics and OR: Operationalising discourse ethics. *European Journal of Operational Research* **210**(1) (2011) 114 – 124
36. Habermas, J.: Discourse ethics, law and *sittlichkeit*. In Dews, P., ed.: *Autonomy and Solidarity: Interviews with Jürgen Habermas*. Verso (1992)
37. Habermas, J.: Three normative models of democracy. In Polity Press, C., ed.: *The Inclusion of the Other*. 239–252.
38. Le Menestrel, M., Van Wassenhove, L.: Ethics outside, within, or beyond or models? *European Journal of Operational Research* **153**(2) (2004) 477–484
39. Leunissen, M.: *Explanation and Teleology in Aristotle's Science of Nature*. Cambridge University Press (2010)
40. Psillos, S.: Past and contemporary perspectives on explanation in *General Philosophy of Science: Focal Issues, vol.1*. Handbook of the Philosophy of Science. Elsevier/North Holland (2007)
41. Underhill, G.E.: The use and abuse of final causes. (2008)

A model-based method to support complex system design via systems interactions analysis

Retho Fabien^{1,2}, Smaoui Hichem¹,
Vannier Jean-Claude², Dessante Philippe²

¹EADS Innovation Works, France

²Supélec, Département énergie, France

Abstract Designing a complex system is a multidisciplinary task that involves different profiles of engineers. In this collaborative process, each actor has specific skills, and nobody is able to consider the entire design project alone. From this observation two majors worlds have been identified, the system world including systems engineers and the physical world including discipline experts. Under this discretization, the method proposes to manage collaboration, and interfaces, between engineers coming from systems design modelling and physical worlds. To perform this objective, the method described in this paper is based on an enrichment of information linked to systems, via an analysis of interactions and mutual impacts of each subsystem, and then the building of a bridge to deal with physical world experts. All the methodology is model-based, and introduces a new concept called “model of intention” in order to initiate dialog between both worlds. This paper proposes a first approach to create models of intention for a hybrid (Electric/Thermal) propelled unmanned aerial vehicle (HPUAV) project.

1. Introduction

The new paradigm introduced by the hybridization of a propulsion system has emerged real design problems for propulsion system engineers. A first problem is the introduction of new electrical subsystems into the existing propulsion system. Indeed, these new subsystems have an influence on the other ones, and reciprocally. This report highlights a gap in design process: consideration of mutual impacts of a system on its environment. Due to the fact that direct environment of each system is composed by other systems, introduction of a new system impacts all the entire sizing of the other systems. Consequently, interactions between systems are important to consider in order reaching a valid, or an optimized solution. A concept of impact is introduced

to specify more precisely these interactions. A second problem detected by propulsion engineers, is their non-expertise on electrical systems. Consequently, the design and sizing of the system requires a close and smart collaboration with electrical experts who have the required knowledge.

The method proposed in this paper is focused on these two problems, and addresses them via the use of systems engineering and physical models. In order to offer a common platform to collaborate, the method is model-based. The goal is to strengthen the link between the systems engineering and physical engineering worlds. With the concept of impact, and the collaboration between people of each world, the method supports project technical orientations and decision-making.

2. General concepts

2.1. Main ideas and actors

The global idea of the method, illustrated in Fig 1, is to create a collaborative process between the system design and model world and the physical one (systems design and sizing considering physics). The aim is to be able to transit from one world to another via an interface managed by a new actor of the collaboration named “simulation architect”. The simulation architect (it can be a team) represents enablers for the architect to obtain expertise model-based conclusions. Each simulation architect has a multidisciplinary vision of a product, and simulation knowledge. It is the interface between the architect and the experts in term of models and simulations (model request, virtual product building, simulations...).

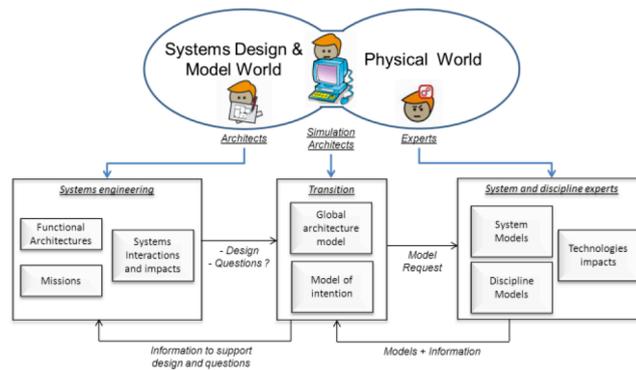


Fig.1 Approach proposed

2.2. Organisation in two worlds

System engineering is a key to develop and produce complex products. The classical V-Cycle is often used to represent and manage the progress of the design process. Following this approach, the suggested method allows to perform integration, verification and validation activities (right part of the V) virtually and frequently, via the use of behavioural model and simulation. In that case, behavioural modelling is mandatory, but cannot be managed by architect only. In the method, the architect takes decision at high level. He proposes the functional architecture which represents an input data of the process. All the functional architecture analysis is supposed to have been done with methods out of the scope of the proposed work. Missions are also required as an input for the methodology. Mission parameters (phases, vehicle speed, altitude...) must be defined in order to be used for simulations. Another task dedicated to architect is the interactions and impact analysis. It is deduced from a physical analysis of each subsystem environment and relations. The result of this analysis can be represented under the form of context diagrams [1] [2].

In modelling, physical world is represented by experts, who built models with their specific knowledge. One of the interests of the approach is to address clear model request to the expert, in order to obtain the right model at the right moment for the architect. Consequently, the physical model is built based on an intermediate model, called *model of intention*.

Liscouët-Hanke has proposed an approach to manage systems engineering and to transit to behavioural/physical in order to support design [3]. The link between two separated worlds is considered in her works. The formalization of system models to physics models transition motivates the method proposed in this paper.

2.3. Interfaces between worlds: Simulation architect role

De Tenorio works on conceptual design, via collaborative system engineering approach. He shows that the design of a complex system is multidisciplinary, and motivates interactions between systems analyses [4]. Following the conceptual phase, Basset & al. propose a tool to consider multidisciplinary design, and to manage physical discipline coupling [5]. The method proposed in this paper wants to focus more on transition from a world to another, and model building, than on tool or technical issues. The proposed method introduces a job, called "Simulation architect" whose main objective is to facilitate the collaboration between architects and domain experts. Simulation architect role is to insure that a dedicated simulation can be set up through collaborate with experts, using their specific skills in various disciplines, modelling and simulation (M&S). He has to manage a global view of the architecture of the simulation which is one of the virtual views of the architecture of the product, with a behavioural M&S filter. Typically, he is in charge of selecting M&S strategy to support design and architect's questions when M&S seems to be the right path to answer ques-

tion about mitigation in the design. His close link with project allows him to propose the most pertinent modelling strategy. In order to apply his strategy, he will build (or manage the creation of) different *Models of intention*, for a system, or for any interactions, to express what the architect has required in term of Physics (Phenomenon, equations, input/output), Information (Expected results, simulations, Interactions and Impacts) and Operations (Scenario, environment, constraints, missions...).

In fact, a *Model of intention* does not represent a new way to specify a model when software is at stake or when functional model are in interaction: ports, compression processes, multiscale representation are already used. As far as physics is concerned, we have not yet identified any approach that mixes physical models and functional models to prepare integration. Our scope is there: environment of systems is physics (i.e. Thermal, vibration or electromagnetic fields) whilst systems are behaviour (i.e. Signal, black-boxes with I/O ports). The modularity, versus classical document-based model specification based on requirements, allows experts to deliver more relevant models. Indeed, the scenario knowledge, coupled with the systems interaction knowledge, and with informative complementary source, allows experts to propose more adapted models for the project.

2.4. Interactions and Impacts concept (I&I)

I&I is the way selected for the method to augment knowledge and information directly in systems engineering models. This is a concept which considers that a system modifies its direct environment, and consequently the global design of a product.

- *Interaction*: Link between two subsystems with reciprocal action, effect or influence. If the architect modifies a subsystem, via a new technology or a new sizing, systems in interaction with it will suffer, or benefit, of this modification.
- *Impact*: Directed from a subsystem to another. The concept of impact is set in order to allow modelling. For example, thermal behaviour of an electric motor will impact the behaviour of a battery. The relationship is physically easy to understand, but difficult to capture via a model, sizing and simulation.

Paredis & al. proposed an approach based on M&S and interactions [6]. Their approach allows creating a link between two subsystems models through a specific and multi-granularity interaction model. The method proposed in this paper wants to support the building of such interactions models, jointly with system model building, and based on trace and justification inside a global project.

3. Method definition

3.1. Method workflow overview

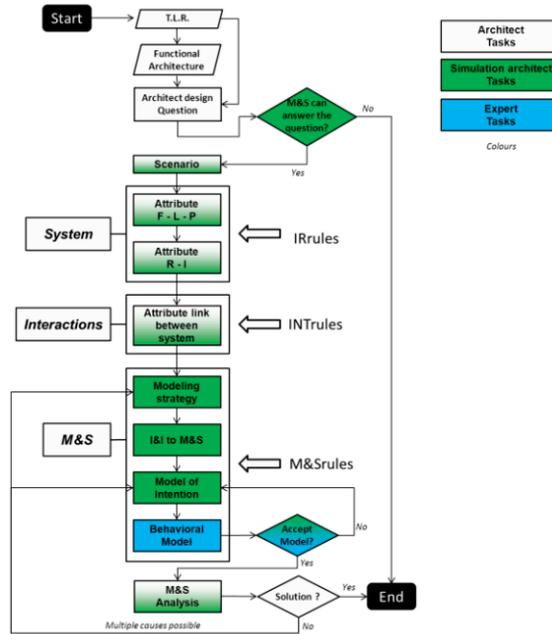


Fig.2 Method workflow and associated tasks

The workflow proposed in Fig. 2 is sequential, and describes different actor's contribution. It starts with two inputs, *Top Level Requirements* (TLR: requirements from customer, safety, project ...) and *Functional Architecture* (developed using methods not considered in this paper), and stops when the architect got an answer to his questions. Indeed, the method is set up to support the architect in his decisions, expressed through questions, with the support of the simulation architect and expert analysis, based on models.

In the workflow, it is possible to highlight three major blocks, *System*, *Interaction* and *M&S*, which will be detailed in the next parts. Sequencing of this block is mandatory; no parallelism is possible due to the dependence of *M&S phase* on *Interaction phase*. Update of inputs during the process requires running it again from the point where the new information is considered. Three rules, one for each block, are defined to support the sub-steps. Description of these rules use is proposed under block description.

3.2. System Block: System description

3.2.1. Customized FLP decomposition

The method is based on decomposition named Requirements Functional Logical and Physical (RFLP) [7]. This decomposition is interesting because it keeps all the Model-Based System Engineering (MBSE) foundations. However, with the method, impact (I) is added to obtain FLP-RI decomposition for a system.

In order to apply FLP-RI approach, all systems are defined with attributes, sorted through F, L or P. An attribute is an object defined with a template (Fig 3) which supports characterization of the system. Attribute supports M&S transition. This work is done for all involved systems, first by the architect, and updated by the simulation architect. In the method, FLP are customized differently from usual:

- *Functional (F)*: All functional aspects; how the system runs and attributes that supports the system during its work.
- *Logical (L)*: Usually, L considers functions allocation on physical systems. Used in the method more as an operational view (mission, hybrid modes...).
- *Physical (P)*: Real objects, hardware and geometrical aspect, are considered.

Field	Description
Description	Description
Dimension	Dimension
Value	Fixed value for the attribute
ValueMax	Maximum Value for constrained requirement
ValueMin	Minimum Value for constrained requirement
Impact_To	[List of attributes impacted]
Impact_By	[List of attributes that impact]
Type	M&S type (variables, parameters, inputs, outputs)

Fig.3 Attribute object template (XML) completed progressively with FLP-RI

3.2.2. Requirements (R) and interaction (I) management

At this point, the system has attributes ranked into FLP. To progress to FLP-RI, the next step is to manage R and I, which are defining new types for corresponding attributes. These types are associated to attributes by the architect, with the support of specific rules.

- *Requirements (R)*: Requirements are cascaded as “top-down” approach, from top-system to subsystems. Requirements can also derive from a specific technology. For example, a battery has specific thermal requirements, not cascaded from TLR. The objective is to associate attributes of each system to an equality or inequality requirement.
- *Impact (I)*: Impacts is a new type introduced to consider how a system will influence another one. This type is associated to an attribute considering as “impacting”. This selection is done by the simulation architect, supervised by the architect, in consideration of project and questions. This attribute selection is done from subsystem to system (“Bottom-up”).

- *Rules (IRrules)*: These rules authorize, or not, the assignment of *R* or *I* type to an attribute. For the moment, these rules are adapted for each project. The work is in progress to determine if rules are strongly dependent to a problem, and if generic rules could exist.

3.3. Interaction block: Management of systems environments

When all systems have been described with attributes, sorted in different FLP, and specified as R-I, it is possible to evolve to interactions management. The objective is to be able to generate connexions between two FLP-RI system descriptions. Due to the large number of potential interactions, generation shall be semi-automatic or automatic.

The idea is to link impacts type attribute from a subsystem to requirements type attribute of another subsystem. Port-based approach, completed with rules named *INTrules* to connect ports, automatize the process. Indeed, use of attribute as object is an advantage for this phase. To create rules, an approach by interaction matrix, Design Structure Matrix (DSM) or N^2 diagram applied to attributes is used [1] [8]. As for *IRrules* proposed in the last paragraph, *INTrules* are for the moment specific to a project, and matrixes fulfilled manually. Multiple solutions for generic rules are under investigation, and are not presented in this paper.

At the end of this sub-process, it is possible for architects to visualize specifically the impacts from one subsystem to another. This information will support the next steps: M&S request and building.

3.4. M&S Block: Transition from system to simulations

3.4.1. Modelling strategy

Modelling strategy is implicitly linked with simulation and scenario. The strategy is defined by the simulation architect to support the architect's questions or design process. This important part of the method drives the next phase, which is based on I&I information, to request model via *model of intention*. For the architect, questions will be used to progress in the design Strategy is defined based on results expected, via a modelling translation of the architect's question. Some several indicators shall be validated:

- *Inputs*: Data available, previous results, scenario.
- *Outputs*: It represents what the model must calculate;
- *Type of model / Granularity*: Number of dimensions (x_D , $x=[0;3]$), mix of models of different dimensions, degree of expected complexity (link with accuracy and validation), simulator or design model.

3.4.2. Transition from I&I to M&S using *model of intention*

A major interest of developing an approach based on I&I is physical information source brought to support a formal model and simulation building. At this point of the method, attributes are defined under FLP-RI decomposition. The objective is now to add a new type to each attribute, type associated to more physical modelling (parameter, variable, input or output). This association will create the link with the physical modelling world.

Modelling strategy already delivers information on model's inputs and outputs. Other attributes need a new set of rules, named *M&Srules*, which allows linking FLP-RI to M&S type. These rules determine, following if attribute is *R* (Requirement) or *I* (Impact), and according to modelling strategy, how to manage the M&S type attribution. As for other rules, no generic solution will be introduced in this paper.

At the end of this step, the simulation architect has a clear vision of the future physical/behavioural model. The addition of M&S filter is mandatory to evolve to a clear *model of intention*. This model will be a mix of all information (I&I and scenario), plus an empty model with interfaces and parameters (Fig 4). *Model of intention* can be transmitted to the expert. Model of intention is a model-based approach to request and specify model(s) for a specific scenario. It helps the experts to propose adequate model(s), and to propose advices, technologies or technical solutions.

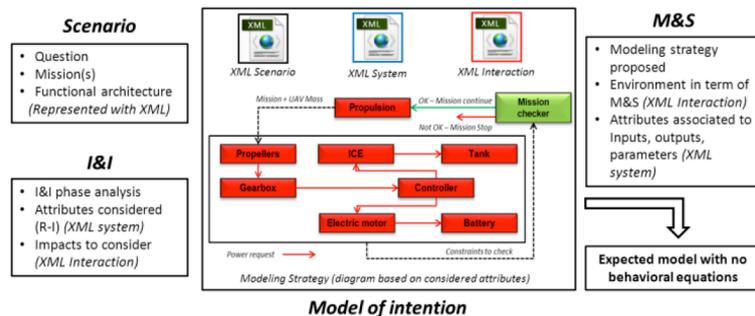


Fig.4 Model of intention for a system's model request

4. Hybrid Propulsion System for UAV

4.1. Project description: Mission & architectures

To demonstrate the methodology, we investigate on the evolution of a Vertical Take-Off and Landing (VTOL) UAV from a Thermal Propulsion (TPUAV) to a Hybrid Propulsion (HPUAV). Hybrid term is defined as: "Vehicle in which propulsion energy is available from two or more kinds or types of energy stores, sources or converters, and at least one of them can deliver electrical energy." [9] [10]. The project's objective

is to check if it is possible to obtain better performances (e.g. range, payload, operational cost...) with HPUAV than with TPUAV, satisfying the same requirements. As a first input, the architect has identified a viable functional architecture, which allows keeping TPUAV subsystems, and just added two supplementary systems (Electric Motor and Batteries, including power electronics). This architecture is inspired by automotive industry, and usually presented as “hybrid parallel” (Fig 5) [11]. In order to perform the entire method, a three phase’s mission is fixed (Take-off, Loitering and Landing). All this work builds the “Scenario” (Fig.2).

