Employing Use-cases for Piecewise Evaluation of Requirements and Claims

Matthijs Westera
Utrecht University, AI
Matthijs.Westera@phil.uu.nl

Jimmy Boschloo
Utrecht University, AI
Jimmy.Boschloo@phil.uu.nl

Jurriaan van Diggelen
TNO Human Factors*
jurriaan.vandiggelen@tno.nl

Laurens S. Koelewijn
University of Groningen, HMC
lskoel@ai.rug.nl

Mark A. Neerincx
TNO* / Delft University of Technology
mark.neerincx@tno.nl

Nanja J.J.M. Smets
TNO Human Factors*
nanja.smets@tno.nl

* TNO Human Factors, P.O. Box 23, 3769 ZG, Soesterberg, The Netherlands.

ABSTRACT

Motivation – Complex design specifications must be partitioned in manageable pieces to be able to evaluate them in separate experiments. No methodology existed to deal with this task.

Research approach – Practical experience in Situated Cognitive Engineering and the Mission Execution Crew Assistant is combined with a theoretical perspective on the relation between use-cases, requirements and claims.

Findings/design – Hierarchical clustering is an effective method for partitioning a design specification. Use-cases provide a good criterion based on which to cluster the requirements and claims.

Originality/Value – A new method and tool are presented for organising requirements and for systematising the evaluation of a complex design specification.

Take away message – Piecewise evaluation benefits from a use-case-based partitioning of the design specification combined with an experimental stance on requirements and claims.

Keywords

Empirical requirements evaluation, Cognitive Engineering, use-case-based decomposition, requirements clustering, piecewise evaluation.

INTRODUCTION

Future manned missions to the Moon and Mars set high demands for personalised support systems taking into account the social, cognitive and affective states of the astronauts. A common way to address the complexity of developing such systems is to follow an iterative human-centered design process (see for an overview e.g. Norman, 1986; Rosson and Carroll, 2002). In this way, the requirements, claims and use-cases in the design specification are developed iteratively in successive stages of empirical evaluation and refinement.

Typically the design specification of complex cognitive support systems is of such a size that it cannot be evaluated as a whole in one experiment. Rather, the design specification must be divided into manageable pieces that can be evaluated in separate experiments. Partitioning the design specification for piecewise evaluation is a non-trivial task due to the many interdependencies between requirements, claims and use-cases. A systematic approach is required to ensure that the quality of the design specification is sufficiently assessed after its parts have been evaluated. The aim of this paper is to present our proposed solution to this problem.

The solution is based on two core ideas. First, experiment design in Cognitive Engineering should be driven by the formulation of *hypotheses* that target pieces of the design specification from an experimental stance (cf. Woods, 1998; Carroll and Rosson, 2003). Second, a suitable partitioning of the design specification for piecewise evaluation can be obtained by applying appropriate use-case-based selection criteria to the complete set of hypotheses.

To emphasise and illustrate the practical value of these ideas, we apply them to the Situated Cognitive Engineering approach in general and the Mission Execution Crew Assistant (MECA) project in particular (Neerincx and Lindenberg, 2008). MECA is an electronic personal assistant aimed to empower the cognitive capacities of human-machine teams during planetary and lunar exploration missions. We present a prototype tool to visualise the MECA design specification and to select hypotheses adequate for evaluation.

The paper is organised as follows. In the next section, we introduce Situated Cognitive Engineering and explain which pieces of a design specification may represent testable hypotheses. In the third section, we describe how requirements and claims can be clustered based on the use-cases to which they apply, yielding a use-case-based (as opposed to functional) decomposition of the design specification. In the fourth section, we explain why a use-case-based decomposition is a useful method for the exploration and selection of hypotheses for empirical evaluation.

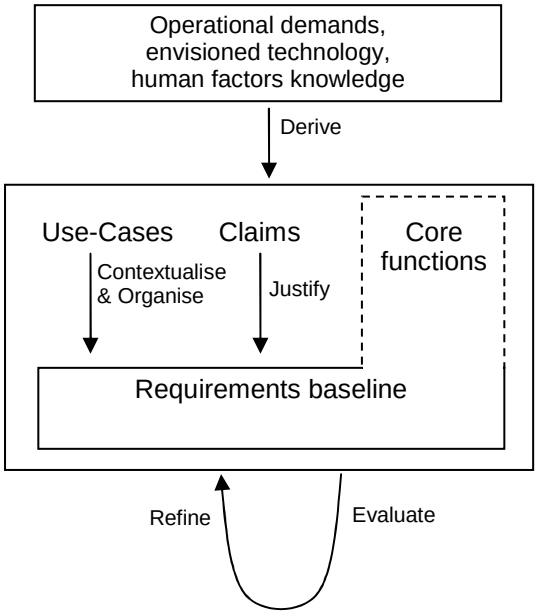

Figure 1. The basic organisation of the MECA design specification. Claims justify requirements, including the high-level core functions. Use-cases, aside from providing context, serve to organise the requirements baseline.

