INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010
Dubrovnik - Croatia, May 17 - 20, 2010.

FROM REQUIREMENTS TO DESIGN SPECIFICATIONS- A FORMAL APPROACH

W. Brace and K. Thramboulidis

Keywords: requirements, requirements checklist, design specification, requirements formalization, model-centric requirements engineering

1. Introduction

The traditional development process for mechatronic systems is criticized as inappropriate for the development of systems characterized by complexity, dynamics and uncertainty as is the case with today's products. Software is playing an always increasing role in the development of these systems and it has become the evolving driver for innovations. It does not only implement a significant part of the functionality of today's mechatronic systems, but it is also used to realize many of their competitive advantages [Thramboulidis 2009]. However, the current process is traditionally divided into software, electronics and mechanics, with every discipline to emphasize on its own approaches, methodologies and tools. Moreover, vocabularies used in processes and methodologies are different making the collaboration between the disciplines a great challenge.

System development activities such as requirements and design specification, implementation and verification are well defined in software engineering. In terms of software engineering the term "design specification" is used to refer to the various models that are produced during the design process and describe the various models of the proposed solutions. Therefore, "design specifications" are descriptions of solution space while "analysis specifications" are descriptions of problem space. Analysis specifications include both requirements specifications but also the problem space structuring that is represented by analysis models such as class diagrams that capture the key concepts of the problem domain and in this sense they provide a specific structuring of the problem space. Dorst and Cross (2001) emphasize on the co-evolution of problem and solution spaces up to the point of time that a match will be found, and consider artefacts of both spaces very important. However, it is clear that the most important are the ones of the problem space since it is not possible to have an acceptable system even with the best solution space if this is based on an incorrect problem space formulation. On the other hand it is possible to have an acceptable system, even without a very good solution space formulation assuming that this is based on a correct problem space model. It is clear that both activities i.e. those involved in problem space formulation (problem space structuring) as well as those involved in solution space formulation include design decisions. Requirements engineering along with domain analysis improve the knowledge on the problem space and make the reasoning steps during the subsequent design process more effective. A well defined model-driven process in software engineering might result at the end to the automatic synthesis of the executable code [Douglas 2006]. Such type of formal processes has not been obtained in engineering design. This is due to the fact that knowledge in engineering design is still more empirical and concepts are not defined with the precision and uniformity as in the software engineering domain. In addition, the role of knowledge management in the early development phases has to be intensified due to increased systems complexity, and more demanding productivity requirements and competitiveness.

Raw requirements, i.e. requirements that have not been analyzed and formulated, are usually expressed in narrative format. These narrative requirements (NR) provide the foundation for the design efforts but do not necessarily provide the complete knowledge required for the subsequent design process. Therefore it is important to analyze and formulate them to become abstract, unambiguous, traceable, and validatable, i.e. well-formed. Several approaches are proposed to refine and extend requirements [Otto et al. 2001, Ulman 2002, Pahl et al. 2007]. According to these, information processing during the initial development phases translates NR to a formalized requirement specification, which should be complete, unambiguous, precise, and verifiable. But these proposals are document centric and very often, labour intensive. Moreover, requirement engineering activities are information-driven and knowledge oriented. There are many characteristics associated with NR information. These have to be analyzed to find relationship between the vast information in order to be structured to facilitate the design activity [Pahl et al. 2007]. Knowledge acquisition, sharing, and integration are among the activities that enable the designer to effectively address complexity in product development. The required information and knowledge is barely explicitly accounted for in the early development phases, and the designer has to collect the required knowledge from various sources.

The approach presented in this paper aims to formalize requirements which are expressed in narrative format, as is usually the case for customer requirements. It is a model-centric approach on refining and extending NR, and is based on an integrated framework of logical reasoning and incidence matrix operations described in [Tomiyama et al. 2002, Kusiack 1992]. The approach is composed of two major sub-processes: that the first models the relevant to the requirements checklist (RC) information and knowledge into a single knowledge framework; and the second formalizes the narrative system requirements using the knowledge framework obtained from the first sub-process. It is not an intention of this work to elaborate on the first stage of requirements elicitation.

The whole approach is considered as an extension of the SysML way of capturing requirements. SysML is exploited in [Thramboulidis 2009] to propose an architectural framework to address the challenge of synergistic integration in engineering design. The 3+1 SysML view model architecture is proposed to define a framework for the synergistic integration of the various disciplines involved in the development of Mechatronic Systems. SysML requirement diagrams are used in this architecture as a means for capturing system requirements. However, such a specification does not take into account the work that has been done during the last years in mechanical engineering design. The approach presented in this paper exploits the existing knowledge on requirements engineering in the mechanical engineering domain and integrates it with the SysML formalism to provide a solid requirements specification for the 3+1 SysML view model architecture, to be used in the subsequent development phases.