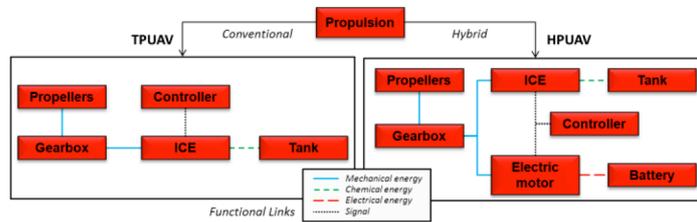


Fig 5: Propulsion system functional architectures (Energy representation)

4.2. Attributes, FLP and R-I

We take the hypothesis that the requirements cascading from UAV to propulsion system is done. The first step of the method is dedicated to subsystems, and highlighting of attributes that characterise them. Then, identified attributes are sorted with FLP decomposition. This identification and decomposition applied to the electric motor is presented in Figure 6. Same work is done for all systems.

The R-I phase is done as presented in previously. With the attributes selected, each of it is treated with R-I type. Requirements identification and management is directed by cascading, or hypothesis (example, minimal efficiency of the motor is fixed at 95%). Impacts are selected on engineer experience, and degree of freedom of scenario. This work corresponds to “System block” in general process (Fig.2).

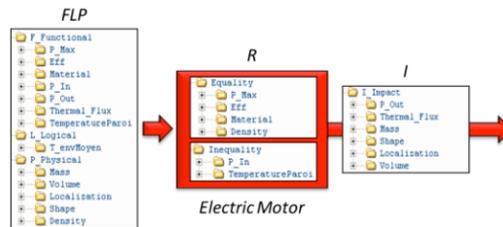


Fig 6: FLP R-I for electric motor (“System block”)

4.3. Interaction between subsystems

Interaction phase objective is to connect all systems in a network. With HPUAV, 28 interactions are possible with 8 systems at 2 levels. The huge number of interactions justifies the automation of interaction object building. For the project, specific *IN-Trules* has been created around disciplines. Each attribute has an associated discipline and so, Impacts type attributes can be combined to Requirements with the same discipline. This tag association allows creating a simplified network, which represents an information source that can be filtered by the architect. As a simple example, it is possible to visualize that the size of a system will impact possible size of another one. Note that with such tag association, some trans-disciplinary impacts cannot be identified directly. For example, the impact of subsystem's size on system thermal behaviour is not detected. However, thermal dissipation of subsystem is linked to its size and shape, so links are done inside each subsystem model. Finally, all subsystems are connected. The architect can express his questions. This work corresponds to "Interaction block" in general process (fig.2).

4.4. M&S transition

The question is proposed by the architect, based on project's objectives: Is it possible to overpass requirements imposed to TPUAV with HPUAV, in term of endurance on loitering phase and payload?

The simulation architect proposes a first modelling strategy considering mass (for payload) and duration (for endurance). He proposes to develop a power exchange based simulator, which runs on the sizing mission. An optimization process must then be performed with this simulator. Due to power exchange modelling, 0D causal dynamic modelling is proposed. Modelling strategy used in this part is presented in upper left of Fig 4. This strategy starts with propulsion system, which delivers *Mission* and *Mass* information to Propeller. Propeller requests power to perform mission, and is cascaded to other subsystems and then reach the two different energy sources (battery and tank). Power requested to source is integrated versus time in order to determine energy. Energy contained in sources is directly determined with the mass of energy source subsystem (internal energy density). Simulator determines if mission is a success, and delivers results for future questions (i.e. design phases). To determine success of a mission, numerous constraints (based on *R*) are applied to subsystems (for example, no more energy in battery). Each set of parameters for a simulation represents an architecture sizing: if mission is successful, sizing is correct. In order to compare efficiently architectures, block-based modelling is proposed. It allows reusing subsystems models and easily builds architectures. Custom control of hybridation logic is applied.

4.5. Build and use of the model

With scenario and modelling strategy, I&I phase is used to select pertinent attributes. Each selected attribute receives a specific M&S type, according to M&Srules (parame-

ter, input...). Subsystems models have a description, but no behavioural equation: *Model of intention* is built and can be sent to the experts. As this example is quite simple, models received are integrated in a Modelica tool, Openmodelica, supported by a custom library [12]. Optimization is based on maximization of two objectives (payload and loitering duration), and 8 optimization variables. An evolutionary algorithm is selected to perform analysis. The double use of the simulator helps to answer the architect's questions, and brings information for future steps (Fig 9).

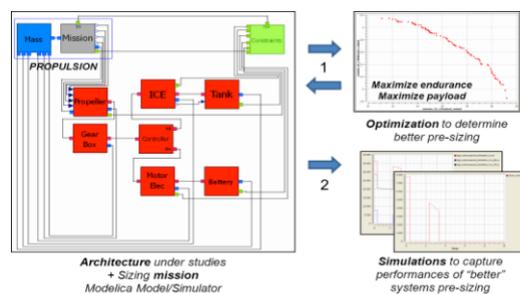


Fig 9: Global model finished: Optimization to check question (1) Information for sizing to increase future *model of intention* and continue to next design phase (2)

5. Conclusion

The proposed method follows the idea that increasing the knowledge on a system under design, with the support of physical world, will aid the architects in their decisions. Around a three-phase workflow, the method manages the transition from pure system approach to physical modelling and simulation, with enhancing of collaboration and expertise. Each method's phase brings traceable, storable, and reusable information. The dynamicity and flexibility of the approach allow managing different phases of a system design project.

For the moment, the method does not address the ranking of interactions, used to highlight where it is important to remain attentive and to deploy M&S studies. This feature will be added, by the use of parametric weights on impacts and requirements attributes. Another point concerns multidisciplinary aspects. Due to the flexibility, it is possible to add new discipline's attributes. Work progresses to consider, in the HPUAV use-case, thermal, electromagnetism and geometry (3D).

References

- [1] INCOSE (2011) Systems Engineering Handbook
- [2] NASA (2007) Systems Engineering Handbook
- [3] Liscouët-Hanke, S. (2008). A model Based Methodology for Integrated Preliminary Sizing and Analysis of Aircraft Power System Architectures.

- [4] De Tenorio, C. (2010). Methods for collaborative conceptual design of aircraft power architectures.
- [5] Basset, P.-M., Tremolet, A., Cuzieux, F., Reboul, G., Costes, M., Tristrant, D., Petot, D. (2012) C.R.E.A.T.I.O.N. the Onera multi-level rotorcraft concepts evaluation tool : the foundations, Future Vertical Lift Aircraft Design Conference.
- [6] Paredis, C. J. J., Diaz-Calderon, a, Sinha, R., & Khosla, P. K. (2001). Composable Models for Simulation-Based Design. *Engineering With Computers*, 17(2), 112-128. doi: 10.1007/PL00007197.
- [7] Vuillemin B., Croue N., & Loembe S. (2012) MBSE Applied to an Aerospace “Force Fighting” Application, ERTS 2012
- [8] Browning, T.R. (2001) Applying the design structure matrix to system decomposition and integration problems : A review and new directions, *IEEE transactions on engineering management*, Vol 48, N°3
- [9] International Electro technical Commission, Technical Committee 69
- [10] Chau, K. T. (2002). Overview of power management in hybrid electric vehicles. *Energy Conversion and Management*, 43(15), 1953-1968. doi: 10.1016/S0196-8904(01)00148-0.
- [11] Chan, C. C. (2002). The state of the art of electric and hybrid vehicles. *Proceedings of the IEEE*, 90(2), 247-275. doi: 10.1109/5.989873.
- [12] Openmodelica, <https://www.openmodelica.org/>

Model-Based Interchange Formats: a Generic Set of Tools for Validating Structured Data against a Knowledge Base

Pascal Rivière¹, Olivier Rosec²

¹Head of Methodology Dept.

²Head of Social Data Interchange Unit
(within the Methodology Dept.)

Caisse nationale d'assurance vieillesse (Cnav)
110 – 112 avenue de Flandre F-75019 Paris

Abstract -A Opting for a model-based approach to develop a set of tools for validating structured data concentrated at the beginning on the generic control engine which would read a knowledge base containing rules. But to attain this goal, one had to develop a Model Editor which over time evolved into a full-fledged Integrated Development Environment (IDE) for the modeling of structured data formats, the specification of their validating rules and the generation of the knowledge base.

A) Introduction

In this era of fine-grain interactions between complex systems across distributed architectures, file processing has acquired a somewhat quaint flavor. But even nowadays there is no other way to transmit complex data from one system to another.

Difficulties are many at all stages:

- Defining the interchange format between partners is not easy;
- Ensuring the actual files meet the quality standards expected in production means multiplying control rules;
- Defining the architecture, protocols and syntax for file processing platforms adds yet another dimension to the conundrum.

This paper is about the search for a generic approach covering the first two points.

B) Problem Domain: Rationalizing the Social Data Collection Format

In France social protection is split up in several schemes administered by different agencies. Over the course of time each agency has discovered the hard way that it is less costly to acquire data about the future claimants of benefits in a steady stream, at the source, directly from the payroll system, rather than on an *ad-hoc* case per case basis, from the claimant.

Thus all agencies which pay out old age pensions on the basis of contributions paid along one's working life have turned to collecting the data about employees' pay on a yearly basis, instead of collecting it from the faded pay slips of a lifetime when the employee claims his or her pension.

Social data, as a broad term covering pay-related data, also serves to test whether an employee is entitled to this or that benefit, be it a sickness benefit or an unemployment benefit.

Other services outside the social protection sphere have been interested as well: the Inland Revenue, the National Office of Statistics want to use that kind of "big data" either for sending out tax forms pre-printed with income returns or to conduct surveys.

Initially the data collection process relied on paper forms. The paper forms merged into a single one, and as the process went digital during the 1980s, that huge form gave birth to a file format. The interchange medium switched from tapes and diskettes to the Web around year 2000. Today, more than one million employers send files at the beginning of each year.

The success of that particular community has attracted more and more partners, because once a reliable channel for the transmission of data from payroll systems to the information systems of public services has been found, it is far easier to plug into it than to set up a brand new one from scratch.

A new standards body was set up in 2008 to organize the process of collecting requirements beyond the original community of partners. But the standards body has no leverage whatsoever on the data collection process: the data is in fact distributed over a series of platforms. It cascades through a complex splitting and filtering process so that each administration gets the data relevant to its business purposes and just that. And some partners insist on running their own platform.

C) The Interchange Format as a Maintenance Nightmare

The **interchange format** is represented as a **hierarchy of data blocks governed by an alphanumeric naming scheme**.

At file level, data elements are physically represented on a key-value basis, the key being the identifier of the data element according to the naming scheme. There is no physical notation of data element blocks. The naming scheme enforces the model organization within the flat file format.

Separators and an end of line character are other precisions given in the file format specification, as well as the character encoding, with restrictions for particular data elements.

There is no typing other than alphanumeric, numeric, date. Typing can be further refined by regular expressions and minimum and maximum lengths. Some data elements have to belong to a list of values defined as an enumeration or carried by an external referential.

Control rules, written out in natural language, describe consistency checks between data elements: co-occurrence, comparison tests enforce semantic validity at file level.

Yearly Change Requests : the Maintenance Challenge

Each year nearly one thousand change requests are introduced by partners because of:

- changes in legislation;
- “patches” to solve production issues arisen during the last data collection campaign.

The national agency in charge of the format must then update the file specification and each team must update the corresponding application code on their data-processing platform.

The frequency of change requests has created a maintenance challenge which is further aggravated by the following facts:

- The specification is considered as a document to be discussed during countless proof-reading sessions;
- The focus, instead on being on concepts, is on implementation details. There is no proper conceptual data model independent of the file format. There are only broad rules governing the organization of data blocks carrying data elements along several axes:
 - A semantic axis along which one finds in succession the description of the party sending the file, of the employer, the employee and the business data for this employee;

- A temporal axis which governs the insertion of working periods for an employee within the timeframe carried by the file: month, quarter, year;
- An “ownership” axis because business data is split between “common” business data received by all partners and business data specified by and “belonging” to a particular partner.

Administering the format specification along those three axes gives birth to one of those combinatory explosions which go hand in hand with a requirements elicitation process chugging along contentedly in chronic happy-hour mode. The maintenance challenge turns into a nightmare.

The Stand for a Generic Model-Based Approach

The national agency in charge of the file format and historically responsible for the main file processing platform got fed up with:

- The absurdity of writing specific hand-crafted code which had to be thrown away each year as the file format specification evolved;
- Squabbles between developing teams over the interpretation of this or that rule;
- The slow turnaround time when a control program had to be patched.

It made a stand in favor of a generic approach and took a step further the breakaway from a mere paper specification. **From a single referential** which would represent the file format, **one should be able to generate:**

- **The documentation** for implementing it across the community;
- **A knowledge base.**

The knowledge base would be read by a generic engine which would execute all rules. The engine would remain the same over the years. Only the knowledge base would change.

The whole specification would become machine executable. A team would take care of the modeling which would produce both documentation and knowledge base. No more code, no more developers. But first one had to jump over a few hurdles.

D) “Abstract Implementation”: Domain-Specific Languages

To enable the design and development of a suite of tools addressing the needs of the modeling team in charge of the file documentation and knowledge base, first one had to lay the foundations:

- Meta-models for the file format and deliverables;

- And transformation strategies to be applied to the single referential persisting the models, to generate the deliverables.

The software solution has been designed on the basis of a Domain-Specific Language.

“A DSL is a programming language tailored specifically to an application domain: rather than being for a general purpose, it captures precisely the domain's semantics. (...) DSLs allow the concise description of an application's logic reducing the semantic distance between the problem and the program.” [Spinellis, 2000].

Each time we can, we will use Spinellis's taxonomy of patterns in the remainder of this paper to explain the way a DSL supports the software process which is being described.

The priority for the problem domain was to design the data model from which interchange formats would be built. The model articulates three libraries:

- A **Structures library** describing data blocks composed of data elements;
- A **Data types library** describing the types for data elements;
- A **Messages library** describing each interchange format as a hierarchy of data blocks.

The three libraries persist the current data interchange format modeled with the help of the meta-model. This corresponds to the data structure representation creational pattern [Spinellis, 2000].

Data block properties include:

- An identifier composed according to the naming scheme;
- A functional name;
- A description;
- A multiplicity (there can be 0, 1 or N instances of each block).

Data element properties include:

- An identifier composed according to the naming scheme;
- A functional name;
- A description;
- A usage (each data element can be within a certain block mandatory, conditional, optional, or forbidden).

Rules are attached to data elements. Block level rules are attached to the first data element in the block. Rules properties include:

- An identifier;
- An execution context;
- A message to be returned to the user in case the rule is triggered and not satisfied;
- The rule in natural language.

Semantic rules have been represented by a **textual DSL** which was first specified in EBNF. The rules are written as **mathematical propositions enforcing first-order predicate logic**. They can include **existential or universal quantifiers**. Semantic rules are written using the fully qualified identifiers for the data elements. Thus they are easily read and debugged.

Semantic rules can call **macros** and **aliases**. Both can be used as shorthand to simplify a complex rule: for example, does this employee belong to the public sector and if it is true then execute B and if not execute C. Semantic rules can be extended by **functions** mapped to the function prototype of an executable language.

Documentation has been modeled too. A file format specification is a document consisting of:

- Resources which are references to static document or spreadsheet formats;
- Templates for exploring the referential, through a reporting engine which will bring back the selected objects: messages, data blocks with their elements and types and rules.

The DSL which federates the resources and parameters for documentation generation illustrates the system front-end DSL pattern [Spinellis, 2000].

E) “Concrete Implementation”: The Eclipse Modeling Framework

EMF’s main “selling point” (it is for the most part open source and free) is that it is built on top of the Eclipse platform. The Eclipse platform is in itself an asset, providing countless mechanisms and wizards for managing projects, writing, compiling and debugging code, managing code libraries and source repositories, tracing file change. It plugs into most source control and ticketing tools.

EMF started according to the literature [Merks, Gronback, 2009] as a reaction against the profuseness of the Unified Modeling Language. A subset of UML constructs called Ecore articulates the minimum set of components to build models from EElements, EClasses, EAttributes etc.

EMF enables one to build such a model, from scratch through the appropriate editor, or through the transformation of:

- A UML model;
- Annotated Java code;
- XML Schema.

The Model Editor

With EMF one can build quickly an editor to manipulate business models. A powerful API helps enforce Model View Controller (MVC) and command stack mechanisms. Models can be persisted as resources in an XMI style syntax. Various template engines are available for model to model or model to text transformations.

These transformations combine the source-to-source transformation creational pattern and the pipeline behavioral pattern [Spinellis, 2000].

Over three years the File Format Editor has gone through many different versions as models were refined and deliverables tuned to the needs of the user community.

Model resources have been organized into a model bundle within which a catalog file points to all resources such as the three aforementioned libraries.

The same models go into the making of the knowledge base which is compiled as a Java project and organized in directories read by the control engine as it goes through its different processing stages.

Automatic generation reduces turnaround time to deliver a new knowledge base to one hour, including non-regression tests which have been automated (test files reports are parsed to compare the obtained result with the expected result), once for instance a rule has been patched.

The Three Representations of a File Format

The file format is modeled in the Editor through a graphical user interface.

The seminal decision was to represent the file formats in XML Schema in the knowledge base. All other decisions hinge on that choice.

XML Schema is a cheap and common way of structuring data. It offers strong typing. An XML instance can be parsed and validated against the schema it purports to respect.

But the actual files remain true to the legacy flat key-value format.

