Addressing Model Complexity in Automotive System Development
Selection of System Model Elements for Allocation of Requirements

Grischa Liebel[1], Andreea Olaru[2], Henrik Lönn[3], Henrik Kaijser[3], Sunith Rajendran[4], Urban Ingelsson[4] and Richard Berntsson Svensson[5]

[1]*Software Engineering Division, Chalmers | University of Gothenburg, Gothenburg, Sweden*

[2]*Semcon Sweden AB, Gothenburg, Sweden*

[3]*Volvo Group, Advanced Technology and Research, Gothenburg, Sweden*

[4]*Semcon Sweden AB, Linköping, Sweden*

[5]*Blekinge Institute of Technology, Karlskrona, Sweden*

Keywords: System Modelling, EAST-ADL, EATOP, Requirements Allocation, Requirements Engineering, Requirements Traceability, Tool Design, Empirical Research.

Abstract: Modern automotive embedded systems are developed by Original Equipment Manufacturers (OEM) together with multiple suppliers. A key problem for a supplier is to allocate an OEM's requirements specification to their own subsystem design. This is a difficult manual task especially on complex systems and it requires expert knowledge about the system design. To address this problem, this paper presents a design science research to develop and evaluate a Requirements Allocation Assistant tool (RAA). The tool provides functionality to search through and filter requirements and system models to enable efficient requirements allocation even in the presence of complexity. RAA is built on top of the EATOP/Eclipse framework using EAST-ADL as system modelling language. The tool was evaluated and validated during a qualitative usability study with 17 engineers active in the Swedish automotive industry. Key findings are that searching is used to learn about a system, whereas filtering is used to narrow down a set of candidate elements of the system design. Engineers request further support in narrowing down a set of candidate elements and in checking that an allocation is correct.

1 INTRODUCTION

The trend towards embedding electronic and software-based control systems into vehicle functions continues, leading to ever more complex systems. In particular, textual requirements specifications of vehicle functions often span thousands of pages. The challenge at hand is to make system development competitively correct, cheap and quick in the presence of high complexity. This includes addressing manual tasks in the system development process.

One important way to address complexity is to enable reuse of system components. To do so, one of the key manual tasks in the development of embedded systems in collaboration between multiple organisations is requirements analysis. In requirements analysis, engineers allocate requirements to an existing subsystem design, to see whether the design can satisfy them. Complexity may obscure the engineer's insight into the subsystem design and make requirements analysis inefficient.

In cognitive load theory, it is found that structure of information is important and support for finding relevant information is helpful (Passera, 2015). Applied to the issue of complexity in requirements analysis, three orthogonal dimensions of support from tools can be considered, namely the visual representation, the structure of the presented information, and restricting the presented information to a manageable subset. In this paper we address complexity, not by attempting to reduce it, but by investigating how the manual task of requirements analysis can be supported by tools, mainly by exploiting the structure of the information and allowing the presented information to be restricted.

Considering the wide application of model-based engineering in the automotive domain (Liebel et al., 2014), it should be considered how modelling can

Liebel, G., Olaru, A., Lönn, H., Kaijser, H., Rajendran, S., Ingelsson, U. and Svensson, R.
Addressing Model Complexity in Automotive System Development - Selection of System Model Elements for Allocation of Requirements.
DOI: 10.5220/0005652101680175
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 168-175
ISBN: 978-989-758-168-7

be employed to address complexity during requirements analysis. Using a well-known modelling language can ease understanding of the modelled system through the semantics of the language.

An upcoming modelling language is EAST-ADL (EAST-ADL Association, 2015), an architecture description language for automotive embedded systems. EAST-ADL specifies how to represent a majority of the system description. A key benefit of representing a system design in a structured information model such as EAST-ADL is that predefined viewpoints can be applied to visualise information in a way that complexity becomes less of an obstacle. Furthermore, a similar structure could be enforced for requirement specifications and for system models, thus increasing the sense of recognition.