The remainder of this paper is structured as follows: In section 2, background and related work are given. The proposed approach for the requirements formalization is presented in section 3. A case study is presented in section 4 to demonstrate the applicability of the proposed approach. Finally conclusions and future work are given in the last section.

2. Theoretical background

2.1 State of the art in problem structuring

There is no single universally acclaimed sequence of steps in engineering design. Researchers have outlined the design process in as few as 4 steps to as many as 25 steps. A complete design process includes, as the first phase, conceptual design which is the process by which the design is initiated and carried to the point of creating a number of possible solutions. The discrete activities considered under conceptual design are, identification of customer needs (IoC), problem definition (PD), information gathering (IG), conceptualization, concept selection, refinement, and design review. This paper focuses on PD and IG and uses the term task clarification or problem structuring (PS). The first three activities are used to refer to activities that are known in software engineering as requirements elicitation, requirements refinement and formalization, while the latter activities are equivalent to architectural design in the software engineering domain. PS is based on three elements: 1) designers

strategy in PS, 2) the design requirements generated, and 3) the information access [Restrepo et al. 2004]. All three elements are crucial and need a better formalism. Dorst and Cross (2001) in extensive creative design practice studies have come out with two notional design "spaces" – problem space and solution space. These spaces co-evolve with interchange of information between them. PS is a process of drawing upon knowledge (or external information) to compensate for missing information and using it to construct the problem space. Requirements are used to specify the design assignment (defining the problem space) and to constraint the desired solution (exploring the solution space) and are therefore an important aspect of PS [Restrepo et al. 2004]. They are often given in narrative format as design problems (narrative requirements). The design problem can be characterized as not being subjected to systematization, incomplete, vague and there is a lack of information in each of the three components stated. NR is dynamically generated during the design process from a design requirement into a design specification (DS). The requirement formalism is triggered by prior knowledge or by knowledge through interaction with design object or with external sources of information. Requirements are discriminated to functional requirements and non-functional requirements or constraints.
Several approaches that use decomposition to enhance dynamic generation of requirements include the use of RC and function models (scenario creation) [Cross, 2000, Ulman 2002, Otto et al. 2001, Pahl et al. 2007]. Both methods are effective for design specifications and analysis, i.e. problem space structuring. Engineering design is an activity that requires both logic and creativity. Systematic methods and tools have continuously been created to better conduct the logical analysis in order to unleash the designer to engage in the creative aspects of problem solving. Research in different design disciplines has produced models concentrated on different aspects of design process [Cross, 2000, Restrepo et al. 2004]. For instance, architecture models propose that solution concepts go before problem structuring. Models from software design propose the designer negotiating the structure of the design problem. Engineering design provide models, with the basis that problem analysis precedes synthesis of solution [Restrepo et al. 2004, Pahl et al. 2007]. In a complex system design, the design strategy is not discipline specific and a combination of these strategies is crucial. The RC is generic based on design goals, constraints and all relevant system parameters and information needed for a successful system design. RC and function models are used not only by Pahl and Beitz (2007); various engineering textbooks make use of the idea of the checklist with different names and categories. RC such as the one shown in table 1 provides a decomposition strategy useful for logical analysis and intelligent system application for a combination of the different model strategies in requirement formalism. The RC is used to facilitate the shift from a document-centric to a model-centric problem structuring. Rational design methods unlike creative methods encourage a systematic approach to design. Checklist is the simplest kind of rational design method. It externalizes the requirement process so that important design issues are not overlooked, and formalizes the process by making a record of items which can be analysed and checked-off as they are completed. It also allows sub-division of task such as allocation task to different team members [Cross, 2000].

Table 1. Part of a requirements checklist [Pahl et al., 2007]

Category	Example
Geometry	Size, height, length, diameter, space, connection, arrangement
Forces	Direction, magnitude, frequencies, Weight, load, deformation, stiffness, elasticity, inertia, resonance
Energy	Output, efficiency, loss, friction, ventilation, state, pressure, temperature, heating, cooling, supply,
Material	Flow and transport, physical and chemical properties, design for manufacturing (DFM)
Costs	Maximum permissible manufacturing cost, cost of tooling, material cost, time,
Schedules	Time constraints, end date of development, project planning and control, delivery date

2.2 Related work

Many approaches exist for NR processing, refinement and formalization. A graph based language has been used to describe the behavioural specifications of a design as well as the behaviour of the available physical components.

