The rationale paradigm in system development lifecycle

Abd-El-Kader Sahraoui

The Rationale Paradigm in System Development Lifecycle

Abd-El-Kader Sahraoui
CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ;
F-31077 Toulouse Cedex 4, France
sahraoui@laas.fr

Abstract: This paper describes a generic method to assist architectural design of heterogeneous systems. It extends current systems engineering lifecycle by improving design rationalization and incorporation of lessons learned. The author usesModel Driven approach to define architectural models of the system development lifecycle.The approach is based on the translation of requirements in quantitative and measurable technical indicators that could be integrated to the design models.Research presented in this paper rely on the EIA632 processes andIEEE1471 definitions.Unified Modeling Language (UML) and its subset, Systems Modeling Language (SysML) are used in method implementation.

Keywords: systems engineering, design rational system development, lifecycle

1. Introduction and problem statement

In engineering of aeronautic equipment and systems, the following thinking and approach is viewed as independent of specific domains. Specifically, the current perspective relates to architectural design phase in which design decisions have a strong implication on project and program cost, schedule and performance systems engineering. This phase consists on defining the system architecture, as a set of interacting building blocks, and specifying them.

According to IEEE 1471 (software architecture standard), the architecture is a fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. The author intends to follow such an assertion as a guideline for the construction of the logical and physical model of avionic applications.

In current approach, there are two main phases. First, building-up of the system requirements specifications and their functional modeling. In this phase, the author provides a set of requirements able to satisfy the customers and the others stakeholders requirements (i.e., mainly airworthiness and user's requirements for our application). The main outcome is a system requirements and interfaces specifications that should be given to the project actors. The second phase tackles system architecture. Specifically, it addresses how to derive the physical architecture alternatives from the first phase. This allows for allocation of system requirements (specifying building blocks) into each alternative and selection the corresponding requirements. Deriving physical architecture makes use of the system functionalities, performances and interface requirements as inputs. Outcomes are then reported on system-level definition document that should be given to project actors too.

Tackling the architectural design is a complex task, which involves delicate tradeoffs, assessments, arbitrations and agreement, with critical cost and time. This activity is characterized by:

- The needs that have to be sufficiently specified in order to evaluate the architecture acceptability (customer agreement). Nevertheless, usually they are not enough to accurately evaluate the architecture desirability and optimality.
- The designers that have difficulties to link customer satisfaction criteria and system architecture.
- The design is plagued by many assumptions and uncertainties which are tacit in nature.

In this paper, the author a attempts to reformulate the problem in a more structured manner by relating current design issue though the following research questions:

- How to you define implementation independent metrics in order to compare very different physical models of designs?

- How to you manage qualitative and quantitative criteria in an integrated and structured manner?
- How can you compensate for lack of technical data in the preliminary design phase. Additionally, how do you manage increasing bulk data that constantly changes based in decision-makers?

It is from this perspective that the author contends that system development lifecycle requires integration of system metric definition, development of design alternatives (i.e., options), and evaluation metrics into single a lifecycle framework. This enables effective optimization in design architectures specifically on the high levels of the design tree. This approach enables technical skill sharing and transferability from project to project.

Up to this point, leveraging lessons learned has not received its share attention in systems engineering. To advance this issue, this paper is divided into three major sections. Following this introduction is section two that provides a state of the art in the *design rationale* domain necessary in architectural analysis and decision process in systems engineering. Section three presents the suggested approach on decision-making for rationale design. Detailed description of design rationale is provided including measures of effectiveness (MoE), design constructs, architectural analysis and decision process, and SysML implementation. The paper concludes with a brief discussion about future research and expected results.

2. State of the art

It is necessary to start with a recall of *design rationale*. According various encyclopaedias, *Design Rationale* is the explicit listing of decisions made during a design process and the reasoning behind such decisions. This listing often includes not only the reasons behind design decisions, but also justifications for decisions made. This entails discussion of considered alternatives, the evaluated trade-offs, and argumentation foe that lead to selected choices. It can also be stipulated that the primary goal of design rationale is to support system designers. It enables designers to recommend and communicate argumentation and reasoning behind design process. Shum and Hammond [1] espouses that design rationale expresses elements of the reasoning that has been invested behind the design of an artefact. *Design rationale* is the reasoning and argument that leads to the final decision of how the design intent is achieved. Additionally, design rationale can be described in terms statements of reasoning underlying the design process that explain, derive, and justify design decisions[2], [3]. It also entails information that explains why an artefact is structured the way that it is and has the behaviour that it has[4]. Furthermore, design rationales include not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision [5].