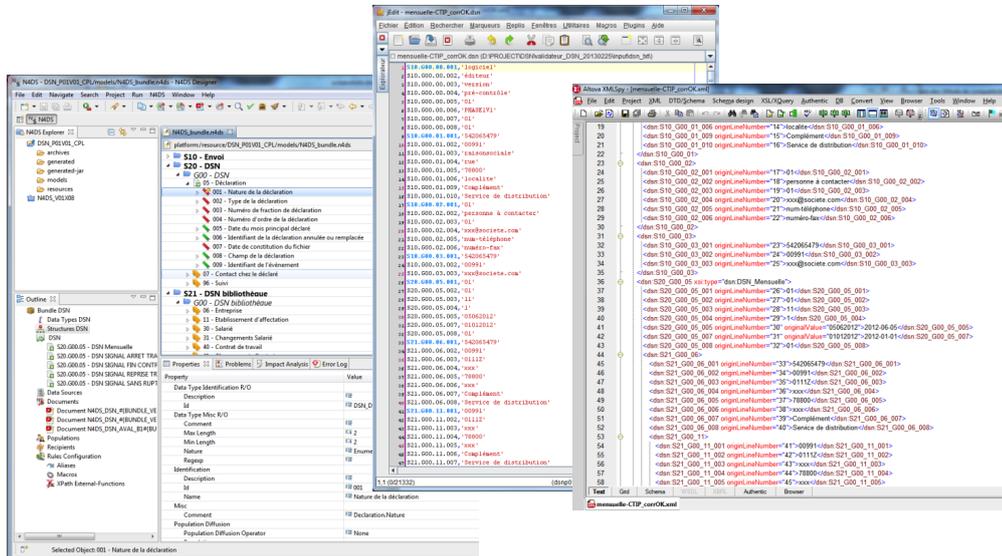


Figure 1 shows side by side the model as seen through the editor, an instance of the flat file legacy format, and its XML conversion.

The Validating Engine and its Processing Stages

The generic control engine processes a file in three stages:

- **Conversion** from the flat key-value legacy format to a hierarchical XML instance;
- **Syntactical control**, by validating the instance XML file against the XML Schemas;
- **Semantic control**, by firing one after the other the rules attached to data elements.

The control logic is static. There is no interface to live databases to check the existence or the status of the value of a data element. Only the knowledge base will be read.

The Three Representations of a Semantic Rule

XML Schema offers no easy way to enforce consistency constraints between data nodes. One has to write specific code. But specificity was not the order of the day. Hence the decision to implement the textual DSL described earlier in the paper.

Another problem dogging the processing of big XML files is memory management which opens up on the usual alternative: event-driven parsing (SAX) or document loading (DOM). In the time-honored way of hand-crafting control code, one positions control rules involving variables belonging to data blocks stretching across the whole file when the last necessary variable will have been read and stored.

The original vision wanted to dispense with the turnaround time associated with hand-crafted code. So the parser which transforms textual DSL had to transform it into machine executable code supporting:

- The test logic which would return a Boolean;
- The data addressing mechanism;
- And ultimately a memory-management mechanism.

A semantic validation API in Java covers all three issues, and more specifically memory-management through a twin set of utility classes loading and unloading variables as the engine fires rule after rule to check a file, however big it may be. The API rests on the convention that data elements always have the same address, the one they have in the “covering message” which is a superset of all messages within the model.

Hosting a semantic rule API in Java corresponds to the piggyback structural pattern [Spinellis, 2000].

F) The Validating Engine in Real Life

One should speak less in terms of an implementation gap and more in terms of a consistent way of dealing with the issues which arose in the course of the project and which had to be solved on the spur of the moment as the product neared roll-out time in late 2012.

The Project Cycle

If one adopts the Y shape used to describe the fusion of the upper branches carrying business requirements and system-level frameworks into an end-product, one should say that the work cycle, instead of trickling down the Y, more or less pulsated in radiating circles from the middle of the Y, as, from version to version, the set of implemented functionalities and the range of transformation strategies and frameworks used to develop the product expanded from the original nucleus.

But there are issues associated with deployment which can be addressed only with the help of real user and qualification team feedback. This feedback accounts for the A shape superimposed on the Y shape. The A shape denotes:

- Deployment issues such as performance, ease of integration;

- Usability in terms of user-friendliness, which means reducing the distance between the original file and the converted file processed by the control engine, by keeping as attributes:
 - the original value of certain data elements transformed from the legacy string format to comply with one of XML Schema's built-in datatypes (for example, dates);
 - the line number of the data element in the original file.

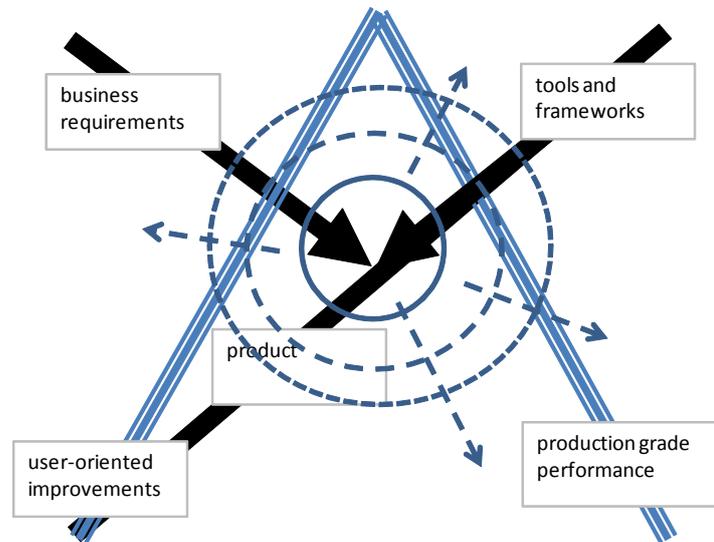


Figure 3 superimposes the Y (development) and A (production and user feedback) cycles

Stateless Mode and Report Stream Related Issues

Processing files in a production environment means processing gracefully even badly damaged files, to return a user oriented report and not just a log trace. And the user community wants validation reports to be exhaustive to understand what was wrong with the file and the system it comes from. The control engine is **stateless** and goes from one stage to another even if errors were detected at an earlier stage. But errors at an early stage provoke errors at later stages: the report gets more and more confusing for the user.

It might prove more efficient in the future to stop processing files at a certain stage. This could mean redesigning the report stream which is open and closed at each stage (intermediate reports are then merged into a full report). A continuous report stream could be a better solution and would provide the interface necessary to stop file processing before the user report loses all relevancy.

G) Return on investment

Originally the suite of tools was developed to support the **Norme pour la Dématérialisation des Déclarations de Données Sociales** (N4DS: 800 data elements, 600 semantic rules). It now supports the **Déclaration Sociale Nominative** (DSN: 400 data elements, 120 semantic rules) as well, with no fork in the code of both Editor and Engine. Since the roll-out of the first DSN validating component, numerous releases have been made, including several emergency knowledge base patches within half a day. This would have been impossible with hand-crafted code.

References

Diomidis Spinellis, « Notable Design Patterns for Domain-Specific Languages », *Journal of Systems and Software*, vol. 56, no 1, 2001, p. 91-99

Dave Steinberg, Frank Budinski, Marcela Paternostro, Ed Merks, “EMF Eclipse Modeling Framework”, Addison-Wesley, Pearson Education, 2009.

Richard C Gronback, “Eclipse Modeling Project, a Domain-Specific Language (DSL) Toolkit”, Addison-Wesley, Pearson Education, 2009.

Runtime Fault Prediction and Prevention for Emerging Services in System of Systems

Imad Sanduka¹, Tim Lochow¹ and Roman Obermaisser²

¹EADS Innovation Work

Imad.sanduka@eads.net, Tim.lochow@eads.net

²University of Siegen

roman.obermaisser@uni-siegen.de

Abstract: SoS Models used for SoS design and requirements elicitation. At runtime operations SoS is under failure risk that resulted from its emergent behavior due to its constituent systems autonomy and system complexity. In order to mitigate SoS failure effects and prevent SoS failure we propose SoS model extensions and system failure prediction and prevention framework that enhance the usage of SoS state of arts models and provide emergent behavior control over SoS runtime operations.

1. Introduction

Emerging services are one of the prime reasons of constructing System of Systems (SoS). SoS considers joining diverse systems capabilities and taking advantages from their positive interaction. However, achieving the constituent systems interaction and collaboration under SoS community brings system engineering complexity and emergent behaviour challenges. SoS emergent behaviour can be desired or undesired behaviour. The desired behaviour is the goal of constructing SoS where it delivers services that can't be provided by one system alone. The undesired emergent behaviour is unexpected behaviour resulted from the interaction of autonomous constituent systems with operational and managerial independence. The undesired emergent behaviour affects negatively the SoS emergent services and increases the risk of system failure which must be avoided. During the SoS life cycle and under run time conditions where the system is in operation, undesired emergent behaviour and failure detection reduces the failure risk and makes the system more reliable.

This paper presents a framework for SoS modelling approach, for failure detection and prevention abilities. The framework enhances the SoS models with dynamic analysis properties and its usability under run time operations for undesired emergent behaviour detection and prevention. We introduce a failure detection approach with

failure handling and risk mitigation methodology supported by prediction of possible future SoS behaviour to guarantee the SoS survivability.

The first section is an introduction of the paper and its scope. In the second section SoS is discussed with its definition and emergent behaviour characteristic. The failure detection and prevention approach is presented in section three and a use case is illustrated in section four. Finally the future outlook and conclusion are presented in section five.

2. System of systems

Definition: System of Systems (SoS) are large-scale concurrent and distributed systems that are comprised of complex constituent systems (Sahin et al. 2007). The constituent systems are distributed over hardware, software, services and organizational systems, and characterized by operationally and managerially independence, evolutionary development and geographically distributed systems (Maier 1998). SoS are used nowadays in many sectors, e.g. military, air traffic management, emergency response, and water management and distinguished from monolithic systems by their unexpected emergent behaviour.

Emergent Services: The purpose of constructing the SoS is to provide emergent services that could not be achieved by one constituent system alone. The emergent services resulted from the cross interaction between the constituent systems that belong to the SoS and offer their services to each other. For example, in a military mission the air force system, surveillance systems, and communications systems work together in order to destroy the enemy tanks, constructing a SoS with clearly defined goals. Another example of emergence (Jamshidi 2008) is the symphony produced by an orchestra. The symphony is produced due to the interaction between different instruments and music players, where none of the music players can produce it in isolation.

While integrating the capabilities of heterogeneous systems brings new desired services, it may also lead to undesired emergent behaviour because of the constituent systems autonomy and their unexpected interactions. According to (Zhou 2011) ‘The emergence of SoS cannot be foreseen through analysis because it comes from collaboration and autonomy of constituent systems’. In (Jamshidi 2008) the author presents the preferred definition of emergence as ‘something unexpected in the collective behaviour of an entity within its environment, not attributed to any subset of its parts, that it present (and observed) in a given view and not present (and observed) in any other view’. As emergent behaviour is unexpected, a climate facilitating the emergence of desired behaviour and mechanisms for the early detection of undesired emergent behaviour is required (Gorod & Gove 2007).

Desired and undesired emergent behaviour: Emergent behaviour is often unexpected behaviour resulted from the interaction of different autonomous systems. It could be a desired behaviour, represents an opportunity to the system, or undesired (bad) behaviour that forms a risk for the SoS. Desired emergent behaviour is the purpose of build-

ing the SoS in the first place, where a monolithic system cannot fulfil the requirements alone. Since the number of interactions between constituent systems increases exponentially with their number, emergent behaviour cannot be predicted in the current state-of-the-art. The choice for the constituent systems needed for constructing the SoS depends on their capabilities, communication characteristics, and their individual constraints, without considering the behaviour of these systems when they are working together. The problem with undesired emergent behaviour is its consequences on the environment, the SoS, and the constituent systems. Unexpected behaviour can prevent the SoS from offering its services, which can even imply the loss of lives in safety-critical systems such as military or emergency cases.

SoS failure: A failure is defined as the deviation of the behaviour from the specification that prevents a system from providing its intended services. In SoS we can distinguish two levels of failures: constituent systems failure and SoS failure. The consequences of failures at constituent systems level depend on the failure type and role of the constituent system itself in the SoS. Failures can occur due to physical effects that hinder the system from providing the required operations or due to software faults resulting in improper system behaviour. Failures can also be caused by faulty inputs related to the human behaviour and human decisions that could be mitigated by training and experience.

SoS fails when its desired and expected emergent services deviate from the specification. There are different sources of SoS failures such as design errors, constituent systems failure, environment parameters, and undesired emergent behaviour. Constituent systems failures can be tolerated using fault-tolerance mechanisms such as active redundancy defined at design-time. However failure caused by emergent behaviour or unexpected environmental parameters can only be mitigated by adaptation at runtime. The main challenge then is the detection of undesired emergent behaviour and the prevention of ensuing SoS failures.

Predicting SoS failure: SoS is a very large system where failures can cause a great damage. Thus predicting the SoS failure and preventing its occurrence is often preferred than dealing with the failure after its occurrence. In the design phase it is hard to predict all possible SoS failures due to several challenges:

- **Complexity:** SoS is usually large complex system consisting of heterogeneous and geographically distributed systems. Autonomous behaviour of the constituent systems increases the complexity of the SoS and its operations. During its life cycle the SoS evolves over time by joining new systems while others leaving and by targeting new missions and goals. The SoS operation depends on the interference between different systems with inherent dynamic changes and evolution through the SoS life cycle.
- **SoS boundaries and environment:** The boundaries of SoS are often ambiguous and cannot be determined. Due to the constituent systems diversity and their dynamic interactions under different systems boundaries, it is difficult to observe and enclose the interactions between constituent systems, and the interaction between the SoS with their environment. The SoS environment is not clear too, the constituent

systems interact with their own environment and are considered as a part of the others' environment, which increases the SoS environment complexity. SoS keeps evolving over the time, considering constituent system evolution and technology adaptation, causing a dynamic changes on SoS boundaries and making them unstable. Constituent systems are also in dynamic changes; during the SoS life cycle, there are systems leaving the SoS and other new systems joining it, changing the SoS capabilities and offered services. The dynamicity of the SoS and its evolution over the time increase the uncertainty of its environment and its interaction with the environment. Another important property of the SoS that affects its environment uncertainty is the wide variety of its stakeholders due to its constituent system diversity.

- **SoS Life Cycle:** The system life cycle is defined as 'The evolution over time of a system-of-interest from conception through to retirement' (Haskins 2011). Monolithic systems have a clear life cycle that starts from the development of the system until it is out of commission. However, the definition of SoS life cycle depends on the type of the SoS and how it is constructed. Some SoS are constructed after the constituent systems realize that they can work together better under SoS environment, others are designed by the owner of the SoS in order to achieve a specified purpose. SoS that are constructed to achieve pre-defined objectives, the end of their life-cycle can be determined by achieving these objectives. For example, systems that work under a SoS in the military that is developed to achieve a specified mission, once the mission is ended, the SoS is no longer exist. In other SoS the life-cycle is not clear and there is no end for the life-cycle. In emergency case SoS, where the constituent systems work together to deal with emergency cases, there is no limitation on time. The constituent systems themselves have their own life cycle, some of them will be out of commission and leave the SoS while others will join. The SoS is typically subject to continuous evolution, it tries to deal with its diverse environment and so its life-cycle cannot be determined.
- **Emergent behaviour:** As mentioned before the emergent behaviour is a result of constituent systems interaction in the SoS and it cannot be predicted at design time. Therefore, run time monitoring of SoS behaviour is required in order to predict an imminent SoS failure. At run-time the environment parameters are more clear and stable within a short period of time. The following sections present more details for run-time monitoring of SoS behaviour and run-time analysis for failure prediction.

3. Imminent SoS Failure Detection Based on Simulation of Future Behaviour

State of the art architecture: The purpose of the architecture synthesis process is to construct a complete SoS model that supports the operational and behavioural analysis of the SoS. SoS model is used by a variety of stakeholders at the development stage of the SoS, e.g. system integrators investigating the interactions between con-

stituent systems, suppliers implementing constituent systems, and public authorities and certification bodies assessing the safety of the SoS.

The current state-of-the-art modelling approach for SoS is to use architecture frameworks that describe the SoS from different viewpoints. The mainly used frameworks in this field are US Department of Defence Architecture Framework (DoDAF) (DoD Architecture Framework Working Group 2003), British Ministry of Defence Architecture Framework (MODAF) (British Ministry of Defense 2010), and NATO Architecture framework (NAF) (Anon 2007). The Unified Profile for DoDAF and MoDAF (UPDM) (OMG 2010) was created by OMG group to enable modelling of SoS based on the DoDAF and MoDAF architectures. It supports the ability to model a wide range of complex systems at different levels of abstraction. UPDM has the capability to describe the operations and functions of SoS, but it does not support the specification of SoS behaviour and its constituent systems. UPDM introduces multiple views that depict different aspects of the system. It is useful in requirements elicitation, system components specifications definition, and constituent systems interfaces description and their interaction points. Using UPDM, the required functions to be achieved by the SoS operations can be defined. These functions are considered as the major point in selecting the constituent systems to achieve SoS goals.

Fig. 1 illustrates the model growth through the requirements elicitation and analysis process.

