Jurnal
Teknologi

Full Paper

CHALLENGES IN REQUIREMENTS ENGINEERING FOR MECHATRONIC SYSTEMS–PROBLEM ANALYSIS AND FIRST APPROACH

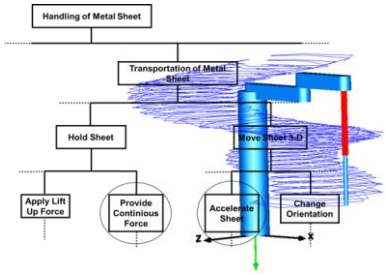
Christopher Lankeit*, Matthias Lochbichler, Ansgar Trächtler

Heinz Nixdorf Institut, University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany

Article history
Received
1 December 2014
Received in revised form
30 April 2015
Accepted
29 June 2015

Corresponding author
*christopher.lankeit@hni.upb.de

Graphical abstract



Abstract

In the development of intelligent mechatronic systems, a gap of tools and methodologies appears to exist, when technical requirements meet physical behavior modeling. Consistency between requirements, development and modeling is not fully achieved and more sophisticated methods seem to be needed. The meaning of requirements in terms of intelligent mechatronic systems is pointed out. Challenges in obtaining technical requirements are worked out by literature review. The influence, identified as most crucial, is domain knowledge. This challenge of the right assumptions considering domain knowledge is reinforced by the, in literature, predominantly applied discrete models. An approach for a solution of these challenges is presented at the end of this paper.

Keywords: Requirements, goals, hierarchy of functions, mechatronic systems, domain knowledge, modeling, continuous models, design methodologies

1.0 INTRODUCTION

Since projects develop to be larger, and systems become bigger, developers work together multidisciplinary. A common understanding of the objective is of vital importance for the system's success. Requirements are predestined to handle that job. Unfortunately, products and processes still fail fairly often [2], which is mostly due to improper requirements [1]. In this paper, a focus is set on *technical requirements*. First, the meaning of *technical requirements* is pointed out, from the point of view of mechatronic engineering. Subsequently, the challenges of adequate domain knowledge are introduced. Furthermore, the problems, which arise with the often performed discretized view on the problem, are mentioned. At the end, a first approach is suggested to use continuous models, to tackle some of the challenges.

2.0 DIFFERENT MEANINGS OF THE TERM "REQUIREMENTS"

The term *requirement* has several meanings in different areas of research and application. This passage is not intended to provide a strict definition of the meaning of the term *requirement*. It is supposed to increase awareness about its different meanings.

According to the IEEE Std. 1998 [3], a system requirements specification should have certain features. Besides others, it needs to be *correct, unambiguous, complete, consistent, verifiable, modifiable, and traceable*. Those are strong conditions comparing to the idea that requirements should be available at the beginning of a development process. Certainly, some ideas exist in the beginning, but these ideas are often not *technical requirements*. Most likely, these ideas are *goals*. However, *goals* of a system are often interpreted as requirements. As van Lamsweerde points out, *goals* mean objectives, which should be achieved by the system [4]. Anyway, there is a gap between *goals* and well defined *technical requirements*.

The *customer requirements* or *user requirements* are not necessarily equal to *technical requirements*. Often, those *customer requirements* are similar to *goals*. Additional information can be found in further literature [5].

Van Lamsweerde presents reasons for *goals* to be important in requirements engineering (RE) [4]. One of them is that *goals* offer precise criterions for sufficient completeness of requirements. It is mentioned that it is worked on using goals for deriving and refining architectures and to annotate design patterns.

Goals may represent functions, which are to be realized [6]. From that point of view *functional requirements*, which define what functions need to be done to accomplish the objectives [7] become relevant. The *performance requirements* define, how well a function needs to be fulfilled. Subsequently, *interface requirements* describe internal and external interfaces. These could be *acceleration, vibration, shock, static loads, acoustic, thermal, electromagnetic interference, electromagnetic compatibility*, just to name a few [7]. The intersection of *functional*, *performance*, and *interface requirements* is, where this paper sees the term *technical requirements* in the context of mechatronic engineering.

There are several additional types and definitions of requirements, but those are not in the focus of this paper.