Most of the survey carried out was focussed mainly on design rationale in software engineering. As we will see the application can be deployed for systems engineering. *Design rationale* research ranges from general notations, such as IBIS [6] to tools designed for specific types of design in specific domains, such as ADD [7], which is specifically for parametric design in the heating, ventilation, and air conditioning (HVAC) domain. Review of literature also reveals that some research has been done in software design (i.e., the creation of generic elements models in software design rationale). This research has also been extended by Lee in his Decision Representation Language (DRL), the language used in SYBYL[8].

Design Recommendation and Intent Model (DRIM) was used in a system to improve design patterns with design rationale [9]. This system is used to select design patterns based on designer intent and other constraints. Software process model can be used to obtain design decisions and causal dependencies between them. These causal dependencies are then considered to be the design rationale. The system looks like tasks (i.e., goals to be reached during the process), products (i.e., the products produced, such as specifications), methods (i.e., the way that the task is accomplished) and agents (i.e., the human or computer who works on the task). When the process model is designed, the information flow between the tasks is captured and used to deduce the dependencies between them. In this instance, the rationale includes of the justification for choosing a particular method. This is inferred from the dependencies in the model. For example, choosing a particular method may mandate specific values for system variables. The rationale for those variables assignment is the validity of the method choice.

There are several types of justifications for a decision: justification based on validity or invalidity of previous decisions and existing parameter values. WinWin [10] is an approach aimed at coordinating decision-making activities made by various *stakeholders* in the software development process. The work described in this document, is not an argumentation system. It does not keep track of alternatives tried and rejected and the reasons behind them. Instead, it uses the dependencies to trace through the dependency chain to find the justification for each decision. If a decision is changed, the system detects changes and invalidates the reasons for other dependent decisions.

One way to encourage capture and use of rationale is to make it an integral part of the development process. This is essential in software related projects since projects have a long lead-time. Process improvement initiatives, such as the Software Engineering Institute's Capability Maturity Model are intended to meet the primary software development goals of increased quality and reduced development costs. Implementing rationale as an integral part of the development process could aid in traceability.

Software development processes are commonly defined as software life-cycle models. There are a number of different models that have been used over the past thirty years. These include linear sequential models such as the step-wise model and its more commonly known descendant the waterfall model. In these models, the development process proceeds through a number of phases that may or may not have feedback loops between them. Design rationales can be traced back to any item. Such a rational can be attached to a transition between a requirement and a design but also between an implementation and test procedure [11].

In order to encourage design rational used in system development, the process chosen should be similar to those that had been given wide acceptance in the software community. One of the processes that is widely accepted in the Unified process. Unified Process is an evolutionary process that grew out of a number of processes and modelling techniques, and is based on the object-oriented programming paradigm and is centred on use-cases. Use-cases are descriptions of the interactions between the system and its environment. An interaction is when the user invokes the system, usually to perform a specific function, and terminates when the user has completed their action. The process is represented using the Unified Modelling Language (UML). One major advantage of this process is that it is has tool supports.

Now let us look how trade studies are described in several system engineering norms. In EIA 632 (systems engineering standard), Trade-off process is a part of System Analysis Process. It is a generic description which is applicable both on logical and physical solution. It advises cost and schedule performance and risk analysis, selection of criteria, associated with weighting factor, and the use of model and representation such as simulation to evaluate each option. Outcomes are recommendations to decision makers to be applied as part of logical and physical solution definition process. The Trade-off process includes specific activities for quantitative analysis effectiveness and risk analysis. It is mostly identify to evaluation each option and provide recommendation to decision makers.

On the other hand, EIA 632 defines measures of effectiveness (MoE) as means by which user, customer, or acquirer will measure satisfaction or acceptance, or overall expectations against which satisfaction will be determined. Sproles[12] had also proposed a definition of MoE as "Measures of Effectiveness are standards against which the capability of a solution to meet the needs of a problem may be judged. The standards are specific properties that any potential solution must exhibit to some extent. MoEs are independent of any solution and specify neither performance nor criteria."However, Sproles does not suggest how MoE could be used in process design.