An application of the concept of Quality Function Deployment (QFD) to improve product quality based on customer requirements has been described [Cross 2000, Otto et al. 2001, Ulman 2002]. Kusiack [Kusiack 1992] utilizes logic and matrix to describe requirements and transform them into function specifications. SysML, a flexible modelling language derived from UML, supports specification, analysis, design, verification, and validation of a broad range of complex systems. Requirements are defined in SysML in the form of diagrams. The requirements diagram can depict the requirements in graphical, tabular, or tree structure format. Tomiyama et al. (2002) have identified the following knowledge/information operations in a design phase: knowledge/information acquisition, reorganizing, confirmation, revision, conflict resolution, solution synthesis, and object analysis. The approach of Kusiack and Tomiyama seems promising as it also manipulates information using basic logical procedures. Since requirement activities are information driven and the requirement checklist is knowledge oriented we built on these approaches.

In summary, most of the requirements are modelled based on small parts of the information needed in the product development process. For instance the QFD method is best for collecting and refining functional requirements hence the "F" in its name. In the QFD the "how" step consists of the design specification and this must be modelled beforehand.

3. The proposed approach

In this section, we describe our approach for requirements formalization. The approach is motivated by the need to better organize the knowledge necessary to enhance the ability of the designer to express the proposed solution. By logically formalising the requirement process, a computable requirement formulation model can be arrived.

3.1 The basis

The transformation of NR to DS is modelled by logical formalism. Logic basically has two types of reasoning which is by deduction and reduction. A mathematical logical reasoning scheme is as shown in equation (1) [Tomiyama et al. 2002]:

$$A \vdash_{\sigma} T_h \tag{1}$$

Equation (1) shows the simplest set-up of reasoning where deduction is a reasoning process to derive axioms (A) from given theorems (T_h) using inference rule "σ" (*modus ponens)*. The sets A and T_h consist of logical formulae. The symbol "⊢" denotes "deducible". Axioms A is decomposed as follows:

$$A = K \oplus F_d \tag{2}$$

$$F_d \in T_d \tag{3}$$

Where K is knowledge, F_d is a set of "definition facts", and T_d is semantic distinction of definitions. Observed facts F_o can be considered as part of T_h such that:

$$T_h \supset F_o \in T_o \tag{4}$$

Where T_o is statement about observations in extra-logic world. Therefore equation (1) can be written as:

$$K \cup F_d \vdash_{\sigma} T_h \tag{5}$$

Deduction is to obtain T_h from K and F_d, and abduction could be performed in the following three reasoning mode:

1. Obtain K from Fd from partially given Th
2. Obtain K from Fd and partially given Th
3. Obtain Fd from K and partially given Th

In order to clarify the relationship between these logical operations and reasoning in the requirement formulation process, we adopt the model-based abduction method. Abduction in design is a process to derive design solution (entity) from required properties [Tomiyama et al. 2002]. To formalize abduction in design, design knowledge (i.e., axioms) is classified by:

- Knowledge that describes an entity´s functions: $e \rightarrow f$
- Knowledge that describes an entity´s property: $e \rightarrow p$

Where e denotes an entity, p denotes an attribute, and f denotes a function. Both these types of models can be used bidirectional.
The algorithm of model-based abduction is outlined as follows:
The process adopted is to derive solution candidates from axiom K_i:

1. Set requirements that can be treated by axioms Ki as theory Thi (equation 5)
$$K_i\{e_1 \rightarrow p_1, e_1 \rightarrow p_2 \ldots e_2 \rightarrow f_1 ..\} \cup F_{di} \vdash_\sigma T_{hi}\{p_\mathrm{i}, f_\mathrm{i}\} \tag{6}$$
2. Derive solution candidates by abduction with assumptions. Incidence matrix is used to establish the relationship between Fdi and Ki.
$$F_{di} := \left(a_{ij}\right)_{(mxn)};\ a_{ij} = \begin{cases} 1 & if\ e_i\ belongs\ to\ f_i\ or\ p_i \\ 0 & if\ e_i\ does\ not\ belong\ to\ f_i\ or\ p_i \end{cases} \tag{7}$$
The matrix includes both the simple one-to-one relationship (equation 7) and a more complex interaction. Each nonzero entry in the matrix has a value αij ≥ 1. For αij > 1, Fdi is satisfied by multiple categories from Ki. Thus a Fdi can appear in more than one Ki-axiom.
3. Analyze Fdi in the attribute and function space by deduction using Ki . This will enrich Thi.

In the next sections, we formalize the algorithm of model-based abduction as operations to the formalization of the RC and the narrative system requirement.

3.2 Requirements checklist modelling

An application of the logical formalism in requirement formulation is shown in figure 1. Information access is one of the foundation of PS. Choices of the designers during problem structuring depend partly on obtaining proper information and having easy access to information. Relevance is a product of the interaction between the designer and the information source. Improving the relevance of the information retrieved by a system will improve its accessibility [Restrepo et al. 2004].