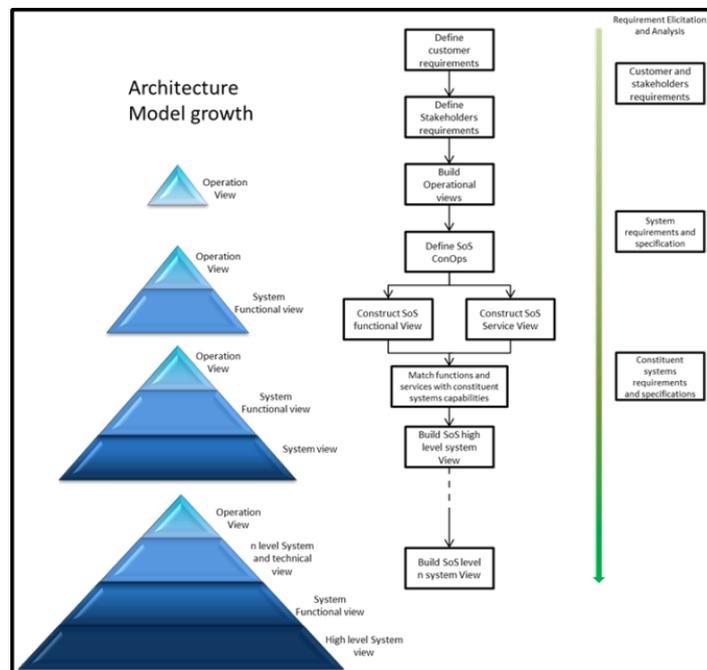


Fig. 1 SoS model propagation

The process starts from the customer side in order to collect the customer needs and convert them to requirements. Together with the stakeholders' requirements, the requirements are documented and analysed to build the first high-level specifications of the SoS. The first step in building the SoS architecture model starts with constructing an operational view which used to illustrate the operational aspects of the SoS. It gives a general view of what the system will achieve and through which operations. A scenario-driven process can be used in order to reduce the complexity of the operational flow construction process. At this stage the requirements are modified and new requirements are added and documented.

4. Model Extension

SoS and constituent systems behaviour: Dynamic analysis requires both the SoS behaviour and the constituent systems behaviour. At a high level of abstraction discrete behavioural models are efficiently used to describe the system behaviour. Discrete models are generated using SysML state charts. The state charts or state machine diagram describes the lifecycle behaviour of a system; it depicts the different states of the system and their transitions in response to events during the block life cycle (Friedenthal et al. 2006). As UPDM is an extension profile to SysML and UML, the SysML profile is integrated. Using the system view 10a(SV10a) in DoDAF within the UPDM profile, SoS behaviour using the state chart model is described. The same applies to the constituent systems where another SV10a is generated to include the constituent systems behaviour. As soon as we get all the specification of required systems behaviour, SV10a views are connected in the systems view, where the SoS architecture is built, and the constituent systems interactions are described. The next required step is to develop an extension profile that connects the operational and functional flow views with the SoS behaviour chart. This profile will make it possible to synchronize the SoS status with its operations and functions.

Operations constrains: The failure detection process depends on indicators and constrains defined at the operational level. It is required to define the critical constraints and map them to the SoS operations in the operational flow diagram to represent a behavioural reference for SoS desired behaviour. The failure detection process depends on monitoring these constraints and make sure that they are satisfied. If not, it indicates systems failure (deviation from the desired behaviour). It is possible to add these constraints as attributes within the UPDM SoS model for each operation. Constrains are used as indicators which facilitate failure detection using temporal logic definition.

Architecture Patterns: As described in (Kalawsky 2013) architecture patterns are used to facilitate the system reconfiguration and evolution. According to (Kalawsky 2013) Patterns could be used as a template for the structure and behaviour of the system. Using the system view from DoDAF under UPDM and by providing Constituent

systems to functions mapping, and constituent systems capabilities, possible architecture patterns are generated in the system view. At the design phase different patterns could be generated and trade studies between these patterns are implemented. Due to the SoS complexity and evolution, the wide variety for the constituent systems and their interaction, and the instability of SoS environment, one pattern is not sufficient for all SoS statuses and operations. For example, some of the patterns are cost effective, others are operational effective while others are time effective. The choice of which pattern to be used depends on the current and nearest future objectives and constrains of SoS.

By connecting the constituent systems to the generated architecture patterns we end up with executable model that includes the SoS behaviour as well as the constituent systems behaviour.

5. Run-Time Fault Prediction and Preventing Engine

The next step is to extend the model for analysis and failure detection. For this purpose a Runtime Fault Prediction and Prevention Engine (RFPPE) will be developed. The engine will be configured with the specification of the SoS and the constituent systems. The purpose of RFPPE is to monitor the SoS and its constituent systems behaviour. Figure 2 illustrates the main parts of RFPPE:

- Failure detection unit
- Next step simulation engine
- System reconfiguration
- External rules controller

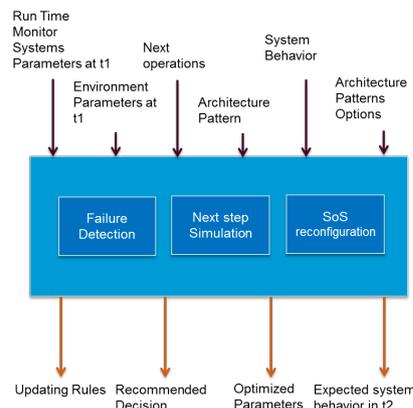


Fig. 2 RFPPE

The RFPPE targets directed and acknowledge SoS. Directed where SoS has a specific purposes for which it is built and managed to Achieve (Maier 1998) and the constituent systems share part of their ownership and funding. Acknowledge SoS has a defined goals and objective and its own management and resources while the constituent

systems keep their own independency regarding ownership, funding and goals (USA Department of Defense 2008) . In both cases when a constituent system decides to belong for the SoS it shares information about its environment, status, and capabilities with SoS management. RFPPE approach collects information about the SoS constituent systems states and parameters, the SoS environment parameters (collected from services systems e.g. surveillance , traffic management), the current state of the SoS (from SoS internal report) , the next step for SoS (from the operational view) , and the different options for the SoS architecture patterns that could be used. Depending on the current status and the coming steps, RFPPE provides the best option (that fits with the SoS goals and constrains, and prevents future failure) to be used while predicting the results using a short simulation for the next operational steps. Using the short simulation process, it predicts the future emergent behaviour that will affect the system operation and tries to mitigate or eliminate its bad effect by reconfiguring the SoS structure and its constituent systems parameters.

The purpose of RFPPE is to effectively:

1. Manage SoS operations: controlling the expected behaviour of SoS using rules
2. Enhance decision making process: by predicting the future results of decision options and providing the optimum.
3. Detect and Prevent SoS Failures: by monitoring the SoS behaviour, detecting failures, and preventing future failures.
4. Mitigate undesired effect of system emergent behaviour: responding to emergent behaviour by reconfiguring the SoS to reduce the bad emergent behaviour effect.
5. Prevent future undesired emergent behaviour effect for the next operational steps: reacting to predicted future emergent behaviour before its occurrence
6. Leverage, modify, and update system working rules: saving feedback information for failure handling process and SoS configuration to be used for repeatable failure modes.

Failure detection unit: Used to detect failures at SoS level by monitoring the SoS behaviour and constituent systems behaviour. UPDM model provides information about SoS operational and functional flow. At each operation SoS parameters and indicators are defined. For example assume an operation number x ($op(x)$) , Knowing that the expected value for system indicator A at $op(x)$ must be within the range of $y \leq A_{op(x)} \leq z$, failure occurs when the system indicator is out of its expected value. Failure unit detects such behaviour deviation and alerts the system for failure existence.

Next step simulation engine: The purpose of simulation engine is to predict the future behaviour of the SoS under certain configuration. By connecting the SoS model with SoS behaviour and the constituent system behaviour the future behaviour could be predicted by simulating the next step of the SoS under the current systems and environment parameters. The Operational flow indicates the next step of the SoS, and by extending the SoS model with the connection of functional flow diagram and the constituent systems behavioural model, the exact position of the current status is defined. Knowing the current status and parameters, next steps and operations, and cur-

rent systems configuration, SoS future behaviour is simulated and together with the failure detection engine, any future failure or undesired emergent behaviour will be detected and reported.

System reconfiguration: The system reconfiguration task is to reconfigure the SoS in the case of future failure is predicted out of the current configuration. Its purpose is to find the possible configuration patterns that could be used to overcome the current failure. The new configuration will be also checked for future failures by simulating the systems again with the new configuration.

External Controller: Within our SoS model failures are classified to:

1. Expected failures: failures that are expected at design stage and could not be avoided or the risk for failure elimination cost is more than failure handling. In this case the failure handling process and the required SoS reconfiguration for failure effects mitigation process are already defined.
2. Unexpected failure due to emergent behavior or external environmental impact.
3. Historical failure: failures that occurred before and handling process is already defined and saved.

The purpose of external rules controller is to increase the failure handling process speed and efficiency. It considers the expected and historical failures where predefined procedure and SoS reconfiguration are defined before and no further analysis required. Once unexpected failure occurs and handling process is defined, the controller will be updated and the failure conditions will be registered. Figure 3 shows the integration of RFPPE within the whole model.

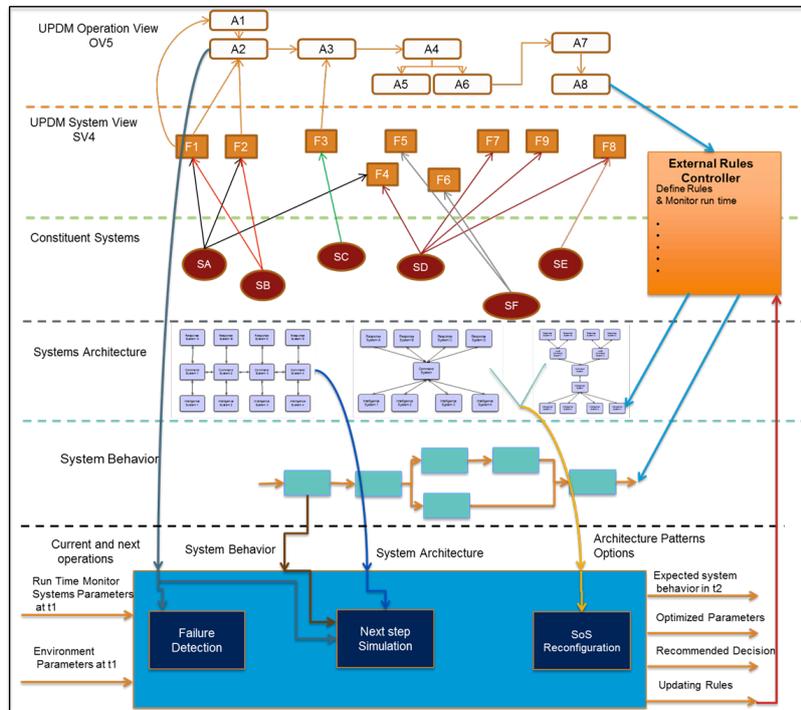


Fig. 3. RFPPE with model integration

Functional Description: Figure 4 describes the functional flow for the whole system. The run time operations are monitored for the current and next step of operation. If the system faces a known and expected behavior that is pre-defined in the rules, the constituent systems will be reconfigured according to the rules and the next step will be proceed. If the system faces unexpected or emergent behavior where there are no rules were defined, RFPPE will be lunched to choose the best configuration of the system that mitigates the bad effect of the unexpected behavior and prevents the system from running into further bad emergent behavior by predicting the next step results. The RFPPE collects all the information and parameters needed to describe the current status and look through available configuration patterns that could be used to deal with such situation. It will consider these patterns and run a simulation for the nearest next steps. By analyzing the simulation results and detecting any further emergent behavior that can be resulted from the chosen pattern, the System Reconfiguration part will suggest the best pattern to be used for the current situation and under the current environment parameters values.

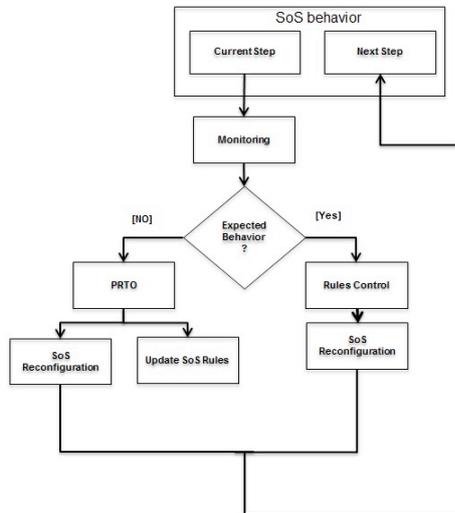


Fig. 4. RFPPE Process description

6. Example Scenario

In Emergency response system diverse systems collaborate together to provide emergency services for civilians. Command and Control Center (CCC) as a part of the emergency response system is a typical example of SoS where heterogeneous systems interacting with each other to efficiently mitigate the risks and consequences of emergency case. CCC as a system node coordinates the operations of other nodes i.e. Police headquarters, fire brigade, and medical units, and uses the emergent services out of this collaboration to deal with emergency cases. Figure 5

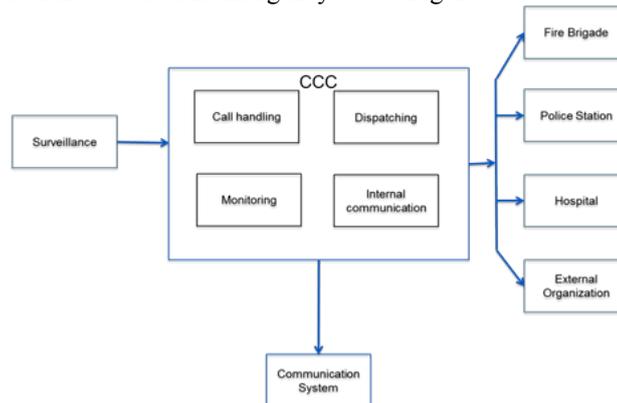


Fig. 5 Command and Control Center

Each of the system nodes represents an interaction unit and includes different operational nodes where autonomous and heterogeneous constituent systems work together to provide the required services and operations. Figure 5 depicts an example of different operational nodes (e.g. Call handling) that working together within the CCC. The operational nodes join the capabilities of diverse constituent systems within different activities to do its operations that require an interaction of hardware, software, services and organizational systems. CCC receives notifications about emergency case from external sources (i.e. people) through its call handling node. The call handling node gathers emergency case information from the callers and distributed surveillance systems and forwards it to the dispatching node where the emergency case information is analysed and the required emergency response is issued together with the dispatching plan. The dispatching plan is then distributed to the emergency units (i.e. Police HQ, Fire Brigade, Hospitals). The emergency units handle the emergency operations on the site and report back the site information to the CCC, which uses this information together with the surveillance information for monitoring the emergency case and taking the required actions and forwarding it back to the emergency units.

The communication systems provide the communication and data exchange services that connect the constituent systems together and facilitate their interaction. At a high level of abstraction there are two parts of communication; internal and external. The internal provides communication services for the constituent systems that are distributed within one system node, while the external one is responsible for the communication between the system nodes. For example, exchanging the emergency report between the call handling and dispatching unit is done using the internal communication system while exchanging this report with the police HQ is done using the external communication system.

Out of this constituent systems collection, there are many causes of emergent behaviour that could affect the emergent services of our SoS. Failure could occur at each of the constituent systems causing unexpected reactions of other systems. The same could happen where unconsidered behaviour by one of the systems resulting in a conflict for other systems. The environment parameters and environment changes represent a very important source for emergent behaviour where the SoS environment can't be defined at the design phase and thus not all the environment parameters are considered in design.

Communication systems failure, in our use case, considered as a critical failure that affects negatively the SoS behaviour and services. The constituent systems connectivity depends on the communication services that provided by the communication systems. Once the connection is lost, the SoS could not deliver its emergent services as the systems will be disconnected and their autonomous behaviour leads their interaction under the current situation. As an example of communication failure, consider an emergency rescue for people detained by fire in a building. Joint operations required between the police and firemen in order to evacuate and rescue the detained people. CCC is responsible for organizing this situation and issuing the required functions and dispatching plans for the police HQ and fire brigade.

There are pre-defined rules for this kind of operations where the number and kind of functions that must be sent to the site are defined. After the notification is received an explosion causes damage in the communication network between the CCC, PoliceHQ, and Fire Brigade preventing data exchange process between the systems at these nodes. At this kind of problem each of the systems starts to operate autonomously according to its pre-defined rules without knowing any information about the other systems. The dispatching systems at each node issue the required functions for the situation and try to fulfil the required operations by its own resources, which will double that functions that will be sent to the site. This behaviour causes chaos at the site and blocks the roads to the building due to the double number of rescue cars which consequently hinders the evacuation process resulting in loss of lives and increase the number of injured people.

The failure of the communication system could be detected by monitoring the data transfer rate between the constituent systems. By simulating the next step for the emergency response process, the deviation of the number of functions sent to the site from the required one implies undesired emergent behaviour out of the communication failure.

To avoid the undesired emergent behaviour, the CCC should change the communication pattern between the constituent systems to another one that could avoid the communication damage, and do the appropriate reconfigurations. One pattern could be used is to use the mobile communication unit that provides a temporal communication service at the emergency site and mitigates the risk of the communication failure. Another pattern could be changing the control and organization rules to local collaboration units that use the Radio communication system. The simulation results of the next steps after choosing one of the patterns ensure if a future emergent behaviour will occur out of using this pattern. If so, the pattern will be excluded and another one will be used.

7. Conclusion

The RFPPE approach is a method used to construct an adaptive robust SoS model that mitigates the effect of undesired emergent behavior, and offers a solution for reducing the SoS complexity by automatically chose the optimum design and pattern that fits for the current SoS situation. It can be used in two stages in the SoS development; Real time operations and Design phase under simulated environment. Different technologies are planned to be considered within the PRTO solution developments, Design Patterns, Design by Rules, and UPDM modeling as well as SyML modeling languages. Other technologies that could be part of the solution are the Agent based behavioral models that models the SoS evolution and game theory that enhances the model adaptability.

Acknowledgments This work was supported in part to European Commission for funding the Large-scale integrating project (IP) proposal under the ICT Call 7 (FP7-ICT-2011-7) ‘Design for Adaptability and evolution in System of systems Engineering (DANSE)’ .