Prior work on tool support for source code visualisation has suggested searching and filtering as valuable tool features (Bassil and Keller, 2001). By employing a modelling language, the semantics can be employed to create various types of search (filter) queries. Queries can apply to specific modelling element types, attributes and relationships, as well as to exploit traceability. To our knowledge, no prior work has investigated the use of searching and filtering tool features employing the semantics of a structured modelling language to reduce the cognitive load when performing requirements allocation.

Therefore, our contribution in this paper is to:

- express the need for tool support that reduces the cognitive load when performing manual tasks (specifically requirements analysis) to enable efficient collaborative development of complex embedded systems.
- present a tool prototype called Requirement Allocation Assistant (RAA) that enables searching and filtering of requirements specifications and system models represented in EAST-ADL. RAA is implemented within the EATOP open source tool.
- provide an evaluation of RAA by applying qualitative techniques from usability engineering (Faulkner, 2000).
- present increased understanding of the process of requirements allocation, that may be formulated as a theory to guide further effort towards tool support.

While it is an ambitions goal to provide tool support for requirements analysis to reduce the manual effort, the scope of this paper is restricted to studying two tool features, namely searching and filtering, motivated by a suggestion by (Bassil and Keller, 2001). We formulated two research questions to evaluate the two tool features ant to compare them, to gain understanding of how they align to the manual (cognitive) process of requirements analysis.

RQ 1: How can the use of searching and filtering tool features aid requirement allocation in a system development effort in terms of intuitiveness, helpfulness and decision support?

RQ 2: Which tool feature, searching or filtering, is best aligned to support requirements analysis and why?

2 RELATED WORK

This section discusses the literature and the concepts that are related to this paper. I.e., EAST-ADL is presented as a prerequisite to understanding our tool prototype presented in Section 4 and related publications to this paper are presented thereafter.

2.1 EAST-ADL

EAST-ADL (EAST-ADL Association, 2015) is an architecture description language that has been specified over the last decade. The ambition is to standardise a structured information model that can be employed to represent an automotive embedded system model from concept level to implementation level (Blom et al., 2013). It is aligned with AUTOSAR for software architecture (AUTOSAR, 2015) and with ISO 26262 (ISO, 2011). When developing a system in accordance with ISO 26262, the scope of modelling and requirements specification covers the whole system on each abstraction level, with corresponding traceability. Consequently, each vehicle feature is realised by functions that are in turn realised by design-level components and so on. Similarly requirements of vehicle features are specified into functional requirements and further into requirements and design. These requirements are allocated to the corresponding system model elements by adding a *Satisy* relationship. The *Satisfy* relationship connects a requirement to system model elements that satisfy it. The requirements modelling package of EAST-ADL is aligned with SysML (Object Management Group, 2015).

2.2 Related Publications

Tool support to reduce the cognitive challenge imposed by complexity has been considered before, especially in the realm of software design. Architectural viewpoints which show a particular aspect of a software architecture and excludes other information are a well-known concept (Kruchten, 1995; ISO, 1998) These viewpoints enable engineers to easily recognise key aspects of the software architecture.

In the realm of requirement analysis, (Hofmeister et al., 2005) argue that the complexity of requirements analysis is impacted by a conceptual gap between requirements and architecture. This means that to perform requirements analysis one must first learn about the system design. To lessen this gap, the authors present an approach, based on four viewpoints, guiding the architecture design process. The tool support considered in this paper is complementary to the viewpoints in (Hofmeister et al., 2005), with the novelty of exploiting a structured modelling language.

Telea et al. (Telea et al., 2010) conduct a review of architectural visualisation tools based on three user categories. They report that technical users consider a visualisation tool useful if it provides IDE integration and generates desired custom viewpoints with as few operations as possible. They report that architecture visualisation tools should provide interactive ways to compare, correlate and search data. The authors mention that requirements allocation viewpoints are much harder to implement in a tool because this activity requires the user's active participation.