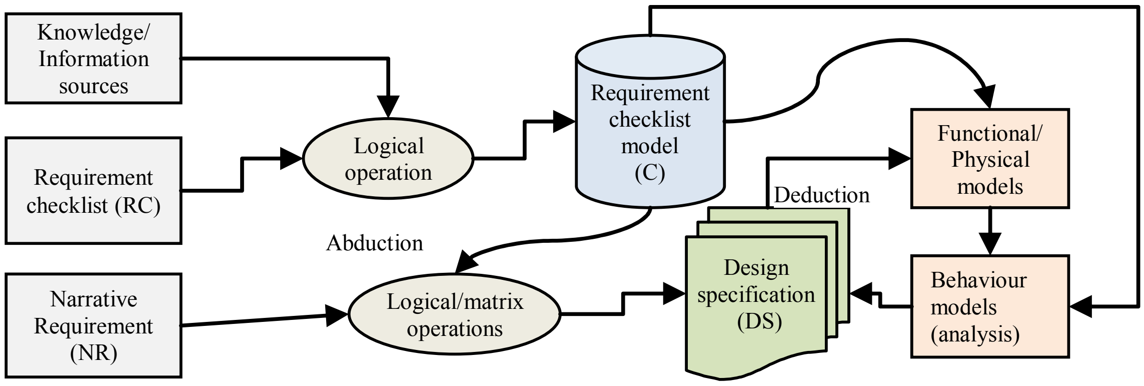

Knowledge/ Information sources
Requirement checklist (RC)
Narrative Requirement (NR)
Logical operation
Logical/matrix operations
Abduction
Requirement checklist model (C)
Deduction
Design specification (DS)
Functional/ Physical models
Behaviour models (analysis)

Figure 1. Logical approach to requirement checklist model and requirement formalization

To deal with the knowledge and relevant information required to synthesise the NR, we propose to organize them based on categories as the ones defined in existing RCs, for instance, as the one shown in Table 1. Since knowledge base is domain specific, the RC is first analyzed to identify additional categories. These categories are further expanded based on specific domain (for instance the car industries have its own specific knowledge/information apart from the general knowledge for requirement formulation). In the RC modelling, various design relevant knowledge/information are integrated and stored in a single work space.

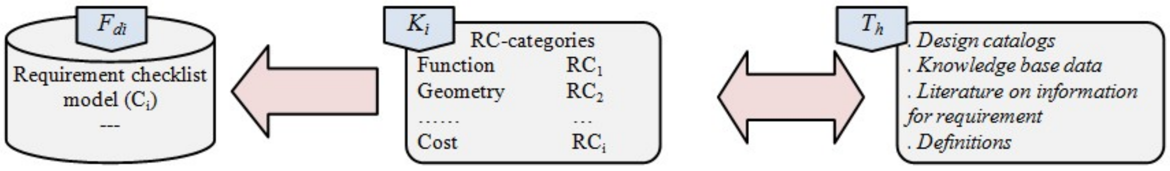
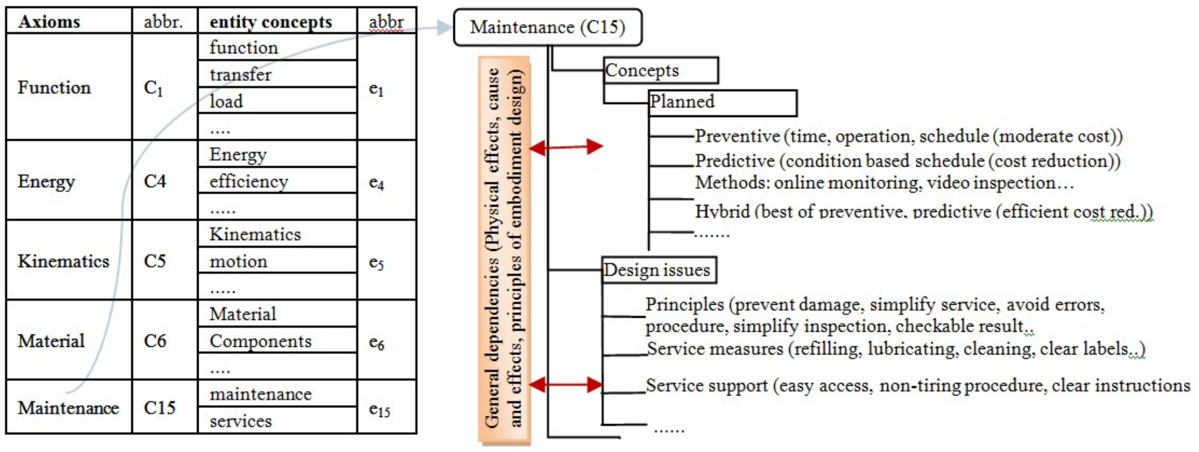


Figure 2. Logical interpretation of requirement category model

The algorithm of model-based abduction is formalized as operations to the RC modelling as shown in figure 2. The steps included are the following:

- Step-1: Set up sources of knowledge/information for requirement formulation in the working space (T_h)
- Step-2: Select identified categories (RC_i) to derive abstract entity concepts from T_h (relevant reasoning)
- Step-3: build aspect-specific RC model (C_i) using RC_i. Transport information from T_h and other modellers, if necessary. This corresponds to enriching information about RC and obtaining a specialized RC model as the one shown in Figure 3.