References:

- Anon, 2007. *NATO ARCHITECTURE FRAMEWORK v3*,
- DoD Architecture Framework Working Group, 2003. DoD Architecture Framework: Volume I: Definitions and guidelines. , I(April 2007). Available at: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:DoD+Architecture+Framework+Volume+I:+Definitions+and+Guidelines#1> [Accessed February 19, 2013].
- British Ministry of Defense, 2010. MOD Architecture Framework. Available at: <https://www.gov.uk/mod-architecture-framework#use-and-examples-of-modaf>.
- Friedenthal, S., Moore, A. & Steiner, R., 2006. OMG systems modeling language (OMG SysML) tutorial. *INCOSE Intl. Symp.* Available at: <http://eng.umd.edu/~austin/enes489p/lecture-resources/SysML-Friedenthal-Tutorial-INCOSE2006.pdf> [Accessed October 23, 2012].
- Gorod, A. & Gove, R., 2007. System of Systems Management : A Network Management Approach. *System*, pp.1–5.
- Haskins, C., 2011. *Systems engineering handbook v3.2 ed.*, INCOSE. Available at: <http://smslab.kaist.ac.kr/Course/CC532/2012/LectureNote/2012/INCOSE Systems Engineering Handbook v3.1 5-Sep-2007.pdf> [Accessed May 22, 2012].
- Jamshidi, M., 2008. *System of Systems Engineering: Principles and Applications*, CRC Press.
- Kalawsky, R.S. et al., 2013. Using Architecture Patterns to Architect and Analyze Systems of Systems. *Procedia Computer Science*, 16, pp.283–292. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S1877050913000318> [Accessed April 8, 2013].
- Maier, M., 1998. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4), pp.267–284. Available at: <http://www.infoed.com/Open/PAPERS/systems.htm> [Accessed May 16, 2012].
- OMG, 2010. *Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF)*, Available at: <http://www.omg.org/spec/UPDM/2.0>.
- Sahin, F., Jamshidi, M. & Sridhar, P., 2007. A Discrete Event XML based Simulation Framework for System of Systems Architectures. *IEEE*, pp.1–7.
- USA Department of Defense, 2008. *Systems Engineering Guide for Systems of Systems*,
- Zhou, B. et al., 2011. Modeling system of systems: A generic method based on system characteristics and interface. *2011 9th IEEE International Conference on Industrial Informatics*, pp.361–368. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6034903>.

A different view on system decomposition – subsystem-centered property evaluation in multiple supersystems

Arne Herberg* and Udo Lindemann*

* Institute of Product Development, Technische Universität München,
Munich, Germany

Abstract This paper motivates the need for enhanced support for subsystem development and evaluation in the context of large engineering systems. Established approaches e.g. from the field of systems engineering seem to provide too little differentiation between how system development ideally should be approached and what “channels” (means) are realistically available for a company developing these systems. The research presented is motivated by the challenges identified in the context of the offshore petroleum drilling industry and the assumption that the challenges system providers face in that context are representative for other industries as well.

We propose a framework for addressing these challenges, aiming at supporting the evaluation basis for a subsystem by its projection into multiple supersystems and stakeholder systems. We specify the requirements differentiating different possible strategic and operational directions. We confront the identified requirements and potential directions to specific research areas, and established methods and tools, usually being applied in the context of overall system development or described only generally. We investigate which sub-aspects of the envisaged framework are being implicitly or explicitly addressed by these approaches and estimate their transferability potential. The identified potentials and limitations of the different approaches constitute the basis for a further substantiation of the framework.

1 Introduction

One of the main goals of system development can be narrowed down to the challenges of identifying which properties are considered valuable by the stakeholders and finding a solution to consistently incorporate these properties into a system. It is self-evident, that the difficulty of achieving that goal is closely related to the complexity of the system to design. Another domain largely determinative for the specific challenges of that task is the “value creating network“ (VCN) comprising the stakeholders involved in creating and maintaining the system and/or delivering their services using the system or parts of it. Furthermore, the VCN determines the ways in which stakeholders are contractually interrelated and the established mechanisms existing in particular industries constitute important boundary conditions for system development.

Moreover, it constitutes the basis for the interests and preferences of the involved stakeholders.

There are different overall approaches supporting the structured and organized development of complex systems as well as specific methods and tools supporting particular tasks or perspectives. Two of the most established overall approaches are systematic design [1, 2] and systems engineering [3, 4], both inheriting systems thinking as a very central aspect. While systematic design focuses on certain core principles and emphasizes a systematic approach to problem solving, systems engineering aims at an integrated consideration of the technical, social, and business aspects of a system. It constitutes a holistic, hierarchical decomposition approach to the design process, incorporating subsystem interactions, emergent functional behaviors and system integration [5]. As an iterative process of top-down synthesis it is considered to “enable the realization of successful systems” in a “near optimal manner” [3].

Nonetheless, these established approaches seem to provide too little differentiation between how system development ideally should be approached and what “channels” (means) are realistically available for a company developing these systems. The research presented is motivated by the challenges identified in the context of the offshore drilling industry and the assumption that the challenges system providers face in that context apply similarly for other industries.

Historically grown and established business structures and mechanisms within the VCN of certain industries can constitute obstacles to system providers to directly apply the principles of hierarchical decomposition and top-down synthesis on an overall system level. Different drivers have contributed to the fact that systems have grown somehow evolutionary and are still developed by reusing proven designs, explicitly accepting not to know how far away these systems’ properties are from a possible optimum. Examples for these drivers are conservatism and high investments on the one hand and complex and extremely time-critical tendering and bidding processes during system acquisition and specification on the other hand as shown in [6].

However, a fundamental part of the basis for overall system development is the portfolio of subsystems available within a company at that point in time, constituting more or less distributed modules when integrated in an overall system. In certain industries developing and deploying large-scale industrial systems, the development of these modules run decoupled from overall system level design. The term “design channel” is introduced in order to emphasize that considering design on different levels has to include the corresponding development phases and cycles as well as the respective development conditions. While subsystem development aims at customer neutrally updating or enhancing a company’s portfolio, overall system level design is essentially customer driven and thereby governed by tendering processes and other business related mechanisms. Subsystem development consequently must be considered by system providers as the only “design channel” providing realistic conditions for a sound (methodic) consideration and evaluation of the subsystems’ properties and its contribution to the various supersystems’ properties and their behavior (the term supersystem is very important in this paper and used – analogous to “subsystem” – referring to levels higher in the system hierarchy; the overall system is a supersystem

for each subsystem, but also elements on intermediate levels constitute supersystems for elements on lower levels).

Nonetheless, the perspective of subsystem development seems considerably underrepresented in the established approaches, handling subsystem development as an integrated part of system development rather than as a discipline with very specific challenges and potentials and a completely differing starting point and problems to solve.

Method and structure

In order to further clarify the outlined challenges, in the next chapter we summarize the situation acquired in an in-depth case study from the offshore drilling industry based on industrial publications (e.g. [7-9]) as well as on several workshops with a system provider (including experts from management and business development but also experts with operational experience) and additional semi-structured interviews with experts from further core stakeholders of the VCN: two with an oil company (operator) and one each with experts from a main drilling contractor and a ship yard. We illustrate the importance for the system supplier to enhance systems thinking especially during subsystem development, and thereby to exploit this design channel more consciously. Our core interest is directed to the multi-faceted domain of properties on different hierarchical levels of the system, the role of property types, the question of dependencies and aggregation as well as the subjective interest or “value” related to properties from different stakeholders’ perspectives. Focusing on subsystem development as the selected design channel, the dependencies between properties have to be considered not only in one specific supersystem but in different relevant supersystems. The differentiation of supersystems has to apply on the one hand within one overall system (e.g. different subsets of subsystems, contributing to certain technical main processes), on the other hand as a differentiation of overall systems (as supersystems) themselves. Additionally, variations in the stakeholder network can result in different possible sets of stakeholder interests related to these properties.

In the third chapter we propose a framework integrating these requirements, aiming at supporting the evaluation basis for a subsystem by its projection into multiple supersystems and stakeholder systems. We confront the identified framework elements to specific research areas, and established methods and tools, usually applied in the context of overall system development or in a non-specified context. We investigate which sub-aspects of the envisaged framework are being implicitly or explicitly addressed by these approaches and estimate their transferability potential. The identified potentials and limitations of the different approaches constitute the basis for a further substantiation of the framework.

2 In-depth case study for systematic clarification of the problem

In the value creating networks (VCN) of the offshore drilling industry, numerous stakeholders with very different expertise and economic power contribute with their

systems and services to the achievement of the overall objective of drilling a well in order to detect oil or gas reservoirs, create access to them and assure exploitability completing the well with the required installations. The large-scale complex drilling systems deployed for these purposes from the water surface have to enable very different operational processes (OP) under the water and in the formation under the seabed such as drilling, measuring, pressure control or stabilizing the borehole by cementing casings into it.

The hierarchy within drilling systems

The drilling systems consist of numerous interacting subsystems (some of which can be seen as modules) arranged on and integrated into the hull of a floating platform or ship. Different subsets (“operational process systems” – OPS) of interacting subsystems are needed for different OPs, to a large extent linked to the transport and mounting/dismounting of very different functional elements (e.g. drill bits, drill pipes, measuring equipment, huge valves, etc.) needed in the borehole as well as their electronic, mechanical or hydraulic actuation. The OPSs for the different OPs are not independent and decoupled but highly overlapping. Rigging and adapting subsystems when changing from one OP to another is often necessary. The use of a subsystem for different OPs leads to reduced space and weight consumption, being a crucial issue for these systems, as well as to potential investment reductions. On the other hand this limits the possibilities of concurrent execution of OPs, and increases the importance of reliability and durability. Optimal system performance is thus depending on properties across all hierarchy levels from the overall system architecture, over the OPSs to the subsystems. Nonetheless, in reality, these levels are addressed over different design channels, and not in an integral, top-down system synthesis process.

In this sense, similarities to the field of systems of systems (SoS) exist. On the other hand some of the key characteristics of SoS do not apply to the described class of systems such as operational independence (subsystems achieve well substantiated purposes even if detached from the SoS), managerial independence (subsystems are developed and managed for their own purposes).

Stakeholder roles, constellations and perspectives

The result of successful system development is the embodiment of the set of properties that bears the most value. Besides the technical challenge of incorporating these properties into a system, the question of the most valuable set of properties will lead to different answers depending on the stakeholder. Per definition, in business environments, the stakeholders’ major interest into the properties of a system is how they affect the profitability (long-term or short-term, depending on their strategy) of their business (which doesn’t mean that they have a clear judgment on the effects).

In the drilling industry, a high number of stakeholders contribute to overall value creation. Major stakeholders and their typical tasks are

- (SP) the system provider: responsible for designing and manufacturing the drilling system

- (OP) the drilling operator (oil company): possessing the rights to drill and exploit the resources in a defined area
- (DC) the main drilling contractor, being engaged by the operator for the execution of the drilling services – main user of the drilling system
- (SY) the shipyard, constructing the hull and integrating the drilling system
- several other stakeholders such as the hull designer, suppliers of subsystems, special equipment or consumables, sub-contractors for special services, etc.

According to the stakeholders' roles and their interfaces to the system, different properties are relevant for them. Their priorities and preferences regarding these properties' values are often conflicting as also stated in [10]. The stakeholders form a value creating network (VCN) whose structure results from the existence of business relationships amongst each other. But not only the structure of the stakeholder system (fig. 1, right) is relevant, also the specific relation type, meaning the agreed obligations (constituting cost and risk) and remuneration principles.

For the system provider it is essential, if the drilling contractor (DC), the operator (OP) or the shipyard (SY) is his direct customer, which refers to the structural dimension. Possible value related properties relevant for these stakeholders can be high reliability and availability (for DC), high time efficiency (for OP), or low equipment and engineering cost (for SY). But for the DC, time efficiency can gain relative importance compared to reliability if incentives are integrated in the remuneration such as being rewarded by meters drilled per day instead of fixed day rates – this refers to the dimension of relation type. Both dimensions together are referred to as the “stakeholder constellation”.

Incremental development and design channels

A lot of systems deployed in the drilling industry are far away from providing an optimal behavior, which on the one hand has to do with a high uncertainty in various domains over the lifecycle of a drilling system, e.g. related to changing operational contexts or market aspects as elaborated upon in [6]. On the other hand, we identified several industry inherent triggers (simplified):

- Due to strict safety requirements, sticking widely to proven system designs with medium performance finds more acceptance than aiming at radical improvements with higher risk and certification effort.
- High investment costs for the development of completely new concepts.
- Based on fast growing requirements with respect to higher safety, higher water and drilling depths, wider functional scope and areas with more extreme natural conditions (e.g. drilling in the arctic), efforts have been concentrated on extending established system designs' absolute capabilities, which did not provide the room for holistically re-thinking the overall system.
- A growing variety of system designs due to more radical changes would constitute less flexibility regarding the allocation of operating and maintenance staff (respectively higher training efforts)
- The resulting evolutionary, incremental development of the overall system designs is also mirrored on subsystem level, where clear modules for certain functions have

evolved over time. Making radical changes on the system level would also necessitate breaking up some of these established modules, which would again be very costly and enhance the proneness of failure.

- Usually, an overall system design is being proposed as a reaction to a call for tender. The fact that the system provider usually has to bid within a very short period of time makes it virtually impossible to come up with a solution resulting from a systematic decomposition of the design problem. Consequently, the one existing solution being closest to the required specifications is selected and adapted. Often customers even specify their demands explicitly referring to an existing design.

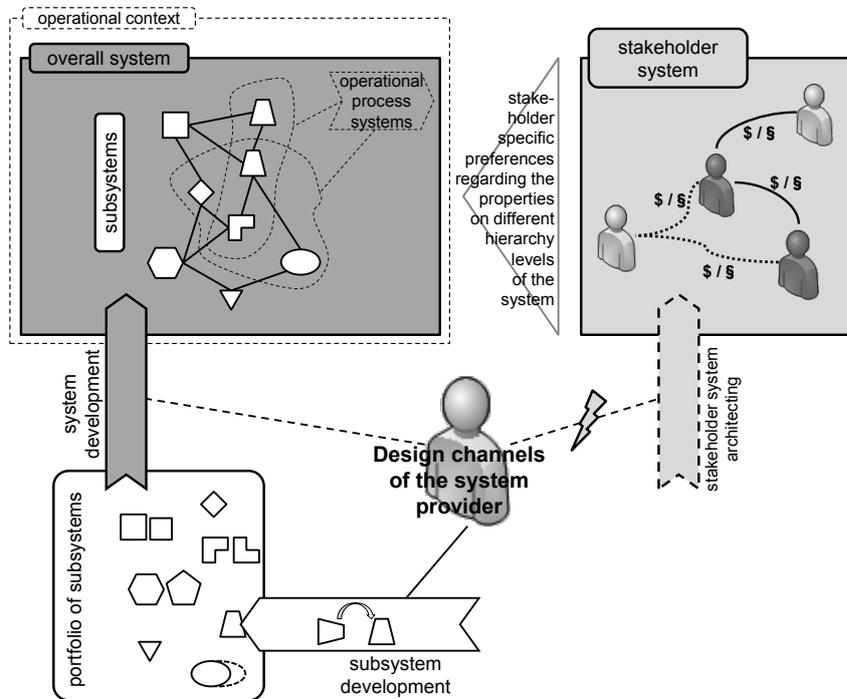


Fig. 1 Different hierarchy levels of a drilling system and related design channels of the system provider (left). Stakeholder system with established roles and constellations – determinative for property preferences but no possibility for the system provider to take influence (right).

Even for ambitious and innovation oriented system providers these factors constitute essential obstacles for challenging established designs and approach new concepts holistically, thus over the design channel of system development. Under these circumstances the strategic meaning of the design channel of subsystem development has to be emphasized. Not only can subsystem development be driven based on internal business cases independently from customer tendering processes. Also, forming the (incrementally developed) portfolio and thereby the building blocks of the future overall systems, the (incrementally developed) subsystems constitute the actual drivers of

future system designs and not vice versa as proposed as ideal approach by systems engineering.

Nonetheless, in our case study we observed a lack of systems thinking in the context of subsystem development. This has been derived from the analysis of different examples of recent subsystem developments and their acceptance on the market. Generalized, problem solving has been too much focused on the main technical objectives, having been achieved very successfully. At the same time, often developments were below expectations as important side effects in other domains have not been identified. Examples are:

- Focusing on the preferences of one stakeholder (even though he is the initiator of a development project) without evaluating a development's resulting property set from the perspectives of other stakeholders can retaliate if those are potential customers of overall systems as well.
- Limiting verification to properties on the subsystem level without estimating their (emergent) effects on higher hierarchy levels.
- Limiting considerations on higher hierarchy levels to single supersystems:
 - The relative importance of a subsystem's reliability depends on the question if the supersystem provides redundancy for its function.
 - Eliminate weaknesses for one OPS can imply essential new weaknesses for another

In the next chapter we specify the need for approaches supporting systems thinking explicitly in the context of subsystem development and evaluation and propose a conceptual framework derived from these needs.

3 The needs for a multi-supersystem evaluation framework

As we have shown, certain industries entail boundary conditions that constitute obstacles for systematic development on the overall system level, so that the design channel of subsystem development gains importance in order to systematically introduce improvements and guarantee competitiveness. Nonetheless, introducing changes on the subsystem level, it becomes all the more critical to consider that the effects space of a subsystem design is larger than the design space itself, and the supersystems are of multiple nature.

Numerous authors emphasize the importance of early validation and verification [3, 11] and pinpoint at the risk of high market losses due to launching decisions without appropriate evaluation activities [12]. With the framework proposed in the following and its further development we want to contribute to this field especially with respect to an enhancement of transparency of the different effects the changed properties of a subsystem can have on other hierarchy levels, in the context of multiple possible supersystems and under consideration of varying sets of stakeholder preferences.