Bassil and Keller (Bassil and Keller, 2001) present an evaluation of tools for visualising source code to software engineers. Key aspects of these tools include browsing based on the code structure and searching to identify relevant parts of the code.

In this paper we address the challenge to implement a viewpoint for requirements allocation. To bridge the conceptual gap between requirements and system design, we take on the challenge to implement a viewpoint for requirements allocation. The purpose is to support engineers to use their expert knowledge and quickly learn about the system. In contrast to architectural viewpoints, we cannot generally define what information is needed when analysing any given requirement, so it becomes necessary to allow an engineer to use their expert knowledge. Consequently, we develop two tool features, namely searching and filtering, to make the engineer efficient in defining the relevant information, and integrate into an IDE called EATOP. Further, we are exploiting the structure imposed by EAST-ADL to further aid engineers to learn about a given system design.

3 RESEARCH METHODOLOGY

This section presents the research methodology employed in the work presented in this paper, with the purpose of enabling continued research on the same topic with the same or improved methodology.

The investigation presented in this paper was carried out using a *design science research* approach (Hevner et al., 2004). Design science research is the study of artfacts in a context (Hevner et al., 2004) by adding a problem-solving cycle to a theory-building cycle. Design science research aims to develop and evaluate an artefact to meet a specific business need.

We followed Hevner's Design Science framework (Hevner et al., 2004), which consists of two main steps, namely development of an artefact followed by its evaluation. The general idea is to use existing knowledge to develop/build theories/artefacts, where the development step is based on the business needs that originate from the study's environment, which consists of the people, the organisation, and the technology.

3.1 Knowledge Base

The knowledge base was collected from the literature described in Section 2 as well as from knowledge about Eclipse plug-in development and the EATOP project.

3.2 Environment

The study was conducted in the automotive engineering department of Semcon, a global company providing engineering services, as a contribution to the research project Synligare[1]. Semcon provides engineering services to automotive Original Equipment Manufacturers (OEMs) and to their tier-1 suppliers. Sometimes, Semcon helps tier-1 suppliers in allocating requirements from their OEM customer to their own system models. This collaborative development strategy, where requirements and an existing system model are originating from two different companies, is nowadays common practice in the automotive domain and therefore not unique to Semcon.

3.3 Development Process

RAA was developed as a plug-in extending EATOP. It was developed from a given specification by a small team consisting of a computer programmer, an expert on the Eclipse platform, and a number of industrial and academic advisors. The development process was informal and contained informal peer review of design and code. The design of RAA is described in detail in Section 4.

[1]www.synligare.eu

3.4 Evaluation Process

The evaluation of RAA was carried out using usability evaluation techniques (Faulkner, 2000). That is, the evaluation was focused on answering how *intuitive* and *helpful* RAA is, as well as how RAA provides *decision support*.

We conducted a usability study with 17 engineers from the case company and from the consortium of the Swedish research project Synligare. It started with a presentation of EAST-ADL, RAA and concrete examples of using the tool features. After the presentation, the participants were provided with an installation of RAA together with a sample requirements model and system model. During the study, the questions and reflection of the participants were written down in order to be used for analysing the results.

Each participant filled in an anonymous questionnaire containing 13 questions. It started with a section gathering demographic data, and the participants' experience regarding requirements tools with allocation functionality. This part was followed by specific questions about the two searching and filtering tool features.

We analysed the data starting with coding the free-text answers by theme or category, followed by identifying the set of data relevant for each research question. This data provides the reactions to RAA and can be used as evidence of existence of opinions, but it does not tell how representative the opinion is. To get an overview of the data, tabulation (Runeson and Höst, 2009) was used as an analysis technique, i.e. arranging the data in tables based on the research questions. In addition, graphs were used to visualise the data. The analysis results are presented in Section 5.

3.5 Validity Threats

The threats to validity of the study and the claims made based on the collected data are presented here together with possible countermeasures taken to increase validity.

