ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-977

From Formal Requirements on Technical Systems to Complete Designs - A Holistic Approach

Björn Böttcher[1] and **Natalia Moriz** [2] and **Oliver Niggemann**[2]

Abstract. The design processes of todays more and more complex automation systems require computer-based support to maintain their manageability. As a base for that, the authors introduce a holistic design approach for these systems. Requirements on the system to be designed are represented by an extended feature model which serves as consistent requirements model during the entire design process. A grammar-based synthesis applies formalised expert knowledge to generate solutions to these requirements. The paper's main contribution is to combine formalisms from overlapping areas of artificial intelligence and software engineering to obtain a holistic design process for industrial automation systems.

1 Introduction

The paper's focus is on the automated design of industrial automation systems based on formalised requirements. In this work, an industrial automation system is defined as the system that realises the correct behaviour for the control of a technical process by connecting control devices, software function blocks, sensors and actors.
Up until now, there is no well established computer-based support for the following issues during automated design: (*1*) retrieval of previous design knowledge, (*2*) adaption and combination of previous design knowledge into new designs, (*3*) validation of designs in concern to the requirements, (*4*) explicit computation of all valid design variants [2].
The authors present an approach to combine analogy- and grammar-based design to overcome all four listed issues. Issues (*1*) and (*2*) are addressed by the field of analogy-based design whereas the other two are situated within grammar-based design by the presented approach. This is accomplished by applying the analogy-based design at the requirements level, so that instead of designs formalised requirements are reused. A grammar-based synthesis applies formalised expert knowledge in form of rules to generate solutions to these requirements.

2 State of the art

In [2] a broad and precise survey on computer-based design is presented. According to that, the field of analogy-based design has its strengths in the reuse of existing designs. The function- and grammar-based approaches could improve the application of design knowledge within the analogy-based approach. The key issues identified by [2] are that grammars are very promising for formal knowledge-representation but that there is a lack of a common language and tools for supporting designers to formalise their knowledge. Table 1 compares the three major design synthesis approaches. The review on "research directions in requirements engineering" [3] shows a lack of formal notations and formalisms for the validation and verification of requirements. It is a common agreement in design research that designers need a kind of sketch during the conceptual design phase. The main purpose of this initial model is that the designer becomes aware of latent requirements [5]. Additionally to such initial sketches, the reuse of common requirements is of huge importance for regarding requirements in a more rigorous and systematic way [3]. According to [1, 4, 8], product lines in form of feature models (FM) precisely fulfil industries needs concerning such reuse of requirements (see table 1). In [4] FMs are considered as a bridge between requirements and design.
A comprehensive review on the field of FM is presented in [1]. A FM is a hierarchically ordered set of features that symbolise "a prominent characteristic of a product" [1]. The expressiveness of FMs can be enhanced by Extended Feature Models (EFM) which expand features with attributes and constraints. In [1] it is mentioned that EFMs can be represented as Constraint Satisfaction Problems (CSP).

3 Solution Approach

This section presents the holistic design approach assisted design for automation systems (AD4AS). In the following, AD4AS is explained by using the example of a flying saw. Such a saw typically consists of a conveyor for materials and an attached sledge supporting a circular saw. For applying one cut, the sledge gets accelerated from its starting position in parallel with the conveyor. When the sledge's speed equals the conveyor's speed, the saw cuts the material and returns to its starting position afterwards.
The task at hand is to formalise the requirements for the saw, to verify their consistency and to generate at least one appropriate automation system solution. According to the analysed industrial development-projects, human designers proceed as follows to develop the automation system of such a saw:
(*1*) The designer retrieves predefined, textual project-documents with solutions for cutting continuous material. These documents describe comprehensive, technical solutions which summarise an enterprises' experiences for certain fields of application. (*2*) The designer then follows the instructions of the selected solutions and adopts them to the present requirements. (*3*) Based on his expert knowledge, experiences and these instructions, he designs a customer-specific plan for the saw's automation system. (*4*) For realising this plan, he has to retrieve appropriate components and products.
AD4AS reflects these steps and defines the following proceeding to

[1] Fraunhofer Application Center Industrial Automation (INA), Germany
[2] Institute of Industrial Information Technologies (inIT), Germany

	requirements-design separation	requirements validation	knowledge representation	design variants	reuse of designs
function-based	yes	no	informal	yes	yes
grammar-based	yes	yes	formal	yes	no
analogy-based	weak	weak	informal	no	yes
product line	yes	yes	formal	yes	yes

Table 1. Comparison of the three major design synthesis themes [2] and product line (regarded as a configurable set of predefined designs)

plan the saw's automation system (see figure 2).

(*1*) As observed in the above analysis and cited in section 2, the reuse of common requirements is of huge importance for practice but theres a lack of appropriate formalisms. AD4AS solves this by splitting requirements into so called templates. These templates are represented by FMs and already contain reusable engineering experience in form of dependencies and constraints. In this first step, templates are combined into an initial model of required entities which is also a FM. This model is refined until the reasonable modelling depth for further design steps is reached.