3.0 CHALLENGES IN REQUIREMENTS FOR MECHATRONIC SYSTEMS

3.1 Design Methodologies

Design methodologies look like a defined path for requirements to evolve. Estefan gives an overview of some model based systems engineering methodologies [8]. A challenge though, is the detailed concern of multidiscipline phenomena. Inside one discipline, requirements may be identified by experts. However, multidisciplinary system elements influence each other through boundaries of the disciplines. A lack of generic consideration of multidiscipline systems can be pointed out [15]. Of course, more disciplines are accompanied by more necessary domain knowledge. Contiguous to these thoughts, Jung analyzes different development methodologies and states that many of the present approaches for supporting requirements application are not singularly suitable for future development projects, which will be more complex and interdisciplinary [16].

3.2 Domain Knowledge

Challenges of requirements in systems engineering have been analyzed by many authors. Here, we present a small excerpt of their results in terms of domain knowledge.

Walia and Carver [9] identify requirement errors in software engineering literature with the goal to develop a taxonomy of errors. Besides others, the following mistakes are identified: *Not understanding the domain; Lack of proper methods for collecting requirements; Lack of domain knowledge or lack of system knowledge; Incorrect model(s) while trying to construct and analyze solution; Mistakes in developing models for analyzing requirements*.

Zave and Jackson analyze challenges of requirements in detail [10]. One of those problems also is adequate domain knowledge. They say that goals need to be combined with additional information about the environment. The possible impact of wrongly assumed environment conditions is pointed out. Moreover, they investigate the gap between requirements and specifications. Closing this gap is always done with support of domain knowledge [10].

3.3 Models

The importance of models is going to rise according to the MBSE initiative [11], which sees models to replace documents as primary products or artifacts of systems engineering processes. Models that represent knowledge, are probably meant. But models obviously can also be used to gain knowledge.

The problems mentioned above might be tackled by adequately applied modeling. In industry, this is recognized, too. A survey concerning the general situation of system engineering shows that most companies know the impact and possible benefits of model-based system engineering and the importance of requirements, but they know as well that their current positions in these areas are not sufficient [12]. That seems to make the research field of the symbiosis of design methodologies, models, and requirements relevant, not only in a scientific context, but also in the context of industrial demands at the present time.

3.4 Continuous Environment Vs. Discrete Models

When regarded from the perspective of mechatronic engineering, a focused view on software requirements engineering can be perceived in literature. Especially, methods for formal analysis of requirements are located in the software engineering discipline [13]. Hull, Jackson, and Dick particularly point out that "dealing with requirements is an essential part of every engineering endeavor" and "requirements engineering is a subset of systems engineering in general, not just software engineering" [1].

An examination of this and several other research hotspots in requirements engineering is proposed by Cheng and Atlee [14]. Increased reliance on the environment is one of them. They state that "the environment or context in which a software system will run is often the least understand and most uncertain aspect of a proposed system; and RE technologies and tools for reasoning about the integration of physical environment, human behavior, interface devices, and software system are among the least mature" [14]. Formalizations are innately not perfect

approximations of continuous phenomena and "better abstractions are needed to model the behaviors of physical and human entities and their interfaces with computing elements" [14].

4.0 TACKLING THE CHALLENGES THROUGH CONTINUOUS MODELS

4.1 Enriched Hierarchy of Functions

Consistency between requirements, development and modeling cannot be fully achieved by the regarded methods. Anyway, an approach to tackle this, would have advantages for development. It could accelerate processes, help to increase safety, and reduce costs, because boundaries of domain knowledge become indistinct and transitions between the boundaries become more fluent. In the following, a first approach is presented to tackle some of the identified problems.

It is not a novel approach to gain information about a system via modeling, but it seems not to be used during early system design phases. Particularly in the crossover between different design phases, a gap can be identified. This makes consistency holey. To gain and to maintain information, the application of models is convenient. Especially the preservation of knowledge in dynamic models is advantageous, considering a consistent progress inside development processes.

To help developers in early phases of a system development process, partial models are established. An example for a partial model is a *hierarchy of functions*. The *hierarchy of functions* helps to understand the technical problems, which need to be solved. The decomposition of goals into more and more detailed functions works until domain knowledge restricts further decomposition. Overcoming this boundary is one target of the approach presented here. In a first step, it is assumed that continuous dynamic models are anyhow available. Domain knowledge can certainly be stored in those kind of models, and it is not limited to system knowledge in terms of solutions. Those models can also provide knowledge for analyzing the problem domain and to detail the hierarchy of functions. This is because in such a model, missing information can be seen immediately and it is not indistinct. As a result, a *hierarchy of functions* has more content, than it often has in practice.