Construct Validity: To avoid misleading or confusing usability study participants, the questionnaire was reviewed by two researchers involved in the study and improved based on their feedback. The usability study was performed using a single station to give all participants the same experience. Consequently, RAA could be used by one participant at a time. This could give the participants the feeling that they were evaluated, making evaluation apprehension a realistic threat. To alleviate this threat, the questionnaire was filled in anonymously at a separate desk, unassisted and unobserved.

External Validity: This study involved participants with various backgrounds and experience from different companies in the automotive industry in Sweden. The participants are potential target users for RAA and their companies do requirements analysis in the context of collaborative development. Thus, the findings of this study may be valid for companies facing similar problems in the automotive and the embedded systems domain. However, organisational and regional culture could have affected the outcomes. We will address this threat in the future by replicating the usability study with engineers of different background.

Internal Validity: Each subject participated only once in the study, which alleviates maturation threats. A possible threat is the participants' lack of knowledge about EAST-ADL, which may distract from achieving an untainted experience of the usability of RAA. To alleviate this threat, we showed each participant an example of a requirement and a system model element such that both contain enough information for a feasible allocation. We also gave a short overview of EAST-ADL.

Conclusion Validity: Usability issues in RAA, other than those considered in the research questions, could have influenced the participants. This threat was alleviated by ensuring clarity in the questionnaire, such that the questions are aligned with the research questions.

4 RAA

RAA is built as a plugin to the EAST-ADL Tool Platform (EATOP[2]) RAA adds tool features on top of EATOP and can be combined with other plugins.

4.1 Architectural Design

RAA is intended to aid requirements analysis in the context of collaborative development of complex automotive embedded systems. It does so by easing the identification of relevant information in system models and by assisting engineers in allocating requirements to elements of existing system models. RAA is developed as a proof of concept of how to employ a structured information model to bridge the conceptual gap between requirements and system design, so that engineers doing requirements allocation can easier learn about the system design and be more efficient.

[2]https://www.eclipse.org/eatop/

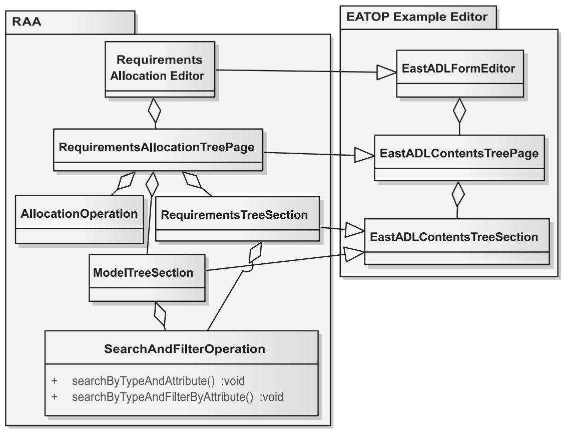
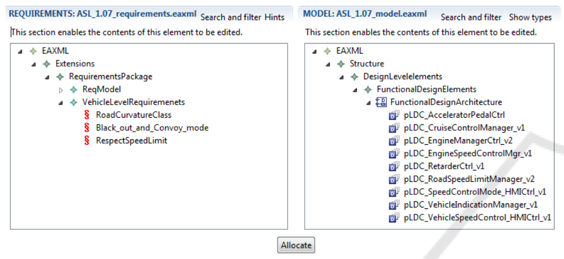

RAA
EATOP Example Editor
Requirements Allocation Editor
EastADLFormEditor
RequirementsAllocationTreePage
EastADLContentsTreePage
AllocationOperation
RequirementsTreeSection
EastADLContentsTreeSection
ModelTreeSection
SearchAndFilterOperation
+ searchByTypeAndAttribute() :void
+ searchByTypeAndFilterByAttribute() :void

Figure 1: Design overview of RAA.


REQUIREMENTS: ASL_1.07_requirements.eaxml Search and filter Hints

This section enables the contents of this element to be edited.