(*2*) Based on the model created in step 1, more specific requirements such as to synchronise two process steps are added. Therefore, the process steps have appropriate properties. In AD4AS such properties are modelled as attributes of features so that the FM representing the sketch is an EFM. For the flying saw, synchronisation means that the speeds of conveying and cutting need to be equal during the cutting process. This is expressed as a constraint within the EFM. Figure 1 shows an exemplary requirements model for the saw. In AD4AS

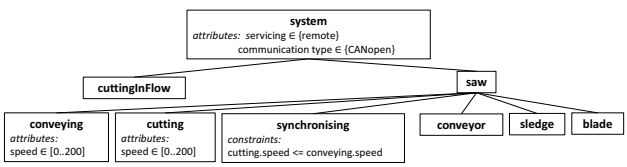

Figure 1. Exemplary requirements model after AD4AS step 2.

EFMs are regarded as a hierarchical composition of required entities with attributes and constraints. The requirements are defined to be consistent if and only if the CSP corresponding to the EFM (see [1]) has at least one solution. By using the CSP formalism, a consistency check of the requirements can be realised with existing constraint solvers.

(*3*) The requirements model resulting from the second step does not define an automation system but contains all validated requirements for it. This model serves as the base for generating an automation system topology of abstract components. Abstract components are placeholders for specific products that are selected in the next step.

This proceeding is based on expert knowledge rules. An example is a rule saying that the process step conveying can be realised by a conveyor with a motor that is powered by a frequency inverter. Another rule is that in a centralised architecture the programmable logic controller (PLC) is connected to each inverter and each motor to exactly one inverter.

As summarised in section 2, grammar-based design is suited for describing such expert knowledge. Based on that, the topology creation in AD4AS is formulated as productions of a graph grammar and all other rules and constraints in form of application conditions (see also [6]). The EFM from the AD4AS step *2* is the host graph for the graph-grammar-based synthesis of topologies. In [6] it is described how such synthesis is accomplished for a similar task. According to that, a topology in this work is a stable graph as defined in definition 7 of [6]. This step results in so called architecture variants which are pairs of a topology and an EFM which features correspond to the abstract components in the topology.

(*4*) In the fourth step, for each architecture variant its EFM has to be instantiated by assigning appropriate products to the features. This is not in the focus of this paper. A similar concept is presented in [7].

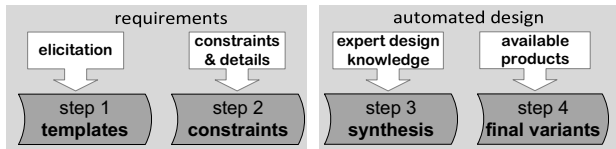

Figure 2. The four design steps of AD4AS.

4 Results & Outlook

Summing up, the first two steps of AD4AS are analogy-based design of a requirements model. The newly introduced templates encapsulate engineering experience and solve industries needs for structuring and formalising of requirements. Requirements are already validated before any solution is generated. General design knowledge in form of rules is expressed as a graph-grammar and seamlessly applied to the EFM describing the requirements. This results in automation system solutions.

Together with the industrial research-partners, the steps for designing the flying saw have been evaluated with a prototype that uses the Gecode constraint-solver framework. The finding is that the human designer is beneficially assisted as intended. Thereby, AD4AS reflects the typical proceeding of human designers.

Additionally to more insights into computational-complexity- and knowledge-formalisation aspects, the authors will present a more detailed description of the synthesis in AD4AS's third step soon.

ACKNOWLEDGEMENTS

Funded by the German Federal Ministry of Education and Research, 'Design Methods for Automation Systems with Model Integration and Automatic Variation Validation', 01M3204B & 16M3204A.

REFERENCES

[1] David Benavides, Sergio Segura, and Antonio Ruiz-Cortés, 'Automated analysis of feature models 20 years later: A literature review', *Information Systems*, **35**(6), 615–636, (2010).

[2] Amaresh Chakrabarti, Kristina Shea, Robert Stone, Jonathan Cagan, Matthew Campbell, Noe V Hernandez, and Kristin L Wood, 'Computer-based design synthesis research: an overview', *Journal of Computing and Information Science in Engineering*, **11**(2), 021003, (2011).

[3] Betty HC Cheng and Joanne M Atlee, 'Research directions in requirements engineering', in *2007 Future of Software Engineering*, pp. 285–303. IEEE Computer Society, (2007).

[4] Krzysztof Czarnecki and Ulrich W Eisenecker, 'Generative programming: methods, tools, and applications', (2000).

[5] John S Gero and Udo Kannengiesser, 'The situated function-behaviour-structure framework', in *Artificial Intelligence in Design*, volume 2, pp. 89–104. ed Norwell, MA, USA: Kluwer Academic Publishers, (2002).

[6] Eric Klavins, 'Directed self-assembly using graph grammars', *Foundations of nanoscience: self assembled architectures and devices, Snowbird, UT*, (2004).

[7] Christoph Ranze, Thorsten Scholz, Thomas Wagner, Andreas Günter, Otthein Herzog, Oliver Hollmann, Christoph Schlieder, and Volker Arlt, 'A structure based configuration tool: Drive solution designer-dsd', in *AAAI/IAAI*, pp. 845–852, (2002).

[8] Valeriy Vyatkin, 'Software engineering in industrial automation: State of the art review', *IEEE Trans. Industrial Informatics*, **9**(3), 1234–1249, (2013).