Axioms	abbr.	entity concepts	abbr
Function	C_1	function	e_1
		transfer	
		load	
		....	
Energy	C4	Energy	e_4
		efficiency	
		.....	
Kinematics	C5	Kinematics	e_5
		motion	
		.....	
Material	C6	Material	e_6
		Components	
		....	
Maintenance	C15	maintenance	e_{15}
		services	

Maintenance (C15)
General dependencies (Physical effects, cause and effects, principles of embodiment design)
Concepts
Planned
Preventive (time, operation, schedule (moderate cost))
Predictive (condition based schedule (cost reduction))
Methods: online monitoring, video inspection…
Hybrid (best of preventive, predictive (efficient cost red.))
....
Design issues
Principles (prevent damage, simplify service, avoid errors, procedure, simplify inspection, checkable result..
Service measures (refilling, lubricating, cleaning, clear labels..)
Service support (easy access, non-tiring procedure, clear instructions
......

Figure 3. Part of a requirement checklist model

3.3 Requirements formalization

The algorithm of model-based abduction is formalized as operations to the requirement formulation framework outlined in figure 1. In the procedure, first from a given NR, an initial requirement (R_i) is set up in the theorem (T_h) working space. Verbs, adverbs (f_i, - function concept) nouns, and adjectives (p_i, - attribute concepts) are then identified in R_i. This is to help identify the key list in R which corresponds to entity concepts (e_i) (Figure 3) of the modelled checklist (C). C is selected in the knowledge base (K_i) domain. Entities e_i from C are utilized to derive abstract concepts (f_i, p_i in R) from T_h. If there is no more e_i to be used, then the operation is terminated and e_i is mapped onto f_i and p_i. The interaction between R and C is modelled with incidence matrix. A design specification list table (DS) is then developed for the selected categories of C and the identified requirements.

The next stage involves analysing the identified initial requirements. This is done using knowledge and information from C. First requirements under the functional performance category in the DS-table are analyzed using the inference "if then", i.e. If sub function A is present, Then sub function B, relationship to arrange sub functions in a logical order. Physical entities in the sub functions are decomposed; in this way, the complexity of the sub-function is reduced. The function requirement is rearranged according to the action necessary to attain that particular function. Next, the physical parameters and subsequent dimensional quantities of the various physical entities are identified. The DS-table is then updated with the newly identified requirements.

In the next step, the non-functional parameters in the DS-table are analyzed. Most requirements under this category are often unbounded, i.e. making relative statements that cannot be verified, as for example “simple maintenance”. These requirements have to be restated defining specific bounds in order to transform them to be validatable. In this case, the requirement is first developed by identifying essential information from C. This information is further refined also using information from C, thus in this way the requirements are defined systematically. The DS-table is then updated. This process is iterative and the procedure described above is applied with other sets of requirements in the table to make DS information well-formed.
The validity of the established requirements is confirmed first by creating specific function models based on the flow of energy, material and signal expressed with the use of a block diagram. The function models which express solution-neutral relationships between inputs and outputs of the system can be numerous. But the above synthesis process allows the narrowing of the range of models. Next the physical relationship of the function model is established using both the physical and dimensional parameters. The behaviours of the function and physical models are then modelled, simulated and compared. The established requirements are arranged in a clear order and in the case of a complex system; they are further assigned into identifiable subsystems, functions, or assemblies. The proposed formalism can be applied for structuring of different types of design problems independent of engineering discipline. Design problem may apply to different design modules namely, original (novelty), adaptive (using established solution principles) or variant (modular) design. In this article the approach is applied to a case study of an ongoing project in an adaptive design of a mobile work machine.

4. Case study: The case of an underground work machine

To understand the approach and formalism presented in the previous section, we consider as a case study the development of a new version of an underground work machine used for loading and carrying soil in a forward motion uphill a certain distance to dump and reverse downhill for more soil.