- EAXML
  - Extensions
    - RequirementsPackage
      - ReqModel
      - VehicleLevelRequirements
        - RoadCurvatureClass
        - Black_out_and_Convoy_mode
        - RespectSpeedLimit

MODEL: ASL_1.07_model.eaxml Search and filter Show types

This section enables the contents of this element to be edited.

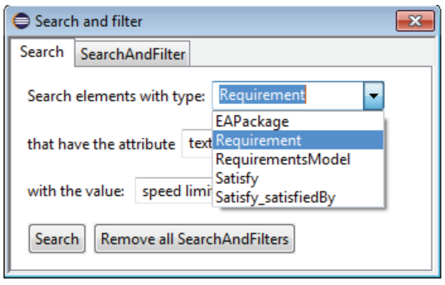
- EAXML
  - Structure
    - DesignLevelelements
      - FunctionalDesignElements
        - FunctionalDesignArchitecture
          - pLDC_AcceleratorPedalCtrl
          - pLDC_CruiseControlManager_v1
          - pLDC_EngineManagerCtrl_v2
          - pLDC_EngineSpeedControlMgr_v1
          - pLDC_RetarderCtrl_v1
          - pLDC_RoadSpeedLimitManager_v2
          - pLDC_SpeedControlMode_HMICtrl_v1
          - pLDC_VehicleIndicationManager_v1
          - pLDC_VehicleSpeedControl_HMICtrl_v1

Figure 2: Main user interface of RAA.

The design of RAA, including the extensions of EATOP and the main selection and allocation tool features, is illustrated in Figure 1.

RAA is an extension of the EATOP Example Editor, which contains a hierarchical view for navigating system model elements. This view (EastADLContentsTree) is instantiated twice in RAA, for showing a requirements specification (RequirementTreeSection) and a system model (ModelTreeSection). An example of the visual appearance of the hierarchical views in RAA is given in Figure 2, with the requirements specification on the left and the system model on the right.

The building blocks for RAA's key tool features are SearchAndFilterOperation and AllocationOperation. The former is instantiated on both hierarchical views, enabling searching and filtering for the requirement specification and for the system model independently. The latter involves elements from both hierarchical views. Therefore, it is architecturally associated with RequirementsAllocationTreePage, a container that holds both hierarchical tree views instances. In the user interface, the AllocationOperation is represented by a button marked Allocate (see Figure 2).



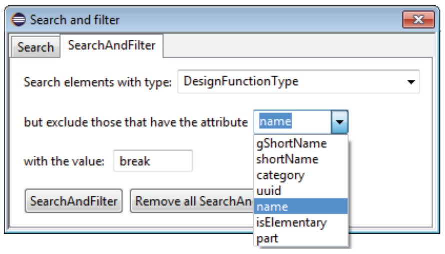
Search elements with type: Requirement

that have the attribute text

with the value: speed limi

EAPackage
Requirement
RequirementsModel
Satisfy
Satisfy_satisfiedBy

Figure 3: Searching among requirements.



Search elements with type: DesignFunctionType

but exclude those that have the attribute name

with the value: break

gShortName
shortName
category
uuid
name
isElementary
part

Figure 4: Filtering among model elements.

4.2 Tool Features

A query for searching consists of an element type and a value for a specific attribute. An example is searching the requirements specification for all requirements (element type) that have in the requirement text (attribute) the string "speed limit" (attribute value) (see Figure 3).

A query for filtering consists of an element type and a value for a specific attribute, such that elements of the chosen element type are included in the result except if the specific attribute has the given value. The example in Figure 4 includes all Design Function Type elements but excludes those that have the word "break" in their name.

In RAA, allocation is performed by pressing the allocation button after selecting at least one requirement in the requirements specification and at least one element in the system model. EAST-ADL offers the possibility to allocate one or many requirements to one or many system model elements through satisfy relationships. For example, allocating one requirement to three elements from the system model means that the selected requirement is satisfied by those three model elements. This allocation would create one *Satisfy* element, containing the reference to the satisfied requirement, and three *satisfiedBy* elements, each pointing at one of the model elements that the requirement is satisfied by.