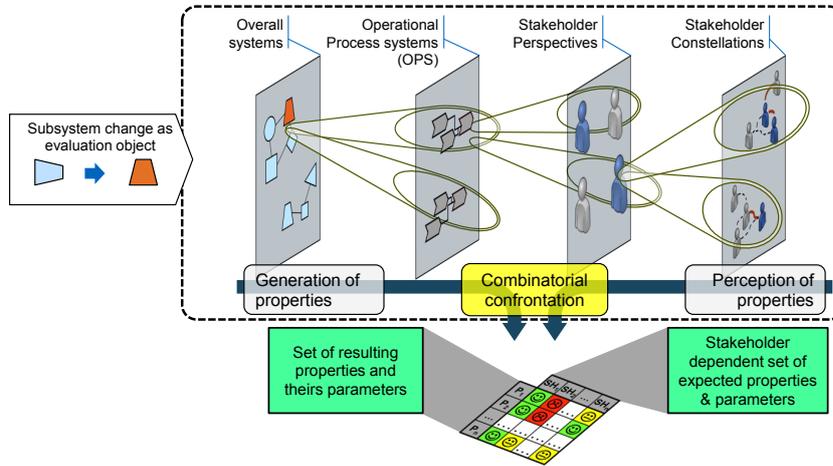


Fig. 2. Basic structure of the framework

As shown in fig. 2, the bottom-line of the framework is the confrontation of the two sides of the “objective” generation of properties through the chosen subsystem design (represented as a change from a former design) and the “subjective” perception of these properties by the stakeholders. The first level of the evaluation problem is based on a selected overall system and a selected stakeholder constellation, the subsystem is projected into. The layer of OPS (see also fig. 1) enables a systematized inquiry of the subsystem’s properties’ effects in the context of the different operational processes it contributes to.

In order to address the evaluation problem holistically, appropriate variations have to be made in the layers of the supersystems (overall systems as well as OPSs within each overall system) as well as the stakeholder constellations, resulting in a set of confrontation results based on their combinatorics.

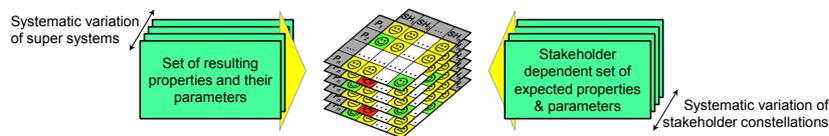


Fig. 3 Combinatorial confrontation

The framework aims at enhancing the transparency regarding the overall picture of confrontation results as well as at supporting the identification of otherwise neglected discrepancies. It shall thereby substantiate the basis for assessment and decision making where the results have to be interpreted based on defined strategies. But which aspects of related approaches and methods can be picked up in order to substantiate this framework? Are their basic ideas compatible and transferable to a subsystem-centered approach? The next chapter discusses some of them with respect to their potential to support possible directions of further development of the framework.

4 Potential and limitations of related approaches

Stakeholder value perspectives and stakeholder networks

There are different concepts of linking system or engineering parameters to subjective stakeholder preferences such as QFD [13] (see below) or value measurement using key parameters (KPs) [14, 15]. The latter provides a quantification method based on stakeholder specific subsets of weighted KPs and the derivation of a total value weighting stakeholders by relative importance. The approach also covers the KPs evolution over time (e.g. due to changing preferences or new technologies) which should be considered as an important perspective for our framework as well. Nonetheless, the interdependencies between the KPs respectively the questions which KPs can be actively influenced are not addressed.

In the area of stakeholder networks many approaches focus on high level systems architecting [16, 17] including the architecture of the stakeholder network itself. From the point of view of a drilling system provider, the stakeholder system has to be dealt with as a boundary condition as no influence on it is given (see fig. 1). Nonetheless, as shown above constellation variations have a high influence on the perception side of the framework, and modeling tangible and intangible value flows [16] can help deriving stakeholder preferences of the system's properties.

Property-based approaches and dependencies between different types and hierarchy levels

The dependencies between attributes directly designable and measurable and attributes resulting from their aggregation (the latter usually being those of interest for the customers and other stakeholders) are the core of many theories and approaches supporting different objectives. Examples are CPM/PDD [18], differentiating between characteristics and properties, or axiomatic design [19], where design parameters are translated into functional parameters. A more detailed differentiation based on the aggregation mechanisms to higher levels in the decomposition is provided in [5], listing (in order of increasing complexity) attributes which aggregate (1) depending on system composition (e.g. mass), (2) system structure (e.g. cost), system operation (e.g. reliability) or (4) resulting from complex emergent behavior (e.g. passenger wait times for a train system).

The house of quality (HoQ) – a largely established visual support developed in the context of the method QFD [13] – allows to allocate engineering characteristics (EC) to customer attributes (CA) as well as to qualitatively represent their direct relations to other engineering characteristics. This supports the reflection on direct and simple indirect consequences of changes of ECs for the CA. Nonetheless, complex aggregation mechanisms cannot be covered by that approach. The fact that the customer speaks with a “common voice” also does not allow for the consideration of conflicting interests – covered conflicts are thereby limited to system inherent “technical” conflicts. Furthermore, the variation of supersystems or interfaced subsystems is not supported.

Lifecycle properties also referred to as “ilities” (e.g. maintainability, safety) constitute another essential group properties “that often manifest themselves after a system

has been put to initial use. [...] they do not include factors that are always present, including size and weight” [20]. DfX-guidelines are valuable sources to identify links between parameters on lower levels and lifecycle properties [21].

5 Discussion

Subsystems driven design in a way conflicts with system engineering’s main principles of top-down synthesis where subsystems result from an explicit decomposition process, and decisions on system and subsystem level can be reflected in both directions, based on the increasingly precise estimation of the resulting properties and their aggregation [5]. Nonetheless – as shown in our case study – in certain industries the design channel of subsystem development provides more potential for systematic development than the design channel of overall system design.

Therefore, the presented framework aims explicitly at supporting the design channel of subsystem development, especially trying to respond to the differing challenges with respect to the resulting evaluation problems. On the one hand, the system of interest [3] (and thus the object of evaluation) becomes smaller in scope meaning also a reduced number of variable parameters in contrast to a holistically designed system. At the same time, the effects space (the overall system) has to be considered in multiple relevant variations in order to reduce the risk of critical discrepancies in the form of the number or severity of mismatches between (sub)system properties and stakeholder preferences.

The framework provides a wide range of potentials from the evaluation of a subsystem design concept to the analysis of existing subsystems in order to derive development goals. Besides, it enhances the understanding of the own systems and their properties, e.g. addressing the question which mechanisms can be found that explain why a property gains or loses importance? Fig. 4 outlines exemplary possible analysis objectives:

- Analysis objective 1 – How does the overall picture of the perceived value that results from a subsystem change in a given overall system change as a function of the underlying stakeholder constellations?
- Analysis objective 2 – How does the perceived value of a specific stakeholder resulting from a subsystem change in a given overall system change as a function of the underlying stakeholder constellations?
- Analysis objective 3 – How does the perceived value of a specific stakeholder in a given stakeholder constellation change depending on the type of overall system a changed subsystem is integrated in? And how can be known which aspects of that variation can be related to the subsystem?

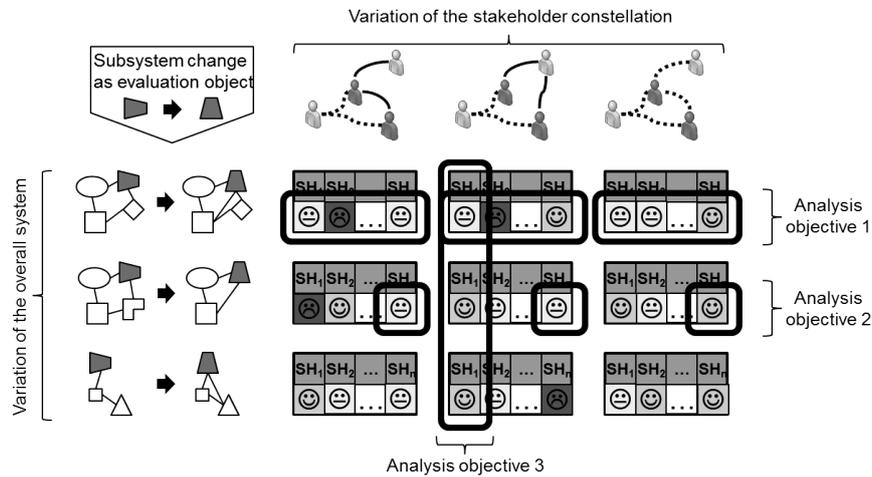


Fig. 4 Exemplary analysis objectives supported by the framework

In all of these cases, average values, variances and outliers might be of relevance, depending on the decision to take.

6 Outlook

For the moment, the framework aims at contributing to the field of approaches supporting the organization of information. It integrates ideas from related approaches such as CPM/PDD [18] or QFD [13], and enhancing these approaches towards some of the identified missing aspects seems feasible such as representing the effects of varying supersystems or the differentiation of stakeholder perspectives.

On the other hand important questions remain unmentioned or unanswered by these approaches, e.g. how can be assured, that all relevant properties have been considered? Although CPM/PDD provides for the integration of “additional properties” [18] – properties that haven’t been originally considered and are identified in the course of the design process – their identification is not supported systematically. Also in simulation approaches applied to estimate the aggregation effects of properties, the considered parameters need to be predefined [5].

Another topic to be addressed in future research is the question of how to support the selection of supersystems for the scenario building and how to integrate the anticipation of future changes on system level. A differentiation between the subsystems’ integration in new systems and the replacement in upgraded systems also has to be investigated with respect to effects on the framework’s requirements.

At this time, we have not completed a detailed application case of these ideas. Nonetheless, we view this framework as a platform for research rather than a finished product. It has many interfaces to related approaches and combines the challenges of

different other related problems, some of which are not satisfactorily solved and to which this research intends to contribute.

References

- [1] G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote, *Engineering design: a systematic approach*, 3rd ed. London: Springer, 2007.
- [2] VDI2221, "VDI Guideline VDI 2221 - Systematic approach to the development and design of technical systems and products," ed. Berlin: Beuth-Verlag, 1987.
- [3] C. Haskins, Ed., *Systems engineering handbook: a guide for sytem life cycle processes and activities*, 3.2 ed. San Diego, CA: INCOSE, 2010.
- [4] B. S. Blanchard, *System engineering management*, 4 ed.: John Wiley & Sons, 2008.
- [5] J. M. Aughenbaugh and C. J. J. Paredis, "The role and limitations of modeling and simulation in systems design," in *2004 ASME International Mechanical Engineering Congress and Exposition*, 2004.
- [6] D. Allaverdi, A. Herberg, and U. Lindemann, "Lifecycle perspective on uncertainty and value robustness in the offshore drilling industry," in *2013 IEEE International System Conference (SysCon 2013)*, Orlando, USA, 2013.
- [7] C. Moomjian, "Drilling Contract Historical Development and Future Trends Post-Macondo: Reflections on a 35 Year Industry Career," in *IADC/SPE Drilling Conference and Exhibition*, 2012.
- [8] J. Hother, "An analysis technique for optimising reliability in system design," in *SPE Annual Technical Conference and Exhibition*, 1999.
- [9] P. Osmundsen, T. Sørenes, and A. Toft, "Drilling contracts and incentives," *Energy Policy*, vol. 36, pp. 3138-3144, 2008.
- [10] R. Stevens, "Profiling Complex Systems," presented at the SysCon 2008 - IEEE International System Conference, Montreal, Canada, 2008.
- [11] C. M. Eckert, P. J. Clarkson, and W. Zanker, "Change and customisation in complex engineering domains," *Research in Engineering Design*, vol. 15, pp. 1-21, 2004.
- [12] R. Bernard, "Early Evaluation of Product Properties within the Integrated Product Development," Dissertation, Lehrstuhl für Produktentwicklung, Technische Universität München, München, 1999.
- [13] J. R. Hauser and D. Clausing, "The House of Quality," *Harvard Business Review*, May-June 1988.
- [14] E. C. Honour and T. R. Browning, "Dynamic optimization of systems of systems using value measurement," *Journal of Integrated Design and Process Science*, vol. 11, pp. 33-53, 2007.
- [15] T. R. Browning and E. C. Honour, "Measuring the life-cycle value of enduring systems," *Systems Engineering*, vol. 11, pp. 187-202, 2008.

- [16] E. Arnautovic and D. Svetinovic, "Value models for engineering of complex sustainable systems," *Procedia Computer Science*, vol. 8, pp. 53-58, 2012.
- [17] D. J. Nightingale and D. H. Rhodes, "Enterprise systems architecting: Emerging art and science within engineering systems," in *MIT Engineering Systems Symposium*, 2004.
- [18] C. Weber, "CPM/PDD - An extended theoretical approach to modelling products and product development processes," presented at the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes, Berlin, Germany, 2005.
- [19] N. P. Suh, "Axiomatic design theory for systems," *Research in Engineering Design*, vol. 10, pp. 189-209, 1998.
- [20] O. L. de Weck, D. Roos, C. L. Magee, and M. Charles, *Engineering Systems: Meeting Human Needs in a Complex Technological World*: MIT Press, 2011.
- [21] C. Hepperle, W. Biedermann, A. Böcker, and U. Lindemann, "Design for X-guidelines and lifecycle phases with relevance for product planning – an MDM-based approach," presented at the 13th International DSM Conference, Cambridge (USA), 2011.

Simulation for all components, phases and life-cycles of complex space systems

Fernand Quartier, Frédéric Manon

Spacebel, Technoparc 8, Rue Jean Bart, 31670 Labège, France
fernand.quartier@spacebel.be
Centre National d'Etudes Spatiales, 18, Avenue Edouard Belin, 31401 Toulouse, France
frederic.manon@cnes.fr

Abstract. This paper describes the use and evolution of discrete event simulators and models throughout CNES, its various space system developments, disciplines and related life-cycles and teams. Simulators and models are built in the first place to ensure that the organization improves its competences in a number of key areas. It presents how a careful federation of means, know-how and models using a bottom-up approach, will meet one day the top-down System of Systems approach.

Keywords: Discrete event simulators, life cycle, multi-disciplinary, functional simulation, space systems

1 Introduction

In a large engineering enterprise, such as Centre National d'Etudes Spatiales (CNES), there are many simulators used and developed. The most demanding is the operational simulator as it has to be representative for a satellite as seen from the ground and because it is used in many verification and qualification chains for control centres, mission control centres and payload control centres. For those qualifications, the real satellite is only used rarely as it incurs very expensive operations with many constraints, while introducing risks on damage and planning. Moreover, testing with real satellites still has limited representativity and fault injection is even more cumbersome. Nowadays, the operational simulators fly many months before the satellite is launched.

The significant efforts to develop such large operational simulators have not only led to a better understanding of the problematic and to better technical solutions, as described in subsequent sections. It equally triggered the awareness of the value of models that contain part of the company's memory and its patrimony and a means of communication and specification of behaviour. The validation and qualification of models takes often much more resources than the development itself, so that reuse is much more rewarding than traditional reuse of software components. But most importantly, models and simulators are creating some sort of biotope that allow improv-

ing key competences and facilitates cooperation between people having various expertise and project roles.

2 Operational Simulators

2.1 Main Requirements

Operational simulators have the following key requirements:

- From the point of view of operators, the simulator should be indistinguishable from the real satellite
- Causality must be respected and all runs must be reproducible
- Failure, fault and reproducible noise injection without changing models
- Fine control and visibility on internals (introspection)
- Formal and automated procedures for model and simulator validation
- Save/restore of context to allow bypassing operational test lead-in times of several days
- Perennity guarantees for 15+ years: Linux, mainstream PC's, Open source versus COTS, heritage/reuse of 15 years

2.2 Content and Performance Requirements

- Independent models in C, Fortran, Matlab, Scilab, object format (industrial secret).
- Start script based model instantiations and connection of model variables without compilation (using naming database)
- Computer emulators are loaded with the production version of the ROM images (1750, ERC32, LEON, ...)
- Performance: minimum is guaranteed real-time, 3 to n times real-time for increased productivity

Although the main content of an operational simulator revolves around its computer simulator, many disciplines are present: on-board software, command and control, guidance and attitude, mechanics, thermal, electric, power ...

As an example, the Pleiades operational simulator contains:

- 200 models, model frequencies of 1 to 128 Hz
- 7 processor emulators, globally up to 80 million of OBSW instructions/sec
- Up to 200 events in scheduler
- 10.000 events per simulated second

Its performance is:

- minimum 2 times real-time
- 10 times real-time preferred (possibly with models that support reduced representativity)

Simulation for all components, phases and life-cycles of complex space systems

– 100.000 events per executed second

The Argos study simulators contain 100.000 models and manage 200.000 events in the scheduler. They run 5 to 500 times the real-time speed, executing 500.000 events per second.

Large simulators tend to have separate specialized teams to

- Develop and validate models, covering various disciplines (mechanics, thermal, power, dynamics, ...)
- Configure, integrate and validate simulators for the specific needs
- Deploy simulators for use in the various operational chains and execute the needed scenarios

2.3 Life Cycles

The life of a satellite simulator has many dimensions as can be seen in the pictures below.