Table 2. a) Example of a narrative requirement and b) Formulated initial requirements

Underground work machine	
Required: a machine to load and transfer soil uphill a total distance of 300m to dump soil with following characteristics:	
Efficient energy consumption	
Easy to maintain	
Maximum machine length	7417mm
Maximum machine width	1473mm
Maximum machine height	2143mm
Maximum operating load	4000kg
Attention to be paid to safety and environmental issues. Cost should not exceed 50,000 euro. Finished product should be marketed within 2 years.	

abbr.	Requirements	
R1	load , transfer and dump soil uphill	*f1,f2,f3,P1,P2*
R2	Efficient energy consumption	*P3,P4*
R3	Simple maintenance	P_5
R4	soil transfer distance ≥ 300m	P_6
R5	machine length ≤ 7417mm	P_7
R6	machine width ≤ 1473mm	P_8
R7	machine height ≤ 2143mm	P_9
R8	operating weight ≤ 4000kg	P_{10}
R9	safety	P_{11}
R10	environment issues	P_{12}
R11	Cost < 50,000 euro	P_{13}
R12	Project time < 2 years	P_{14}

The requirements for this new version are to reduce energy consumption and harmful emission as shown in table 2a. Formalization of the narrative customer requirements using our approach is as follows.

- Step-1: initial requirement setup

From the narrative requirement, an initial requirement is formulated as shown in table 2b. Then, function and attribute concepts are identified (Underlined in the initial requirements list)

- Step-2: mapping entity concepts onto the abstract concepts

Entity concepts (e_i) from C (Figure 3) are mapped onto the function and attribute concepts defined in step 1. As an example we take the initial requirement R1 (table 2b) and referring to e_i, we have the following mapping. $\{e_1 \rightarrow f_1, e_1 \rightarrow f_2, e_6 \rightarrow P_1, e_{14} \rightarrow P_2\} \equiv$ (transfer → transfer, load → load, material → soil, operation → uphill) according to Figure 3 and Table 2b.

- Step-3: creating interaction matrix

An incidence matrix is constructed to identify the interactions between R and C as shown in table 3.

Table 3. Part of incidence matrix table representing formal interaction between initial requirement and requirement checklist

		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Function	C1	e_1											
Geometry	C2					e_2	e_2	e_2					
Force	C3								e_3				
Energy	C4		e_4										
Kinematics	C5				e_5								
Material	C6	e_6											
Safety	C8									e_8			
Ergonomics	C9												
Production	C10					e_{10}	e_{10}	e_{10}	e_{10}				
Transport	C13					e_{13}	e_{13}	e_{13}	e_{13}		e_{13}		
Operations	C14	e_{14}							e_{14}		e_{14}		
Maintenance	C15			e_{15}									
Recycling	C16												
Economics	C17											e_{17}	
schedule	C18												e_{18}

The DS- table is then created for the identified requirements under the various C-categories. Due to space limitation not all identified requirements in our example are shown in table 4.

Table 4. Part of design specification table for underground working machine

Date	**Design Specification sheet** **Project: Underground loading machine**	**Responsible**
	Functional performance Load , transfer and dump soil _Constraints_ _Geometry_ Loader length $\leq$ 7417mm Loader width $\leq$ 1473mm Loader height $\leq$ 2143mm _Force_ Operating weight $\leq$ 4000kg _Energy_ Efficient energy consumption _Kinematics_ Distance moved with load $\geq$ 300m _Material_ Soil loader _Safety_ safety _Maintenance_ Simplify maintenance _Economics_ Cost < 50,000euro _Schedule_ Project time > 2 years	

- Step-4: functional performance requirements analysis

The requirement statement “Load, transfer and dump soil” (table 4) is arranged in a logical order based on the inference rule using block diagrams (figure 4a). Entities in individual blocks are then decomposed, for instance as shown in figure 4b the “load soil” sub function will require two geometric entities and the loader may also require 2 entities.
The sub functions are then rearranged in a logical order to include the actions and entities involved in a particular function. Figure 4c shows the basic procedure for loading including a human actor.

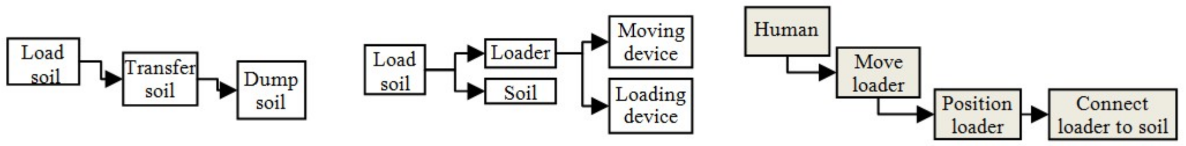


Figure 4. a) Logical order of initial requirement b) Entities required for load function c) Basic procedure for loading

The entities in figure 4c (human, loader, soil) and the functions (move, position, connect) are analyzed using knowledge and information from C. This allows identifying various functions and entities required for the specific process. Examples of “human” and “move” analysis are shown in figure 5.