As our tool handles relationships between two models, there are multiple options on where to store the newly created model elements. The newly created model elements, called *Relations* and *Satisfy1* in

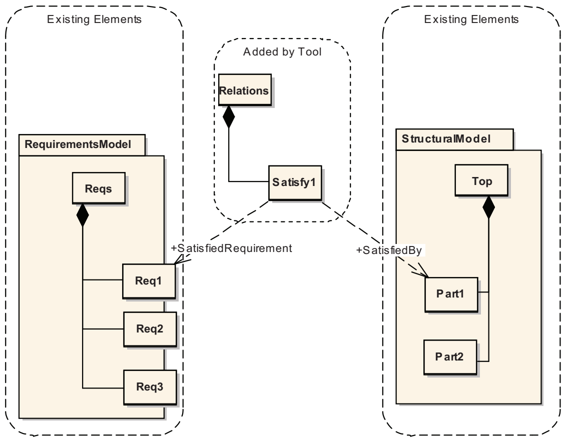


Figure 5: Storage of newly created model elements.

Figure 5, could be stored in the requirements model, in the system model or in a separate model. Additionally, both models could be combined into a single model, including the newly created relationships. Even though we currently only support storage in the requirements model, the remaining scenarios are relevant and will be added in future versions of RAA.

5 EVALUATION

Demographic data of the study participants is presented in Section 5.1, followed by the answers to the research questions in the remaining sections.

5.1 Demographic Data

All participants were engineers in the automotive industry, with various specialisations such as system engineering or management. Eleven of them were from Semcon and six from other companies. The participants had between three and 19 years work experience, with twelve out of 17 participants having prior experience in requirements engineering.

We asked the participants which tool features they deemed important in a tool supporting requirements allocation. All of the practitioners answered that traceability is important. Further, 16 out of 17 answered that searching and filtering tool features are important. Seven participants answered that highlighting information is an important tool feature.

5.2 Aiding Requirements Allocation

Considering Research Question **RQ1**, the intuitiveness, helpfulness and decision support provided by RAA was evaluated by asking the participants to answer direct questions from the questionnaire. There was also opportunity for the participants to leave comments about these aspects of RAA. In the evaluation questionnaire, the participants' opinions regarding the two features were gathered based on three aspects: intuitiveness, helpfulness to find relevant information, and helpfulness in making requirements allocation decisions.

Figure 6 summarises the data regarding the three measured aspects. The count of participants who disagree with the statements is shown in blue and the count of participants who agreed in green. Undecided participants are coloured grey. There are no participants that strongly disagree with any statement. It could be thought obvious that searching and filtering are intuitive and that they help to find relevant information. This may explain the fact that there are many agreements to the statements. For searching and filtering, one reply each voiced disagreement. This may be due to the fact that the searching and filtering features requires search queries that involve specifying an element type and an attribute name. This could be less intuitive as, e.g., free text queries applied to all attributes of all elements, especially for someone who is not familiar with the element types in EAST-ADL. Further, it can be seen that there is more hesitation regarding filtering. Some undecided participants left comments saying that they "usually look for something in particular", rather than attempting to exclude irrelevant information. This observation may reveal a key insight into the thought processes of an engineer performing requirements analysis, leading to a possible answer to the "how" of RQ1. Searching may be intuitive and helpful because it aligns well with the thought processes of the engineer. Filtering, however, is seen as intuitive but used more seldom – a tool feature to use when searching attempts were not successful. A further comment suggested to enable several search and filter queries to be applied in combination, as present in related tools such as DOORS[3].

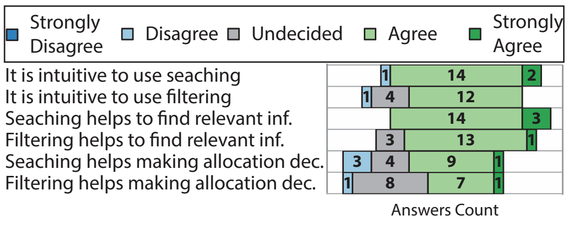


Figure 6: Intuitiveness, helpfulness and decision support of RAA.