3. Approach based on effectiveness metrics for design trade-off

Author has proposes the use of architectural decision and the architectural decision process as key element in design methodology. In this section, the author describes how the intended methodology could be organized (see Figure 1). On the top left, there are the user requirements shown as the project-specific data that it is representative of the stakeholder-identified needs. On the top right, we can see the project-independent, generic architectural constructs that can be both standard components and generic models. Both type of data and models are merged together at design decision phase. In our process, this phase is based on the proposal: 1) evaluation of multiple alternatives of architectural solutions, 2) leading to the selection of the best one. The design decisions are centric of this evaluation and selection process.

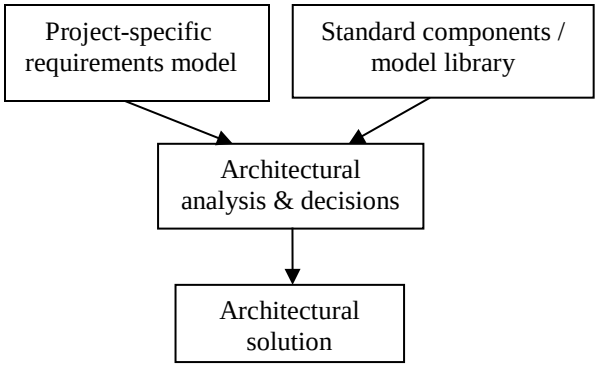

Figure1. A generic way to design system architecture

An architectural decision is the task which produces an architectural construct using, one hand initial project-specific inputs, and on the other hand, tacit and explicit knowledge of the problem domain. This domain knowledge is wide and includes designer technical skills, domain good practices, enterprise, and decision-maker lesson-learned.In the following section, project specific model, project-independent architectural constructs library, decisionprocesses are described.

3.1. Building a set of Measures of Effectiveness for alternatives evaluation

In order to be able to evaluate desirability of each alternative, it is necessary to prepare the evaluation phase early in the design process. During the technical requirement definition process, the requirements are elicited from stakeholders (i.e., customers, acquirers, etc.).

At this stage in design, the author proposes providing system designers with means of establishing a set of implementation-independent MoE. This means that some specific requirements, called MoE, will be used to measure, in a qualitative or quantitative manner, the overall satisfaction of a given stakeholder. The EIA632 and Sproles [12] definitions insist on the fact that MoE should be independent of the implementation, thus allowing system designer (or tradeoffs leader) to objectively compare design alternatives. To be implemented independently, MoE should be elicited and formalized during a logical preliminary design.

The good practices of requirement engineering should also be applied to MoE: Validation process should assess that they are unambiguous, complete, consistent, improved and traceable, as any other system requirement. All project actors should have the same understanding of MoE. It is important to note that validation, verification, and quantification are very difficult to assess in the early stages of the system life cycle. The MoE set will typically include:

- A subset of implementation independent performance/behavioral requirements that is critical for the stakeholder's satisfaction.
- Some systems properties like maintainability, interoperability, degree of design complexity or feasibility.
- Some more application-specific properties like look and feel, integration with environment,
- Cost, schedule performance and risk acceptability.
- Derived requirements from the enabling systems and other products.

It is expected that the MoE will share concerns of customers or others stakeholders especially is the acceptability criteria. The identified MoEs should be prioritized and related to each other (i.e., identifying dependency and resolving conflicts).

3.2. Architectural design constructs

In a first approximation, the design at architectural level mainly set up object interactions and efficient interfaces between a set of implementation concepts and building blocks. Once done, it provides a solution to a generic design problem. Building blocks are bounded to realize the overall functionality of the system. The physical solution includes physical sub-assemblies as a set of interacting components. These sub-assemblies use combined individual properties of its physical parts to realize the given function(s) that meets specified requirements and constraints. These can be viewed as some design patterns chosen by the designers because of the following reasons:

- The patterns so connected provide services that satisfy the functional requirements.

- Their intrinsic properties comply with the system constraints.
- A provided service (e.g., energy, information transmission means, storage means, etc.)
- Required in a physical interface between and among systems (e.g., mechanical, electrical, etc.)
- Involved in a physical domain such as electric, mechanic, hydraulic, etc.