ORGANISING THE DESIGN SPECIFICATION

Following trends in Cognitive Engineering (e.g. Hollnagel and Woods, 1983; Norman, 1986; Rasmussen, 1986), and requirements engineering, we have applied our ideas to the Situated Cognitive Engineering method, as employed in the Mission Execution Crew Assistant project, formulated in (Neerincx and Lindenberg, 2008). Situated Cognitive Engineering centers on the notion that engineering and evaluating complex systems like MECA relies on situated rather than universal theories of cognition: theories which are embedded in the context of design, including operational demands, human factors knowledge and envisioned technology.

Requirements, Claims and Core Functions

Figure 1 summarises the way the design specification is organised. Based on operational demands, envisioned technology and human factors knowledge, a design specification is derived which is then iteratively evaluated and refined. The design specification consists of core functions and requirements that elaborate on these core functions. In addition, claims are included to justify features and design decisions, highlighting the upsides, downsides and trade-offs involved (Carroll and Rosson, 2003). A large set of use-cases contextualises the requirements, indicating in what kinds of situations a given requirement applies.

This approach differs from previous work on (Situated) Cognitive Engineering in three ways. First, as the dashed line in the figure indicates, the set of core functions is treated as part of the requirements baseline, rather than as a high-level division of functional domains (as in for instance functional decomposition). This ensures that dependencies between the different core functions and between the requirements that elaborate on them are not a-priori excluded.

Second, claims are included not only to justify the core functions (as in Neerincx and Lindenberg, 2008), but rather, in accordance with our stance on core functions as requirements, they are included to justify any individual requirement (and, optionally, sets of requirements).

Third, although use-cases still serve to contextualise the requirements (as in Carroll 2000), we show in the remainder of this paper that use-cases are essential for organising the requirements baseline and selecting useful hypotheses for empirical requirements evaluation.

Design Specification as a Collection of Hypotheses

In order to deal with the often troublesome connection between individual design problems and an overarching theory as well as to streamline the evaluation process, it has been proposed to regard (parts of) the design specification as a scientific hypothesis (Carroll and Rosson 2003, Woods 1998):

"An experimental stance means that designers need to recognise that design concepts represent hypotheses or beliefs about the relationship between technology and cognition/collaboration, [and] subject these beliefs to empirical jeopardy by a search for disconfirming and confirming evidence, […]. This experimental stance is needed, not because designers should mimic traditional research roles, but because it will make a difference in developing useful systems […]." (Woods 1998, pp.170-171)

Applied to Situated Cognitive Engineering (or requirements engineering in general), we propose that a design specification can be phrased as a hypothesis as follows:

(1) Any system adhering to the requirements baseline is optimised to help achieving the system's goal.

Since claims, as justifications, are always formulated in line with the overarching goal of the system, we can make the hypothesis in (1) more concrete:

(2) The claims are an adequate justification of the requirements baseline.

After all, if the claims are an adequate justification of the requirements baseline (2), then a system adhering to the requirements baseline will optimally help reach the goal (1); if the claims are not an adequate justification of the requirements baseline (2), then a system adhering to that requirements baseline may not help reach the goal (1). In this formulation the entire requirements baseline is covered, but similarly any subset of requirements with its corresponding claims may function as a hypothesis. This is in line with Rosson and Carroll's suggestion to treat claims as hypotheses (Rosson and Carroll 2008), but our formulation makes the different roles of requirements and claims more

explicit. Below we explain how regarding (parts of) the design specification as a hypothesis serves to streamline the evaluation and refinement loop.

Truthfulness and Exclusiveness

What does it mean for a set of claims to be an 'adequate' justification of some part of the requirements baseline? First, an adequate justification is *truthful*: all information that the justification is based on must be factual. Second, an adequate justification is *exclusive*: it must explain why the current and not some other requirements baseline is optimal. For a claim to be truthful, the upsides, downsides and trade-offs contained in it should occur as such in reality. If, for instance, a claim includes the upside "increases efficiency by at least 10%" whereas factually this is only 5%, the claim must be revised. A revision need not always lead to the requirement(s) becoming worthless. After all, a 5% increase in efficiency is still good, provided that no important downsides exist. However, if new facts cause the downsides to dominate the upsides, the inclusion of the requirement in the design specification is no longer justified and the requirement needs to be modified or removed.

For a claim to exclusively justify a requirement, it must hold that the claim cannot apply to any other requirement while maintaining its truthfulness. After all, if alternative requirements exist that lead to the exact same upsides and downsides and involve the same trade-offs, choosing any one of them over the others would be unjustifiable. If such a situation occurs, a generalisation of the various alternatives should take its place until further research reveals which of its instantiations is the best candidate. The exclusiveness of claims thus ensures that preliminary convergence in the requirements baseline can be detected and blocked. A consequence is that initial requirements are typically general, as well as the claims that justify them. *Refinement* can only iteratively proceed from general to specific, carefully justifying at each step the refinement made.