Considering the reactions to the statements that searching and filtering respectively helps making decisions about allocation, there are more undecided

[3]http://www.ibm.com/software/products/sv/ratidoorfami

replies. For searching, there are three disagreeing replies. Indeed a participant commented that neither searching nor filtering helps making decisions about allocation. This participant voiced the need for a tool feature to "check if the selected element matches the requirement" and to highlight possible elements in the model that may match the selected requirement. This indicates that while searching and filtering are intuitive and helpful tool features and in this way support decision making, they are not in themselves considered to provide decision support in a strict sense. Another possible insight could be that the perceived challenges in requirements analysis include picking candidate elements for allocation of a requirement and checking if selected elements match a requirement. It could be speculated that the participant with these comments expects that some aspects of requirements analysis could be automated.

5.3 Comparison

To collect data to relate to RQ2, participants were asked to compare searching and filtering. They were asked to compare the tool features with respect to how efficiently they could be employed to find relevant information. Six experienced participants consider searching to be more efficient as compared to filtering. Comments from these participants include that it is "more intuitive to search for what you need" and that "searching for a specific type helps narrowing down" the displayed information. Further, it was again commented that "usually you are looking for something in particular and seldom you are aiming to exclude". One participant took the opposing view, that finding relevant information is more efficiently done by filtering. Inexperienced participants are mostly ambivalent.

Next, the participants were asked to compare the tool features with respect to which tool feature they would use when facing high complexity. Eight participants found searching and filtering equally useful when facing high complexity. One of them stated that "searching is useful to make a quick search and filtering is useful to look at details". Another commented that "for large sets both would be needed for easy overview". These comments may reveal the approach that an engineer performing requirements analysis would take, namely to learn about the system design first by searching for information, and then narrow down the set of candidate elements based on what was learned. In this process of narrowing down the set of candidate elements, we speculate that the engineer benefits from obtaining an overview. Indeed, for a small set of elements, searching may be enough to achieve an overview, whereas for large sets of elements, filtering is necessary.

Lastly, the participants were asked if their preference between tool features depends on if they were considering a requirements specification or a system model. Figure 7 shows how the participants answered. Possible answers are "searching", "mostly searching", "equal use", "mostly filtering" or "filtering". The two columns for each possible answer represent the answers for a requirements specification and a system model respectively. The colours of the bars show the answers that are given by participants that are experienced as requirements engineers and the participants who are not requirements engineers. It can be seen that experienced participants prefer searching over filtering for both requirements specifications and system models. In fact, only five participants made a difference between requirements specifications and system models. Comments from these five include that the choice of tool feature should be flexible "according to the situation". However, they chose searching for system models, because when considering them you "more often know what you want". From this, it can be seen that at least one engineer had a different expectation of system models as compared to requirements specifications. It could be that the structure of a system model can be experienced as more tangible as compared to the structure of requirements specifications. A well-made system model has a clear hierarchy in terms of components with interfaces and a structure that comes from realising functions. In the ideal case, the requirements should also have such a clear structure – something that might be helped by having a structured information model like EAST-ADL or by putting effort into maintaining traceability among requirements. Indeed, after the manual task of requirements analysis involving allocation, the structure of the requirements should be more clearly seen through the structure of the system model that the requirements become allocated to.