Furthermore, some generic effectiveness characteristics allow to compare and to relate them to each other (i.e., taking into consideration system *utilities*) For example, maturity, reliability, flexibility, inter-changeability, etc. These sub-assemblies use combined individual properties of its physical parts to realize a given function within specified requirements and constraints. Under some conditions, they are *component centric*. This is particularly true when this central part is innovative. Thus, the designers work on building sub-assemblies that best integrate the new technologies.

3.3. Architectural analysis and architectural decision process

The relevant point of current approach is the architectural decision. In our top-down process, architectural decisions are taken to allow for logical model to be selected followed by physical model (see 1 in Figure 2). This is followed by breaking down the high level architectural elements (or building blocks) into interconnected elements (see 2 in Figure 2) and specifying the building blocks (see 3 in Figure 2).

As previously described, the design decisions make the junction between requirements domain and technical solution domain. The author proposes that designers define few competing architectures, with each one having a likelihood of being selected. Principle of defining more than one solution at this design level allows getting away from conservatism and encourages innovation and creativity without being tedious for designers.

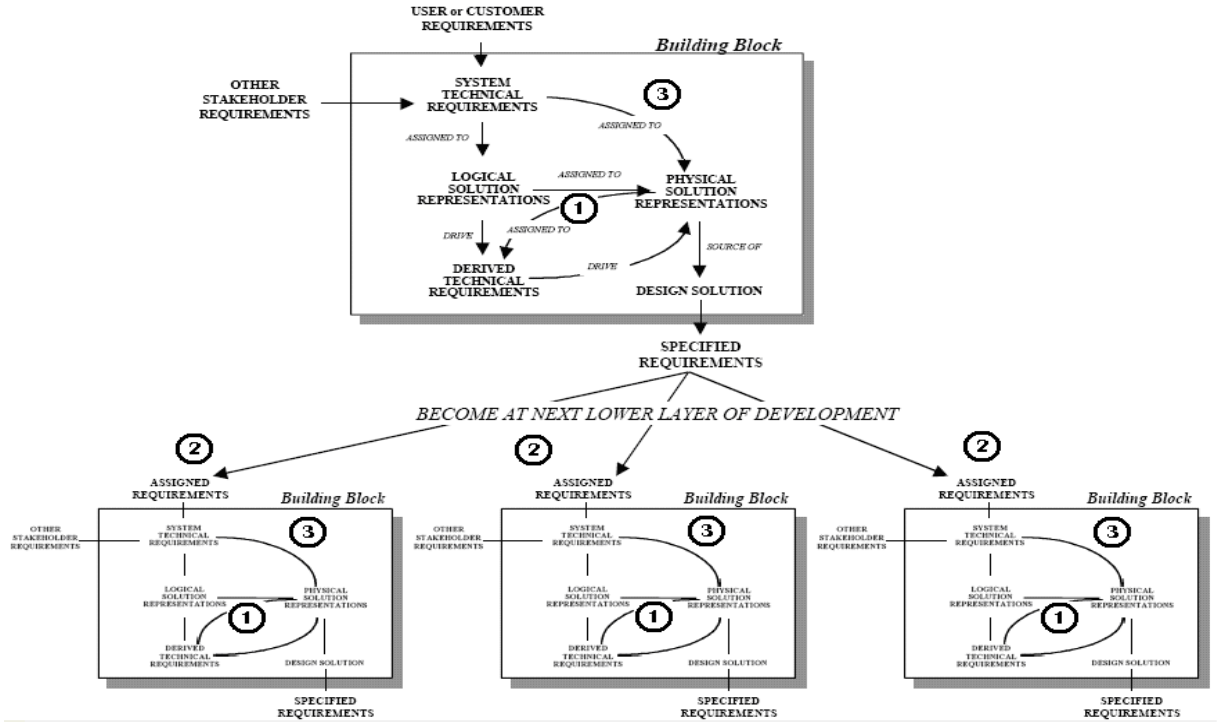

Figure 2. Architectural decisions using EIA632 System design process

This process implies that designers have efficient means to assess high level of design solution (i.e., validating them, including feasibility assessment, and finally evaluate them against customer's expectations). Validation activity focuses on proving that design is able to meet stockholder requirements. The means we propose to assess design architecture is based on a complete and structured rationale process. During decision time, the designer demonstrate how their decision impacts (positively or negatively), the set of MoE, by translating each impacted MoE into quantifiable technical measures. There is a built in assumption regarding set of MoEs (i.e., architectural alternative

is better than the others).. As an operational point of view, the author stipulates that the decision process should be constructed by four successive and iterative tasks within a process as shown in Figure 3.