Mainly two benefits arise with that. The first one is an upgrade of the problem space. The nature of a problem becomes evident, and circular connectedness between functions appears more clearly. Via models, domain knowledge can be made explicit to a certain extend. The second benefit is the conservation of information across a development process. Models can be reused, and problems can be traced. This makes it easier to obtain a sharper formulation of technical requirements.

4.2 Transportation of a Metal Sheet with a SCARA

To illustrate the approach, a simplified application scenario is used. A goal is to transport a metal sheet from one place to another. This is shown illustratively in Figure 1. A lot of technical solutions exist for this task. For better illustration, further assumptions are that a SCARA should be used, and a pneumatic force should be applied. The distance to be covered is 30 cm. The question is, what is needed to know to finally acquire the technical requirements.

As mentioned above, the goal needs to be decomposed into sub functions (Figure 2). When reaching a level of detail, in which domain knowledge becomes more and more important, modeling should be deployed.

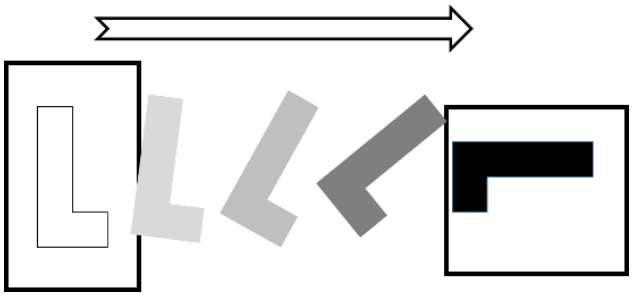


Figure 1 Graphical representation of the goal

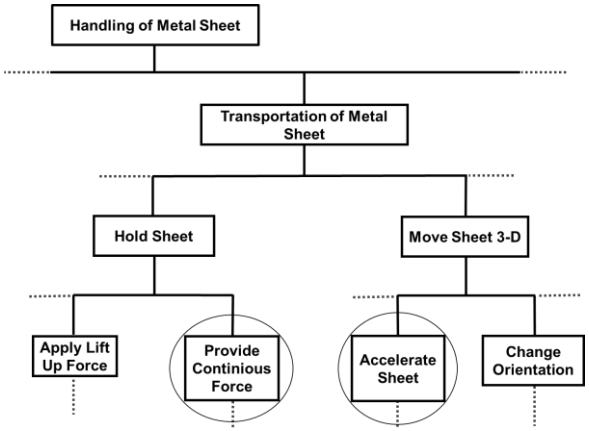


Figure 2 Basic hierarchy of functions

4.3 Modeling to Enrich the Hierarchy of Functions

In this scenario, domain knowledge has a large influence, for example, on the function *Move Sheet 3-D* (Figure 2). The first step of modeling consequently considers this function. A simple model of a SCARA (Figure 3) helps to answer questions about the work space and about kinetic problems, like maximum accelerations. Acceleration obviously influences the required force to hold the metal sheet. The coherence and the magnitude can be determined with this model. Subsequently, the *hierarchy of functions* can be extended, as implied by the circles in Figure 2.

With the prerequisite of a pneumatic generated force, influences on this force need to be determined. A simplified pneumatic model with basic influences can be applied. In this case, the model was available from a previous development. It should be mentioned that the relation between the effort to build the model and its advantages needs to be balanced. This physical motivated model is based on the panel method, which is not described in detail, here. The necessary forces are known, because of the previous simulation of the SCARA model.

In Figure 4, the simulation results of the pneumatic force model can be seen. The influence of the distance between the pneumatic gripper and the workpiece, and of the airflow is shown in a three dimensional plot. The relevance of airflow can therefore be determined. By knowing the required airflow to generate a specific force, the abstract problem has been transformed into a more sophisticated description of the function.

With those analyses, the *hierarchy of functions* can be detailed. Where it is applicable, models can be used to describe functions more precisely and to gain more information. The couplings and interactions of functions can be determined, and the decomposition can be particularized. When the models are linked to the elements of the *hierarchy of functions*, the information is stored.

Hereby, domain knowledge becomes more manageable. This does not mean that management of domain knowledge always is the key to success, as tacit assumptions may occur [17]. But it displays that models have the capability to extend the threshold to domain knowledge, if verified models are available for developers.