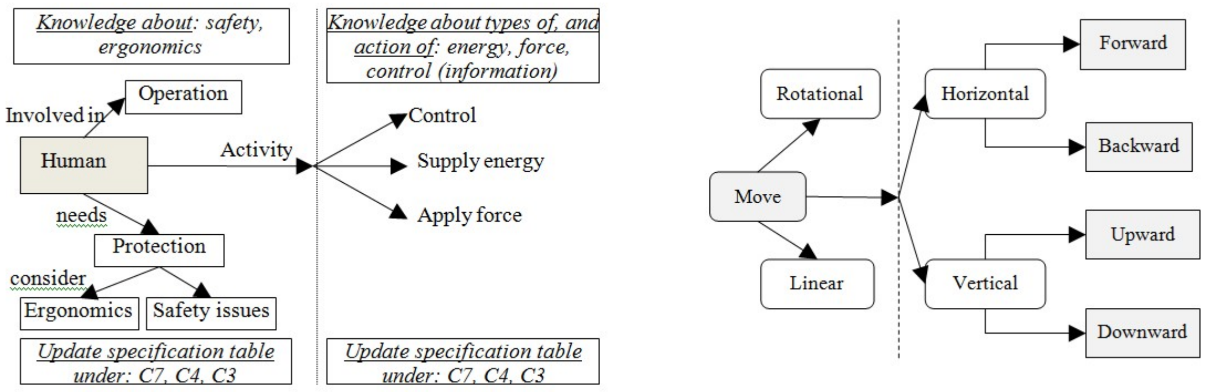

Knowledge about: safety, ergonomics
Knowledge about types of, and action of: energy, force, control (information)
Operation
Involved in
Human
Activity
Control
Supply energy
Apply force
needs
Protection
consider
Ergonomics
Safety issues
Update specification table under: C7, C4, C3
Update specification table under: C7, C4, C3
Rotational
Move
Linear
Horizontal
Vertical
Forward
Backward
Upward
Downward

Figure 5. Analysis of entities and functions in loading machine

- Step-5: Identification of physical parameters and dimensional quantities

The physical parameters and subsequent dimensional quantities of the various entities are then identified and the DS-table is updated.

- Step-6: non-functional requirements analysis

In this step, we take as an example the initial requirement “simplify maintenance” under maintenance category in the DS-table (table 4). This is analyzed using knowledge/information from C15-maintenance (figure 3). The procedure is as shown in figure 6. This procedure is continued to obtain all relevant and identifiable requirements and DS-table is then updated.
The validity of the established requirements is confirmed by first generating function and physical models. The behaviours of these models are simulated and compared. Work on the behaviour simulation and comparison has been done in a previous work [Brace et al. 2009] and interested readers may refer to it. If the evaluation and comparison are acceptable, the DS is asserted and updated. This process is also iterative and continues until a well-formed DS is established.