1) ***Building the architecture alternatives***. In the architectural assessment and architectures proposal and validation activities of figure 3, each alternative is submitted to a validation process ensuring feasibility and completeness.
2) ***Justify the decisions***. This is done by translating sets of MoE into sets of technical measures of performance (MoP) allocated to corresponding design concepts. Assumptions made within this translation should be identified and recorded in the rationale description. The MoP should be quantified according some possible criteria. The means used to realize the effective criteria measures, should be defined. Note that possible means are simulation and virtual prototyping. Engineering judgment, similarity and experience, mathematical / physical analysis could also be used.
3) ***Evaluate performance criteria*** of the alternatives (see effectiveness evaluations & rationale on Figure 3). In this task, the designer carries out the criteria evaluation using the measure means previously identified. Each result should be structured and formalized as an attribute of the physical architecture.
4) ***Effectiveness synthesis***. This last step allows for identification of sensitivities of architectural elements against each MoE. It means identifying architectural elements that have positive or negative impact on a specificMoE. Doing this assists the designer to locate critical elements and weakness of its design, allowing him to refine it, and allow the comparison between the alternatives. It also permits to define an objective function or evaluation matrix, providing the designers with an overall solution desirability indicator.

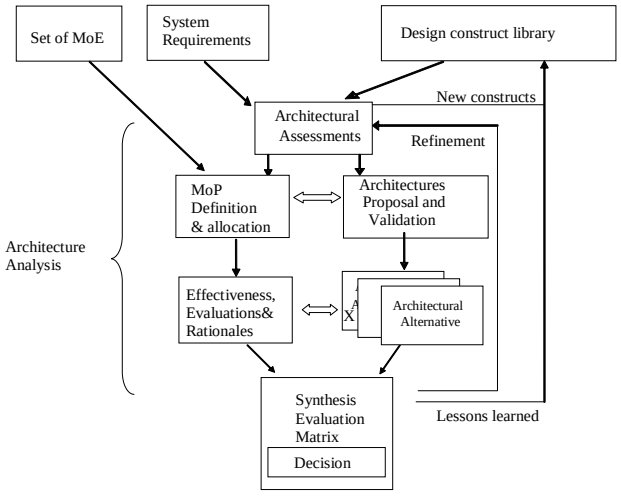

Figure 3: Architectural decision process

3.4 Formalizing the problem through optimisation technique

An efficient multi-objective optimization design method can be developed which requires much fewer design iterations than the traditional design methods. In this new method, the number of prime variables that define the optimization can be related to the formulation of rationale. Such rationale can be defined for each project designer in matter of cost, dependability, cost of manufacturing and other as customer satisfaction index.
The weight attributed in the objective function should comply with established standard. In the second described above (justify the decisions) can be handled by simulation or though Swarm type simulation. Performance criteria can either be at the objective function or formulated in constraints equations

3.5. A SysML Implementation

For implementation, the author has selected the SysML notation and mechanisms as a modelling language as shown in Table 1.Our set of MoE has been included as stereotyped SysML requirements, in conjunction with system requirements. All of these are related to operational use cases and sequence diagrams.

Design rationales are currently modelled by stereotyped comments. These comments are bounded to *satisfy* the relations between System, MoE requirements, and design constructs. A main implementation task should be development of holistic means to integrate rationale models with stereotyped and tagged elements. The use of Internal Block Diagram (IBD) to describe our architectural design constructs and architectural alternatives is selected. For behavioural models, the author associates the state-charts to blocks. Some design constructs, especially in the mechanical domain, have already been modelled and attempts to model theirs properties in consultation with the experts of the domain are on-going efforts. The author aims to use several tools for architectural evaluation means. The author postulates effectiverationale will be used with a bond graph translation of low level IBD architectural constructs in near future.