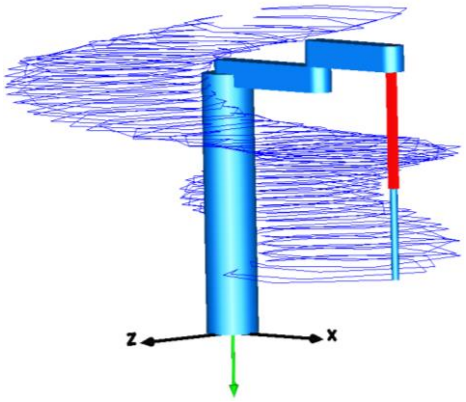


Figure 3 Illustration of the workspace of a SCARA

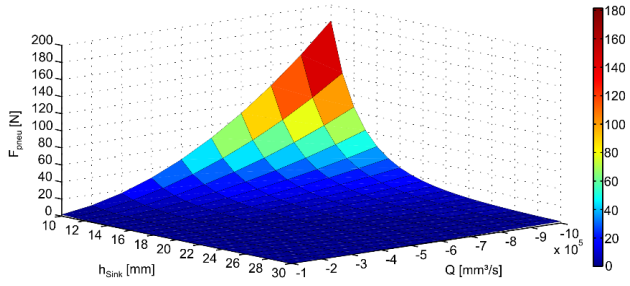


Figure 4 Resulting forces from existing pneumatic model

5.0 SUMMARY

Frameworks provided by development methodologies often try to reduce the negative influence of missing domain knowledge. This paper presents a detailed

view on the challenges of requirements in early design phases. Especially the influence of domain knowledge and continuous models for requirements are investigated with a focus on mechatronic systems.

The state of the scientific research in this area is illustrated in a compendious way. It is shown that an awareness of the presented problem exists and that further research needs to be carried out.

In a condensed description of a first approach, the paper showed the capability of modeling to tackle these challenges. Future work needs to provide a more evolved view on the approach. After all, the benefits of scientific progress in this area can doubtlessly be large.

References

[1] Hull, E., Jackson, K., and Dick, J. 2011. *Requirements Engineering*. Springer London Dordrecht Heidelberg New York.

[2] The Standish Group International Incorporated. 2013. *Chaos Manifesto 2013 Think Big, Act Small*.

[3] The Institute of Electrical and Electronics Engineers, Inc. 1998. *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications,* ISBN 0-7381-0332-2.

[4] Lamsweerde, A. V. 2001. Goal-Oriented Requirements Engineering: A Guided Tour. *Proceedings RE'01, 5th IEEE International Symposium on Requirements Engineering, Toronto*. August 2001. 249-263.

[5] Ahrens, G. 2000. *Das Erfassen und Handhaben von Produktanforderungen*. Berlin.

[6] Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahlking, I. and Angel, S. 1977. *A Pattern Language – Towns, Buildings, Construction*. 1st edition. Oxford University Press.

[7] National Aeronautics and Space Administration. 2007. *NASA Systems Engineering Handbook*. ISBN 978-0-16-079747-7. Washington, D.C.

[8] Estefan, J. A. 2008. *Survey of Model-Based Systems Engineering (MBSE)Methodologies.* INCOSE MBSE Initiative.

[9] Walia, G. S. and Carver, J. C. 2009. A Systematic Literature Review to Identify and Classify Software Requirements Errors. *Information and Software Technology*. 51: 1087-1109.

[10] Zave, P. and Jackson, M. 1997. Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology*. 6(1): 1-30.

[11] Friedenthal, S., Griego, R., Sampson, M. 2007. ICOSE Model Based Systems Engineering (MBSE) Initiative, *INCOSE 2007, San Diego, USA*.

[12] Gausemeier, J., Dumitrescu, R., Steffen, D. 2013. *Systems Engineering in der industriellen Praxis, Paderborn*.

[13] Frappier, M. and Habrias, H. 2006. *A Comparison of the Specification Methods.* Software Specification Methods: an Overview Using a Case Study. ISTE. Hermes Science Publishing.

[14] Cheng, B. H. C., Atlee, J. M. 2007. Research Directions in Requirements Engineering. *Future of Software Engineering (FOSE'07), IEEE*.

[15] Birkhofer, H. 2011. *The Future of Design Methodology*. Springer, London.

[16] Jung, C. 2006. *Anforderungsklärung in interdisziplinärer Entwicklungsumgebung*. München.

[17] Niknafs, A. and Berry, D. M. 2012. The Impact of Domain Knowledge on the Effectiveness of Requirements Idea Generation during Requirements Elicitation, *Requirements Engineering Conference 2012 (RE 2012)*. Chicago, USA.