6 CONCLUSIONS

In this paper, we presented the requirements allocation assistant tool RAA, developed using a reusable design science methodology. RAA was developed to enable searching and filtering on requirements specifications and system models, with the purpose to make requirements analysis involving allocation efficient – even in the presence of high complexity. The way that this work addressed complexity was by allowing the displayed information to be restricted to a

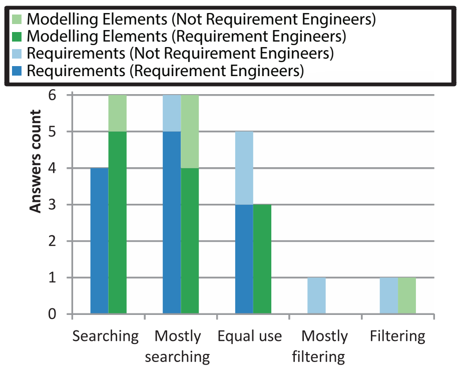


Figure 7: Tool feature preferences for requirements specifications and for system models.

level that can be managed manually, not by reducing the complexity of the system design. Further, the semantics of a structured information model like EAST-ADL may help to make a system model more understandable and better structured.

To evaluate RAA and learn how requirements analysis can be further supported, a qualitative usability study was conducted with 17 engineers. From this, we can form a preliminary theory of the typical process. The process starts by learning about the system model based on considering a given requirement, using the main tool feature. Subsequently, a set of elements for allocation is identified, preferably also by searching. For large information sets, filtering is also used. Finally, the requirements and the system elements are checked to assure that the allocation is correct. In the last two steps, multiple study participants requested further tool support.

This preliminary theory forms the basis for future work. We will include a wider selection of participants in the usability study. Additionally, we will extend the tool in order to suggest a possible set of candidate elements for allocation to a given requirement or to check the correctness of a given allocation.

ACKNOWLEDGEMENTS

We would like to thank the engineers who participated in the evaluation of the prototype. This work was supported by VINNOVA under the FFI programme. This work was conducted as part of the Synligare Project. This paper reports on results that were conducted as a Master Thesis project at Chalmers University of Technology, Sweden (Olaru, 2015).

REFERENCES

AUTOSAR (2015). AUTOSAR. http://www.autosar.org/.

Bassil, S. and Keller, R. (2001). Software visualization tools: survey and analysis. In *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*.

Blom, H., Lönn, H., Hagl, F., Papadopoulos, Y., Reiser, M.-O., Sjöstedt, C.-J., Chen, D.-J., Tagliabò, F., Torchiaro, S., Tucci, S., and Kolagari, R. T. (2013). EAST-ADL: An architecture description language for automotive software-intensive systems. In *Embedded Computing Systems*. IGI Global.

EAST-ADL Association (2015). EAST-ADL v2.1.12. http://www.east-adl.info/Specification/V2.1.12/html/index.html.

Faulkner, X. (2000). *Usability engineering*. Palgrave.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Q.*, 28(1):75–105.

Hofmeister, C., Nord, R., and Soni, D. (2005). Global analysis: moving from software requirements specification to structural views of the software architecture. *Software, IEE Proceedings -*, 152(4):187–197.

ISO (1998). ISO/IEC 10746-1:1998 Information technology – Open Distributed Processing – Reference model: Overview. *ISO/IEC 10746-1:1998*, pages 1–76.

ISO (2011). ISO 26262-1:2011 Road vehicles – Functional safety. *ISO 26262-1:2011*, pages 1–23.

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *Software, IEEE*, 12(6):42–50.

Liebel, G., Marko, N., Tichy, M., Leitner, A., and Hansson, J. (2014). Assessing the state-of-practice of model-based engineering in the embedded systems domain. In *Proc. of ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems*. Springer International Publishing.

Object Management Group (2015). SysML 1.3. http://www.omg.org/spec/SysML/1.3/.

Olaru, A. G. (2015). Visualizing relevant information during requirements allocation to system model elements. Master's thesis, Chalmers University of Technology, Sweden.

Passera, S. (2015). Beyond the wall of text: How information design can make contracts user-friendly. In *Design, User Experience, and Usability: Users and Interactions*. Springer International Publishing.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164.

Telea, A., Voinea, L., and Sassenburg, H. (2010). Visual tools for software architecture understanding: A stakeholder perspective. *Software, IEEE*, 27(6):46–53.