Table 1. Mapping rules between the method and SysML notation

Method Element	SysML notation used
Design Alternative	An Internal Block Diagram (IBD) describes each alternative. A stereotype states that IBD is an alternative
Effectiveness metrics MoE	Because a MoE formulation can be stated in quantitative manner, it is implemented by stereotyping a requirement block. A constraint is that MoE can only be derived.
Performance metrics MoP	MoPs are implemented both by stereotyped requirements blocks and by values specifications. Values specifications are an internal property of the owning block.
Deriving MoP from MoE.	MoP requirements blocks are related to MoE by derivation links. Derivation links from MoE is realised by Derive Reqt association type. Assumptions made during derivation of MoP is implemented by rationale comment and connected to association link.
MoP Measuring criteria & mean	When possible, MoP values specifications should be described by constraints properties. MoP value specification, Measuring means and related MoE can be presented in a parametric diagram.

4. Conclusion and expected results

In this paper, the author espouses that coupling of effectiveness metrics and design rationalization is promising for the purpose of high level design evaluation and lessons learn capitalization. The author proposes the integration of these two aspects of design evaluation into a unique, methodological framework, based on SysML notation. Future research will include proposition of SysML extension to materialize architectural decision and design rationale. Additionally, future research will include process of merging the proposed approach into existing SysML based on computer-aided design (CAD) tools.

Currently, an approach has been proposed for these issues. hence, future research will encompass the application and deepen current dialogue by integrating *design rationale* at each level of abstraction and propose a formal notation that can be integrated in SysML notation. In this respect in top down approach where each level is refined, the constraints can be expressed at each level and hence refined. The rational related to the objective function can hence refined from macro to micro approach. The whole simulation is hence executed at the low levels hence at components levels. Future work include too the dependability issue which is often neglected and hence the emergent properties. Such emergent properties reflect the design complexity of the system.

5. References

[1] S. Shum, N. Hammond, "Argumentation-Based Design Rationale: From Conceptual Roots to Current Use", *Tech. Report* EPC-1993-106, Rank Xerox Research Centre, Cambridge, 1993.

[2] S. Sim, A. Duffy, "A New Perspective to Design Intent and design Rationale", in *Artificial Intelligence Design Workshop* Notes for Representing and Using Design Rationale, 15-18 August 1994, pp. 4-12.

[3] G. Fischer, A. Lemke, R. McCall, A. Morch, "Making Argumentation Serve Design", *in Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carroll, eds., Lawrence Erlbaum Associates, pp. 267-294, 1995.

[4] J. Conklin and K. Burgess-Yakemovic, "A Process-Oriented Approach to Design rationale", in *Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carroll, (eds), Lawrence Erlbaum Associates, Mahwah, NJ, pp. 293-428, 1995.

[5] J. Lee, "Design Rationale Systems: Understanding the Issues", *IEEE Expert*, Vol. 12, No. 3, pp. 78-85, 1997.

[6] J. Conklin and K. Burgess-Yakemovic, "A Process-Oriented Approach to Design rationale", in *Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carroll, (eds), Lawrence Erlbaum Associates, Mahwah, NJ, pp. 293-428, 1995.

[7] A. Garcia, H. Howard, M. Stefik, "Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design", *Tech. Report 82*, Stanford Univ. Center for Integrated Facility Engineering, Stanford, Calif. 1993.

[8] J. Lee, "SIBYL: A qualitative design management system". *Artificial Intelligence* at MIT: Expanding Frontiers, In P.H. Winston and S. Shellard, eds.,Cambridge MA: MIT Press, pp. 104-133, 1990.

[9]Peña-Mora, "Component-based software development for integrated construction management software applications". *Artificial Intelligence for Engineering Design, Analysis and Manufacturingarchive* Volume 15, Issue 2 (April 2001)

[10] B. W. Boehm, P. Bose, E. Horowitz, and M-J. Lee, "Software Requirements As Negotiated Win Conditions", *Proceedings of International Conference on Requirements Engineering*, IEEE, April 1994.

[11]AEK Sahraoui, "Processes for engineering a system. An introduction to processes, methods and tools",*LAAS report No06347, 2nd IEEE International Conference on Information & Communication Technologies: from Theory to Applications* (ICTTA'06),24-28 April 2006.

[12] N. Sproles BA (Hons), ARMIT (Land Survey), "Coming to Grips with Measures of Effectiveness". *Systems Engineering* Volume 3, Issue 1, pp. 50 – 58, 2001.