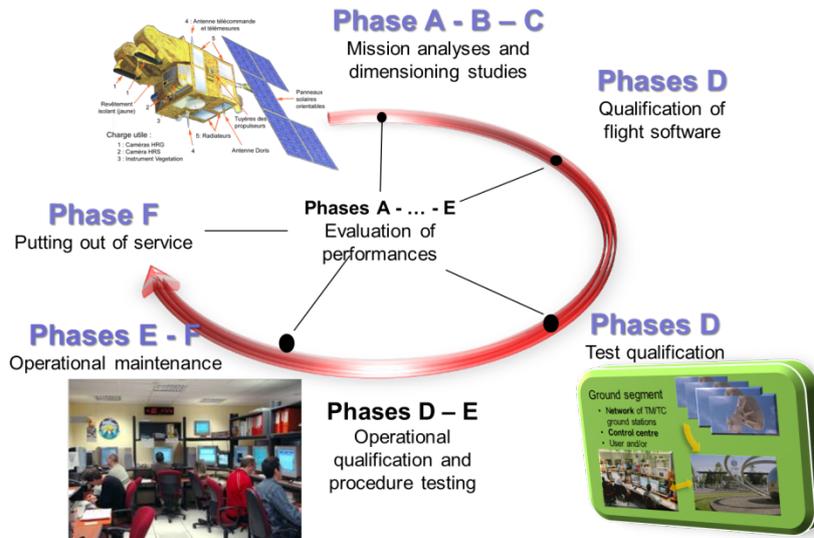


Fig. 1. Life cycles of a spacecraft

Simulation for all components, phases and life-cycles of complex space systems

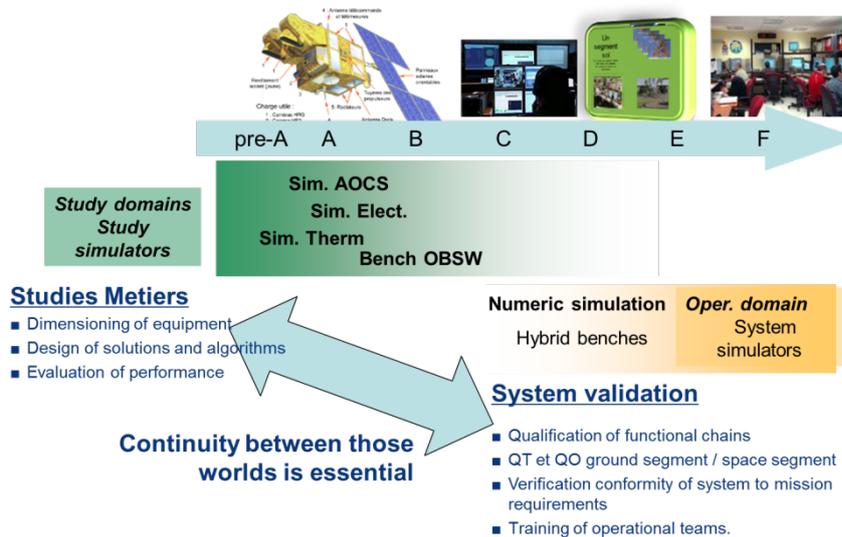


Fig. 2. CNES main simulator needs during a spacecraft life cycle

Other dimensions along the project phases are:

- Instruments that range from simple acquisition subsystems to complex instruments, such as GPS, star trackers, Gyros, ...
- The several space platforms for the various product lines (mini-satellites, micro-satellites)
- Within operational phases, different configurations of the simulators are used, called variants. Typically, representivity and scope is reconfigured as to provide optimal performance for the tests at hand.

All these dimensions need a well thought out approach for testing, validation, configuration management and maintenance.

2.4 Integration with Other Components

Obviously, operational simulators need to have flexible interfaces to connect with the control and operational centres. It must be possible to route those interfaces directly or via the receiving station, through real RF equipment or through Spacelink simulators when representivity is paramount.

Co-simulation with other specialized simulators, such as Saber, is achieved through the use of standard interfaces, such as HLA. In the long run, hardware-in-the-loop will be needed for some components such as instruments and payloads.

3 Towards Better Use and Continuity of Means.

The development of operational simulators is on a crossroad where many project phases, disciplines, models and people come together. Nevertheless, it was observed that the re-use of models, know-how and tools was far from optimal. So it was further investigated.

3.1 Identified Problems and Barriers

The main identified problem is due to the partitioning barriers caused by the many dimensions of the life cycles, project teams, disciplines, platforms.

Building a simple discrete event simulator is not that complex, so that there are many such simulators developed throughout the company. As usually with software, those simple simulators evolve quickly to more complete in-house products and test environments. The more they evolve, the less the models tend to be reusable and the more difficult it becomes to move to a common platform.

3.2 BASILES

To improve the situation, in a first phase, BASILES (BAncs SIMulateurs et Logiciels d'Etude de Satellite) has been created. It is a common simulation platform to promote models and simulation reuse among space programs and among the different simulators that are created during the lifecycle of a project. BASILES provides a methodology and a standard for CNES simulators.

First of all, BASILES is a simulation framework to develop, configure and run simulators. It allows representing complex systems using discrete event simulation. It contains the simulation kernel in charge of time and events handling, logger service, integrators, processor emulator management, distributed simulation handling, etc.

Concerning the development of a new simulator, BASILES features help to easily develop prototypes with basic programming knowledge in a short period of time and with a good level of accuracy. Models are simply configurable. Concerning the execution of a simulator, BASILES provides a great number of self-functionalities to interact and introspect the models and simulation.

Finally, BASILES is also a model library in order to share and reuse models and simulators among space programs.

In order to extend its user base, CNES accepts to attribute licenses of the product to other industries, thereby stressing the system more to achieve quicker full maturity and to expose the product to new user requirements and ideas.

3.3 SMP2

For several years now, the European Cooperation for Space Standardization (ECSS) has taken the initiative to develop the SMP standard (Simulation Model Portability). The aim of this standard is to allow models to be portable among different simulation

infrastructures. Interfaces are specified by SMP independently of simulation infrastructures.

BASILES evolved to the SMP standard and all new developments are SMP based.

3.4 Study Simulators

One of the other families of discrete events simulators is MACSIM, basically used as study simulator, and having a large patrimony of existing models. Study simulators tend to be developed starting with few and relatively basic models in an incremental and iterative way: the developer improves or refines the model, runs it, validates it and restart improving it. The MACSIM environment has been successfully integrated in BASILES.

3.5 Hardware in the Loop

Ideally, many of the models should be replaceable by hardware equipment, although this adds significant constraints. This allows using real equipment, to raise the level of representativity and expose the used models to a broader range of environments.

Such operations have been successfully performed integrating real payloads with the simulator via Mil 1553. The new Nosyca balloon flight computer has been integrated with BASILES via a number of interfaces. In that case, BASILES became a test bench, environment simulator and controller of the Nosyca flight computer.

3.6 Software Validation Facilities

BASILES has been augmented with non-intrusive flight software gdb debugging capabilities on the used processor emulators. That means that breakpoints can be set on specific instructions or data accesses. When such a breakpoint is hit, the clock of the processor and the simulator is frozen and the gdb interface is warned and normal debugging can take place. All external BASILES interfaces remain functional and time progress continues when the processor is released by the debugger.

3.7 Towards New Generation of Modular Real-time Benches

A demonstrator has been build that shows the distributed real-time capability of modern systems. It runs BASILES simulators on different mainstream PC's running standard Linux connected via HLA. Measurements have shown that all simulators were capable of generating output with a time precision and jitter that is better than 50 µseconds.

It is believed that test systems will become more modular and cheaper. In fact, many of the typical test systems are based on huge acquisition and driving front-ends, along with custom interfaces and uncommon processors and real-time RTOS with

specialised drivers. Such complex equipment creates a major constraint on re-use, maintenance and perennity.

For the Nosyca system, interfaces have been made using a series of small micro-controllers such as the PIC32, complemented with the needed connectors and small interface logic and shaping. These little 50€ low power boards (power via USB), with a 20 cm² footprint, contain a 80 MHZ CPU, significant memory and interface variety, including Ethernet. Because the microcontrollers are dedicated to one single function, they are simple, while in many cases, specific interface FPGA's can be avoided as the microcontroller can achieve a time resolution well below the μ second. An approach that uses multiple small systems is better manageable than huge complex and hierarchical systems.

3.8 Defining Simulator Strategy. at Day One of Each Project

From the many experiments and domains BASILES has been used in, it became clear that a complete simulator planning is better studied by the very beginning of each space related project. As has been shown in the Argos and SMAR project, a first global system simulator allows for better dimensioning of many components of the system and helps to create a common understanding of the project.

4 On-going Developments and R&D

4.1 Thematic

There are several R&D projects and investigations going concerning microscopic traffic simulation (one model per car), Software Validation Facilities for Proba and MTg, missile test planning, FDIT management, TDM space communication and improved thermal simulation.

Indeed, precise thermal simulation used to be extremely processing hungry. CNES is in the process of developing fast thermal simulation technology that will allow simulating the thermal behaviour of major satellite components with a precision of a couple of degrees.

4.2 Parallel Processing

Parallel processing of several processor simulators has been proven as an important performance gain. Using the theory of "separability", developed at CNES, we are in a good starting point to engineer the parallelization. Currently, a methodology is being developed to detect model dependencies and allow for parallelization by configuration, without changing the models. This step can be taken when the normal non-parallel simulator is validated.

Another form of parallel work under investigation is the running of a simulator in parallel with the real system. The use would be twofold:

- In a first phase, to dynamically validate (and improve) the simulator versus the real system.
- In a second phase, to compare the real system against the simulator as to warn the operator when something is out of limits. Obviously, such system could have a far more refined warning capability than existing supervision systems.

A frequent context save of such systems would allow to jump backwards in time for deeper investigation of out of limit behaviour and perform what-if scenarios based on a saved context.

4.3 Processor Emulators

Processing emulators are the critical path in operational simulators, so significant efforts are devoted to them.

Current emulators decode each instruction to be executed, which limit their speed to around 70 MHz.

One trail concerns the dynamic translation or Just in Time compilation of flight software. It has been demonstrated that such emulators have the capability to reach 500 MHz emulation capability.

Another trail concerns the emulation of multi-core processors exploiting the multiple cores of the PC.

Another domain being investigated concerns the emulation of the space variant of ARINC 653 (also called TSP and IMA). In IMA, application layers are isolated in partitions that are time sliced by a hypervisor. Such partitions could probably simulated in parallel as by design, they have much fewer interdependencies.

Towards a formal language for systemic requirements

Yann Hourdel

LIX, École Polytechnique, 91128 Palaiseau Cedex, France,
yann.hourdel@polytechnique.edu

Abstract. This work is an attempt to contribute to the field of systems architecture. More precisely, it deals with complex¹ engineered systems analysis. Many informal methods for architecting such systems have been described over the past decade, but a lot of specific points still need to be clarified by a precise (mathematically formalized) definition. In particular, the languages used to manipulate system properties during systemic analysis are one big issue to be tackled. Our approach is the following: we take the framework described in [6] and reviewed in [4] as a starting point, and build a formal language to express functional (behaviour) requirements on models. The result is a formal language that allows architects to manipulate precise constraints on their models and, more importantly, translate them across subsequent systemic levels.

Keywords: Systems modeling, Systems architecture, Systems Engineering, Architecture framework

Introduction

In this work, systems are seen as black boxes. Their behaviour is only functional, so we will only express functional constraints on them. This kind of constraints is one perfect thing to be formalized, since it is very close to mathematical notions. Moreover, let us precise some important points:

- This work only deals with deterministic systems, for which we are able to describe the set of possible executions.
- In this work, time is considered discrete. That allows us to speak about the *previous* or *next* instant of an t .

¹ large, integrated, dense and heterogeneous

Notations

In the present work, the concatenation of two vectors A and B will be noted $A \otimes B$.

More generally, we will note $f \otimes g : A \otimes C \rightarrow B \otimes D$ the concatenation of $f : A \rightarrow B$ and $g : C \rightarrow D$.

1 Preliminary definitions

In this section, we recall the definitions introduced in [3] and reviewed in [4] to formalize the notion of *system*, a timed extension of Mealy machines to model heterogeneous integrated systems and their integration.

Definition 1 (Type). *The notion of **type** will be the classical “set of values” one.*

1.1 Time

Time is an underlying, yet very important, point of our formal approach. Indeed, real-life systems are naturally described according to various types of “times”. As a result, we need to deal uniformly with both continuous and discrete times. While this problem has been shown hard by [2], some solutions have been found in studies like [5] to introduce formal models for mixed discrete-continuous systems. We give here a set of definitions to handle such systems.

Informally, as very well expressed in [3], time is “a linear quantity composed of ordered moments, pairs of which define durations”.

Definition 2 (Time reference). *A time reference is an infinite set T together with an internal law $+^T : T \times T \rightarrow T$ and a pointed subset $(T^+, 0^T)$ satisfying the following conditions:*

- upon T^+ :
 - $\forall a, b \in T^+, a +^T b \in T^+$ closure (Δ_1)
 - $\forall a, b \in T^+, a +^T b = 0^T \implies a = 0^T \wedge b = 0^T$ initiality (Δ_2)
 - $\forall a \in T^+, 0^T +^T a = a$ neutral to left (Δ_3)
- upon T :
 - $\forall a, b, c \in T, a +^T (b +^T c) = (a +^T b) +^T c$ associativity (Δ_4)
 - $\forall a \in T, a +^T 0^T = a$ neutral to right (Δ_5)
 - $\forall a, b, c \in T, a +^T b = a +^T c \implies b = c$ cancelable to left (Δ_6)
 - $\forall a, b \in T, \exists c \in T^+, (a +^T c = b) \vee (b +^T c = a)$ linearity (Δ_7)

Definition 3 (Time scale). *A time scale is any subset \mathbb{T} of a time reference T such that:*

- \mathbb{T} has a minimum $m^{\mathbb{T}} \in \mathbb{T}$
- $\forall t \in \mathbb{T}, \mathbb{T}_{t+} = \{t' \in \mathbb{T} \mid t \prec t'\}$ has a minimum called $\text{succ}^{\mathbb{T}}(t)$

- $\forall t \in T$, when $m^{\mathbb{T}} \prec t$, the set $\mathbb{T}_{t-} = \{t' \in \mathbb{T} \mid t' \prec t\}$ has a maximum called $\text{pred}^{\mathbb{T}}(t)$
- the principle of induction² is true on \mathbb{T} .

The set of all time scales on T is noted $Ts(T)$.

1.2 Dataflows

Together with this unified definition of time, we need a definition of data that allows to handle the heterogeneity of data among real-life systems. We rely on the previous definitions to describe data carried by dataflows.

Definition 4 (ϵ -alphabet). A set D is an ϵ -**alphabet** if $\epsilon \in D$. For any set B , we can define an ϵ -alphabet by $\bar{B} = B \cup \{\epsilon\}$.

Definition 5 (System dataset). A **system dataset**, or **dataset**, is a pair $\mathcal{D} = (D, \mathcal{B})$ such that:

- D is an ϵ -alphabet
- \mathcal{B} , called **data behavior**, is a pair (r, w) with $r : D \rightarrow D$ and $w : D \times D \rightarrow D$ such that³:
 - $r(\epsilon) = \epsilon$ (R1)
 - $r(r(d)) = r(d)$ (R2)
 - $r(w(d, d')) = r(d')$ (R3)
 - $w(r(d'), d) = d$ (W1)
 - $w(w(d, d'), r(d')) = w(d, d')$ (W2)

Definition 6 (Dataflow). Let \mathbb{T} be a time scale. A **dataflow** over $(\mathcal{D}, \mathbb{T})$ is a mapping $X : \mathbb{T} \rightarrow D$.

Definition 7 (Sets of dataflows). The set of all dataflows over $(\mathcal{D}, \mathbb{T})$ is noted $\mathcal{D}^{\mathbb{T}}$. The set of all dataflows over \mathcal{D} with any possible time scale on time reference T is noted $\mathcal{D}^T = \bigcup_{\mathbb{T} \in Ts(T)} \mathcal{D}^{\mathbb{T}}$.

1.3 Systems and integration operators

Given the previous definitions, we are now able to give a mathematical definition of systems. Informally, our definition is very similar to timed Mealy machines with two important differences: the set of states may be infinite and the transfer function transforms dataflows. The key point is to see those systems as black boxes that just behave the way they are supposed to.

Definition 8 (System). A **system** is a 7-tuple $f = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ where:

² For $A \subset \mathbb{T}$, $(m^{\mathbb{T}} \in A \ \& \ \forall t \in A, \text{succ}^{\mathbb{T}}(t) \in A) \Rightarrow A = \mathbb{T}$.

³ These axioms give a relevant semantics and are necessary to define consistent projections of dataflows on time scales.

- \mathbb{T} is a time scale,
- X, Y are input and output datasets,
- Q is a nonempty ϵ -alphabet⁴ of states,
- q_0 is an element of Q , called initial state,
- $\mathcal{F} : X \times Q \times \mathbb{T} \rightarrow Y$ describes a functional behavior,
- $\delta : X \times Q \times \mathbb{T} \rightarrow Q$ describes a state behavior.

Figure 1 illustrates this definition.

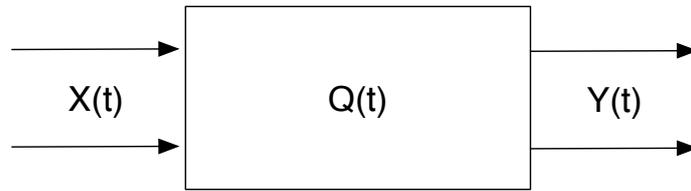
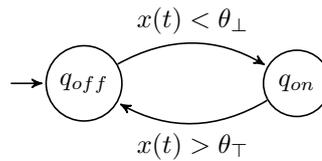


Fig. 1. Illustration of a system

Example 1. A very basic radiator with an internal thermostat placed in a room can be modeled as a system $S = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ with:

- $T \sim \mathbb{N}$
- $X = \text{room temperature} \sim R$
- $Y = \{\text{heat}, \text{nothing}\}$
- $Q = \{q_{on}, q_{off}\}$
- $q_0 = q_{off}$
- $\mathcal{F}(x(t), q(t)) = \begin{cases} \text{heat} & \text{if } q(t) = q_{on} \\ \emptyset & \text{otherwise} \end{cases}$
- δ is as follows:



It is important to understand here that at each time instant of the time scale, the state of the system changes instantly and before \mathcal{F} computes the resulting output. At $m^{\mathbb{T}_s}$, the beginning of the time scale, the state of the system is q_0 . But as soon as the first input data arrives, at $\text{succ}^{\mathbb{T}}(m^{\mathbb{T}_s})$, the state of f changes so that the functional behaviour ignores q_0 . Figure 2 illustrates this behaviour and we give the following formal definition for timed executions of systems.