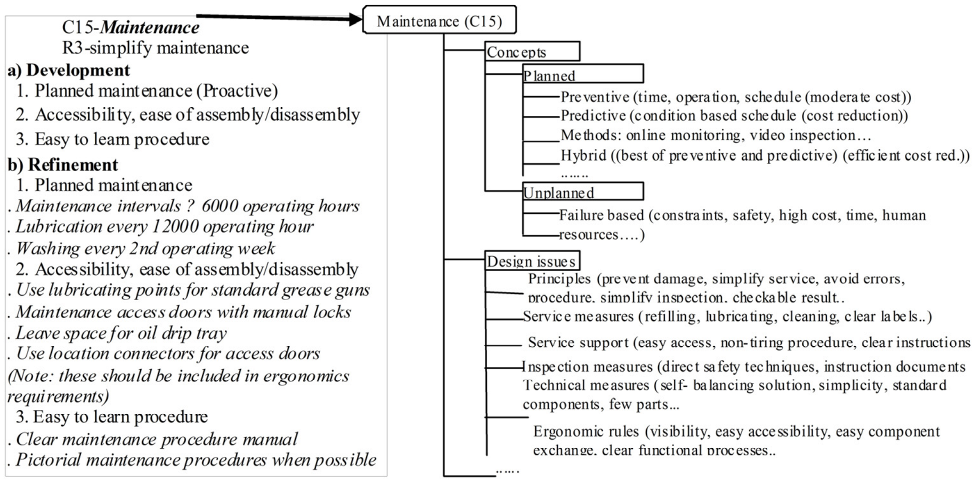

C15-*Maintenance*
R3-simplify maintenance
a) Development
1. Planned maintenance (Proactive)
2. Accessibility, ease of assembly/disassembly
3. Easy to learn procedure
b) Refinement
1. Planned maintenance
. *Maintenance intervals ? 6000 operating hours*
. *Lubrication every 12000 operating hour*
. *Washing every 2nd operating week*
2. Accessibility, ease of assembly/disassembly
. *Use lubricating points for standard grease guns*
. *Maintenance access doors with manual locks*
. *Leave space for oil drip tray*
. *Use location connectors for access doors*
(Note: these should be included in ergonomics requirements)
3. Easy to learn procedure
. *Clear maintenance procedure manual*
. *Pictorial maintenance procedures when possible*
Maintenance (C15)
Concepts
Planned
Preventive (time, operation, schedule (moderate cost))
Predictive (condition based schedule (cost reduction))
Methods: online monitoring, video inspection…
Hybrid ((best of preventive and predictive) (efficient cost red.))
…….
Unplanned
Failure based (constraints, safety, high cost, time, human resources….)
Design issues
Principles (prevent damage, simplify service, avoid errors, procedure, simplify inspection, checkable result..
Service measures (refilling, lubricating, cleaning, clear labels..)
Service support (easy access, non-tiring procedure, clear instructions
Inspection measures (direct safety techniques, instruction documents
Technical measures (self- balancing solution, simplicity, standard components, few parts...
Ergonomic rules (visibility, easy accessibility, easy component exchange, clear functional processes..
……

Figure 6. Non-functional requirement analysis

5. Conclusion

A well formed requirements specification provides a solid basis for the subsequent design phases. Design specifications are the most important artefacts in the product development process given that they are based on a correctly formulated requirements specification. An approach has been described in this paper to formalize the raw requirements that are usually expressed in narrative format. A formal basis for the proposed approach was given taking into account the existing knowledge on requirements management from the mechanical engineering discipline which were combined with current software engineering practices, such as SysML, workflow diagrams and knowledge management. The use of a knowledge rich requirement checklist makes requirement formulation more model-centric and helps the designer to eliminate personal preferences improving this way his effectiveness and productivity. The checklist based requirement specifications are also useful in improving discussion and collaboration in a design team. In systems engineering, it will enhance the system developer to identify and allocate responsibilities based on the checklist categories. The systematic gradual analysis also helps to identify and emit requirements that have no direct bearing on the system functionality and its essential characteristics. This formalized description is our first step towards a semi automated requirements refinement process that will exploit semantic web technologies such as ontologies and knowledge management. Since a first justification of our formal requirements procedure has been established, future work will complete this analysis by focusing on the exploitation of machine readable knowledge representation that will favour the partial automation of the requirements formulization process.

Acknowledgement

This work is part of an ongoing research project "Hybridization of mobile work machines" (Hyblab) funded by the Multidisciplinary Institute of Digitalization and Energy (MIDE) of Aalto University School of Science and Technology.

References

Brace, W., Coatanéa, E., Kauranne, H., Heiska, M., "Early Design Modeling and Simulation of Behaviours: Case study of mobile work machine ", Proceedings of ASME- IDETC 2009, San Diego, USA, 2009.

Cross, N., "Engineering Design Methods Strategies for Product Design" 3rd edition, WileyBlackwell, 2000.

Dorst, K.,Cross, N., "Creativity in the design process: co-evolution of problem–solution". Design Studies, 22(5), pp. 425–437, 2001.

Douglas, S., "Model-Driven Engineering", IEEE Computer, vol. 39,no. 2, pp. 25-31, 2006.

Kusiak, A., ed., "Intelligent Design and Manufacturing", Wiley, New York, 1992.

Otto, K., Wood K., "Product Design: Techniques in Reverse Engineering and New Product Development", Prentice Hall, Upper Saddle River, NJ, 2001.

Pahl, G., Beitz, W., "Engineering Designs: a Systematic Approach. Third Edition", Springer-Verlag, London, 2007.

Restrepo, J., Christiaans, H., "Problem Structuring and Information Access in Design" J. of Design Research, vol. 4, no.2, 2004.

Thramboulidis, K., "The 3+1 SysML View-Model in Model Integrated Mechatronics", Journal of Software Engineering and Applications (JSEA), vol.3, no.2, pp.109-118.

Tomiyama, T., Yoshioka, M., Tsumaya, A., "A Knowledge Operation model of Synthesis", in A. Chakrabarti (ed.): Engineering Design Synthesis - Understanding, Approaches, and Tools, Springer-Verlag, London, 2002, pp. 67-90.

Ullman, D., "The Mechanical Design Process", 3rd edition McGraw Hill, New York, NY. 2002.

William Brace, MSc. Tech.
Research Scientist
Department of Engineering Design and Production
Faculty of Engineering and Architecture, Aalto University School of Science and Technology
Otakaari 4, 00076 AALTO, Finland
Telephone: +358 (0) 9 470 23451
Email: William.brace@tkk.fi