⁴ Defining Q as an ϵ -alphabet (therefore containing ϵ) and not just as a set will make it possible to define a dataflow of states, which is convenient.

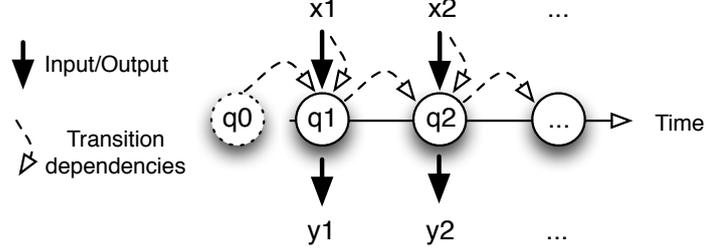


Fig. 2. Transitions of a system throughout its time scale

Definition 9 (Execution of a system). Let $f = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system. Let $In \in X^{\mathbb{T}}$ be an input dataflow for f and $\tilde{In} = In_{\mathbb{T}}$. The **execution** of f on the input dataflow In is the 3-tuple (In, S, Out) where:

- $S \in Q^{\mathbb{T}}$ is recursively defined by:
 - $S(m^{\mathbb{T}}) = \delta(\tilde{In}(m^{\mathbb{T}}), q_0, m^{\mathbb{T}})$
 - $\forall t \in \mathbb{T}, S(t^+) = \delta(\tilde{In}(t^+), S(t), t^+)$
where $t^+ = succ^{\mathbb{T}}(t)$
- $Out \in Y^{\mathbb{T}}$ is defined by:
 - $Out(m^{\mathbb{T}}) = \mathcal{F}(\tilde{In}(m^{\mathbb{T}}), q_0, m^{\mathbb{T}})$
 - $\forall t \in \mathbb{T}, Out(t^+) = \mathcal{F}(\tilde{In}(t^+), S(t), t^+)$
where $t^+ = succ^{\mathbb{T}}(t)$

In , S and Out are respectively input, state and output dataflows.
We note $exec(f)$ the set of possible executions of f .

Definition 10 (Product of systems on a time scale). Let $(f^i)_i = (\mathbb{T}, X_i, Y_i, Q_i, q_{0_i}, \mathcal{F}_i, \delta_i)_i$ be n systems of time scale \mathbb{T} . The **product** $f^1 \otimes \dots \otimes f^n$ is the system $(\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ where:

- $X = X_1 \otimes \dots \otimes X_n$ and $Y = Y_1 \otimes \dots \otimes Y_n$
- $Q = Q_1 \times \dots \times Q_n$ and $q_0 = (q_{0_1}, \dots, q_{0_n}) = q_{0_{1\dots n}}$
- $\mathcal{F}(x_{1\dots n}, q_{1\dots n}, t) = (\mathcal{F}_1(x_1, q_1, t), \dots, \mathcal{F}_n(x_n, q_n, t))$
- $\delta(x_{1\dots n}, q_{1\dots n}, t) = (\delta_1(x_1, q_1, t), \dots, \delta_n(x_n, q_n, t))$

Remark 1. This definition can be extended to systems that do not share a time scale, thanks to a technical operator introduced in [3]. This operator builds a timed-extension of a system, which is a system that has an equivalent input-output behaviour as the original system, but on a wider time scale. Figure 3 illustrates this idea.

Definition 11 (Feedback of a system). Let $f = (\mathbb{T}, (D \times In, \mathcal{I}), (D \times Out, \mathcal{O}), Q, q_0, \mathcal{F}, \delta)$ be a system such that there is no instantaneous influence of dataset D from

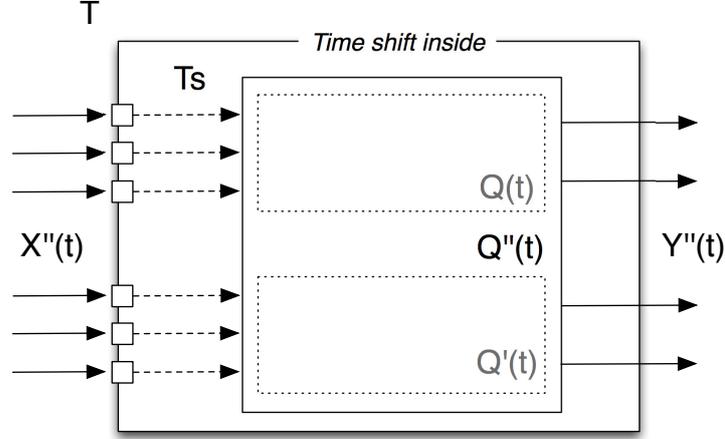


Fig. 3. Product of systems

the input to the output⁵, i.e. $\forall t \in \mathbb{T}, \forall x \in In, \forall d \in D, \mathcal{F}((d, x), q, t)_D = \mathcal{F}((\epsilon, x), q, t)_D$. The **feedback of D in f** is the system $f_{FB(D)} = (\mathbb{T}, (In, \mathcal{I}'), (Out, \mathcal{O}'), Q, q_0, \mathcal{F}', \delta')$ where:

- \mathcal{I}' is the restriction of \mathcal{I} to In , and \mathcal{O}' is the restriction of \mathcal{O} to Out
- $\mathcal{F}'(x \in In, q \in Q, t) = \mathcal{F}((d_{x,q,t}, x), q, t)_{Out}$
- $\delta'(x \in In, q \in Q, t) = \delta((d_{x,q,t}, x), q, t)$

where $d_{x,q,t}$ stands for $\mathcal{F}((\epsilon, x), q, t)_D$.

Figure 4

Definition 12 (Abstraction of a transfer function). Let $F : X^{\mathbb{T}} \rightarrow Y^{\mathbb{T}}$ be a transfer function. Let $A_x : X^{\mathbb{T}} \rightarrow Y_a^{\mathbb{T}^a}$ be an abstraction for input dataflows and $A_y : Y^{\mathbb{T}} \rightarrow Y_a^{\mathbb{T}^a}$ an abstraction for output dataflows. The **abstraction of F** for input and output abstractions (A_x, A_y) with events \mathcal{E} is the new transfer function

$$F_a : (X_a \otimes \mathcal{E})^{\mathbb{T}} \rightarrow Y_a^{\mathbb{T}^a}$$

defined by:

$$\forall x \in X^{\mathbb{T}}, \exists e \in \mathcal{E}^{\mathbb{T}^a}, F_a(A_x(x_{\mathbb{T}}) \otimes e) = A_y(F(x))$$

Figure 5 illustrates this definition.

⁵ As explained informally in [3], this condition makes it possible to define a unique feedback, i.e. without having to solve a fixed point equation that could lead to zero or multiple solutions

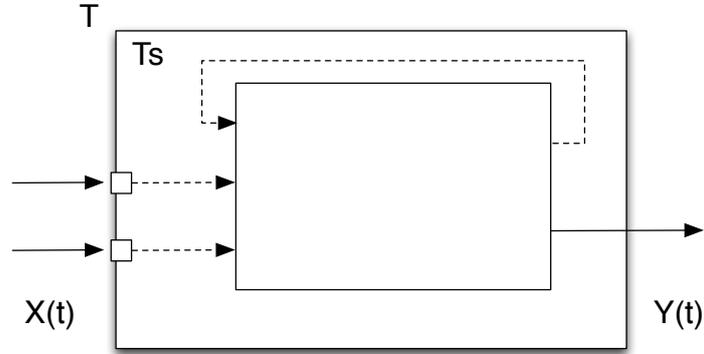


Fig. 4. Feedback of a system

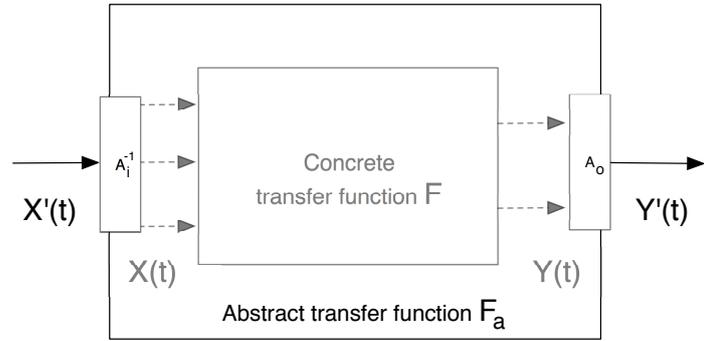


Fig. 5. Abstraction of a transfer function

Definition 13 (Abstraction of a system). Let $f = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system. $f' = (\mathbb{T}_a, X_a \otimes \mathcal{E}, Y_a, Q_a, q_{a0}, \mathcal{F}_a, \delta_a)$ is an abstraction of f for input and output abstractions (A_x, A_y) if, and only if: $\exists A_q : Q^{\mathbb{T}} \rightarrow Q_a^{\mathbb{T}_a}$, for all execution (x, q, y) of f , $\exists E \in \mathcal{E}^{\mathbb{T}_a}$, $(A_x(x_{\mathbb{T}}) \otimes E, A_q(q), A_y(y))$ is an execution of f' . Conversely, f' is a concretization of the system f .

A system captures the behavior of a system that can be observed (functional and states behavior, called together *systemic behavior*). From this definition, we can start expressing behaviour properties.

2 Systemic behaviour properties: a formal semantics

The goal of this section is to be able to describe the behaviour of such a system, in order to express properties and constraints on it. To do so, we will define a

semantics of systems and provide a formal definition of “properties”. The idea here is that systems are described by executions, so we must use timed properties. We will first describe our property syntax language, then our property semantics.

Definition 14 (System property formulas). Let $f = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system.

The set $P(f)$ of property formulas over f , similar to LTL⁶, is inductively defined as follows.

Here are the atomic formulas, where $(x, q, y) \in X \times Q \times Y$:

- $\text{input}(x)$, which means that the current input of f has to be x
- $\text{istate}(q)$, which means that the current state of f has to be q
- $\text{output}(y)$, which means that the current output of f has to be y

And here are the operators, where $(\phi, \phi_1, \phi_2) \in P(f)$ ³:

- $\neg\phi$
- $\phi_1 \wedge \phi_2$
- $\bigcirc\phi$, which means that ϕ has to hold at the next state of the execution
- $\phi_1 \mathcal{U}\phi_2$, which means that ϕ_2 eventually has to hold and until it does, ϕ_1 has to hold (ϕ_1 can hold further)

Definition 15 (Other system property formulas). Let $f = (\mathbb{T}, X, Y, Q, q_0, \mathcal{F}, \delta)$ be a system.

Let $(\phi, \phi_1, \phi_2) \in P(f)$ ³.

The previously defined language $P(f)$ can be extended with the following operators:

- \top
- \perp
- $\phi_1 \vee \phi_2$
- $\phi_1 \Rightarrow \phi_2$ ⁷
- $\diamond\phi$, which means that ϕ has to hold now or in a future state of the execution
- $\square\phi$, which means that ϕ has to hold for the entire subsequent execution
- $\phi_1 \mathcal{R}\phi_2$, which means that ϕ_1 has to hold until and including a state when ϕ_2 holds, which is not forced to happen if ϕ_1 holds forever

$P(f)$ is the *syntax* of our properties language. It gives us a way to express properties on executions of f . We are now able to define our semantics. The following rules describe the satisfaction of $P(f)$ formulas using only the first set of operators, but such rules could be easily extended to the extended set of operators.

Definition 16 (system property satisfaction). Let f be a system.

The **satisfaction** of a $P(f)$ formula ϕ by an execution e of f , written $e \models \phi$, is defined according to the following rules:

⁶ see [1]

⁷ We can also accept $\phi_1 \otimes \phi_2$ or any other “usual” operators

$$\begin{array}{c}
\frac{}{((x_0, -, -), \dots) \models \text{input}(x_0)} [AI] \quad \frac{}{((-, q_0, -), \dots) \models \text{istate}(q_0)} [AS] \\
\frac{}{((-, -, y_0), \dots) \models \text{output}(y_0)} [AO] \quad \frac{[e \models \phi] \Rightarrow \perp}{e \models \neg\phi} [AN] \\
\frac{e \models \phi_1 \quad e \models \phi_2}{e \models \phi_1 \wedge \phi_2} [AW] \quad \frac{(e_1, e_2, \dots) \models \phi}{(-, e_1, e_2, \dots) \models \bigcirc\phi} [AC] \\
\frac{\exists i \leq 0, [((e_i, e_{i+1}, \dots) \models \phi_2) \wedge (\forall k \in \{0, \dots, i\}, (e_k, e_{k+1}, \dots) \models \phi_1)]}{(e_0, e_1, \dots) \models \phi_1 \mathcal{U} \phi_2} [AU]
\end{array}$$

Definition 17 (system behaviour constraint). Let f be a system. Let ϕ be a $P(f)$ formula.

We say that f satisfies ϕ , which is noted $f \models \phi$, iff

$$\forall e \in \text{exec}(f), e \models \phi$$

In this case, ϕ is said to be a **behaviour constraint** on f .

Example 2. Unsing our previous example 1, with $\theta_{\perp} = 15$ and $\theta_{\top} = 20$, here are some behaviour constraints one would want to express on the system:

- $f \models \neg\Diamond(\text{istate}(q_{off}) \wedge \text{output}(\text{heat}))$
- $f \models \neg\Diamond(\text{input}(10) \wedge \bigcirc\text{istate}(q_{off}))$

or, even more generally:

- $\forall \theta > \theta_{\top}, f \models \neg\Diamond(\text{input}(\theta) \wedge \bigcirc\text{istate}(q_{on}))$

3 Computation rules over behaviour constraints

We give here a minimalist set of computation rules to establish proofs about system behaviours. A more advanced set of rules might be needed to ease such proofs.

Proposition 1 (Product of two systems). Let $f_1 = (\mathbb{T}, X_1, Y_1, Q_1, -, -, -)$ and $f_2 = (\mathbb{T}, X_2, Y_2, Q_2, -, -, -)$ be two systems.

Let $(x_1, x_2) \in X_1 \times X_2$, $(y_1, y_2) \in Y_1 \times Y_2$ and $(q_1, q_2) \in Q_1 \times Q_2$.

$$\frac{f_1 \models \text{input}(x_1) \quad f_2 \models \text{input}(x_2)}{f_1 \otimes f_2 \models \text{input}(x_1 \otimes x_2)} [PI]$$

$$\frac{f_1 \models \text{output}(y_1) \quad f_2 \models \text{output}(y_2)}{f_1 \otimes f_2 \models \text{output}(x_1 \otimes x_2)} [PO]$$

$$\frac{f_1 \models \text{istate}(q_1) \quad f_2 \models \text{istate}(q_2)}{f_1 \otimes f_2 \models \text{istate}((q_1, q_2))} [PS]$$

Definition 18 (Composition of two systems). Let $f_1 = (\mathbb{T}, X_1, Y_1, Q_1, \rightarrow, \rightarrow, -)$ and $f_2 = (\mathbb{T}, X_2, Y_2, Q_2, \rightarrow, \rightarrow, -)$ be two systems such that $Y_1 = X_2$. We note $f_2 \circ f_1$ the composition of f_1 and f_2 , obtained by “plugging” the output of f_1 to the input of f_2 .

Proposition 2 (Composition of two systems). Let $f_1 = (\mathbb{T}_1, X_1, Y_1, Q_1, \rightarrow, \rightarrow, -)$ and $f_2 = (\mathbb{T}_2, X_2, Y_2, Q_2, \rightarrow, \rightarrow, -)$ be two systems such that $\mathbb{T}_1 = \mathbb{T}_2$ and $Y_1 = X_2$. Let $(x, y, q_1, q_2) \in X_1 \times Y_2 \times Q_1 \times Q_2$.

$$\frac{f_1 \models \text{input}(x)}{f_2 \circ f_1 \models \text{input}(x)} [WI]$$

$$\frac{f_2 \models \text{output}(y)}{f_2 \circ f_1 \models \text{output}(y)} [WO]$$

$$\frac{f_1 \models \text{istate}(q_1) \quad f_2 \models \text{istate}(q_2)}{f_2 \circ f_1 \models \text{istate}((q_1, q_2))} [WS]$$

Conclusion

In this work we built a semantics of systems and provided a formal definition of “properties” as timed behaviour constraints. The next step is to build a more advanced refinement computation language that could let modelers obtain a system from a set of such constraints.

References

1. LTL. http://en.wikipedia.org/wiki/Linear_temporal_logic. (last accessed on 31/10/2012).
2. O. Bournez and M.-L. Campagnolo. A survey on continuous time computations. *CoRR*, abs/0907.3117, 2009.
3. B. Golden, M. Aiguier, and D. Krob. Modeling of complex systems ii: A minimalist and unified semantics for heterogeneous integrated systems. *Applied Mathematics and Computation*, 218(16):8039–8055, 2012.
4. B. Golden and Y. Hourdel. A minimalist formal framework for systems design. 2013.
5. T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–, Washington, DC, USA, 1996. IEEE Computer Society.
6. D. Krob. *Éléments de systématique - architecture des systèmes*. 2012.