Although treating the refinement process in Situated Cognitive Engineering in detail is beyond the scope of this paper, a short remark is in place here because the refinement of requirements can be (and should be) coupled to evaluation (they both label the same arrow in Figure 1, after all). To see this, consider that the exclusivity of claims requires an experiment to always compare various alternatives. Although in principle such alternatives could be invented especially for such an experiment, it is very hard for the researcher not to be biased in favor of the current requirements. This bias can be avoided by, instead of comparing the current requirements to a set of alternatives, testing various candidate refinements *before* the actual refinement takes place to determine which of all possible refinements would yield the best result. This method avoids the bias because the current requirement is not being questioned (although, of course, it was being questioned at an earlier stage in the evaluation and refinement cycle).

Requirement RF2024	MECA shall communicate with the crew about important events.	
Claim C064	Each MECA unit will alert the crew member regarding for instance scheduled events, low-frequency nominal events and off-nominal events.	
	+	Helps maintain high situation awareness for crew members. Consistent notifications help maintain sufficient trust.
	-	May interrupt with current activities, increasing cognitive task load, and hence decreasing effectiveness or efficiency.
Use-Cases	UC077, UC078, UC080, UC083	

Table 1. An example building block (simplified) of the MECA design specification.

The Mission Execution Crew Assistant

A first design specification for MECA was constructed based on operational demands, human factors knowledge and an envisioned technology. It was then iteratively evaluated and refined using several methods. The current MECA design specification consists of 167 requirements that elaborate on the 6 core functions (viewed as high-level requirements themselves), a set of claims that justify the inclusion of requirements in the baseline and a collection of about 80 use-cases. The core functions are health management, diagnosis, prognosis and prediction, collaboration, resource management, planning, and sense-making. Seven criteria have been derived from human factors knowledge and are referred to in the claims: to increase effectiveness, efficiency and situation awareness, to maintain appropriate trust levels, high learnability and high satisfaction and to incorporate emotional responses. Table 1 contains an example of a MECA requirement with its claim (simplified) and use-cases. As we mentioned, sets of multiple requirements rather than single ones could likewise be accompanied by a claim and use-cases, for instance to highlight dependencies between the requirements.

Following the ideas discussed above, requirement RF2024 with its claim C064, included in Table 1, could be used together with other requirements and claims as a hypothesis. The requirement is possibly too general to be cast in any doubt: of course a personal assistant should communicate with the crew about really important events, for it is the only mean through which they could possibly learn about such events. Its claim with upsides and downsides is just as general, and rather than as a rock-hard justification it should be seen as a rough guideline regarding which aspects to pay attention to when formulating refinements. Its possible refinements, on the other hand, are numerous and not all

Use-case UC083	Alarm handling
Goal	To bundle low-level alarms into meaningful events.
Actors	Astronaut in habitat, MECA of habitat.
…	
Requirements	RF2024, RF2050, RF2080, RF2260, RU3010, RU3011, RU3012, RU4040, RU4120, RU4130, RU4140, RU4170, RU4260, RF2230, RU4200.
Scenario	1. Low-level smoke and IR alarms are activated. 2. MECA combines the low-level alarms to determine the location and scope of the fire. 3. MECA gets the attention of the astronaut 4. MECA provides procedures to the astronaut to fight the fire 5. Astronaut fights fire 6. Fire is put out.

Table 2. An example use-case of the MECA system. Fields like preconditions, post-conditions and comments have been omitted for brevity.

trivially valid or invalid. For instance, should MECA communicate about important events verbally or visually? Should MECA consider postponing certain messages depending on the current cognitive task load? When is an event to be classified as important enough for interrupting the current task? A literature study could narrow down the set of alternatives sufficiently for them to be compared in an empirical evaluation experiment.

USE-CASE-BASED DECOMPOSITION OF THE REQUIREMENTS BASELINE

The design specification contains a large number of use-cases. Use-cases in the MECA project have been crafted along the lines of (Cockburn 2001). They make explicit the various contexts of use, varying from simple interactions with only one crew member to complex sequences of events (often referred to as scenarios rather than use-cases). Table 2 contains an example of a use-case, containing descriptions of the goal and the actors, a set of relevant requirements, and a step-by-step description of the event.

Use-cases are central to engineering complex systems. They allow multiple views and levels of detail, help achieve abstraction and categorisation, are concrete and flexible at the same time, promote work-orientation and invoke reflection (Carroll 2000). In this section we show that an important role for use-cases, not treated by Carroll, is to organise the requirements baseline.

The Requirements Dendrogram

We propose that a key role for use-cases in cognitive engineering is to organise the requirements baseline through *use-case-based decomposition*. By indicating for each requirement the use-cases to which it applies, a

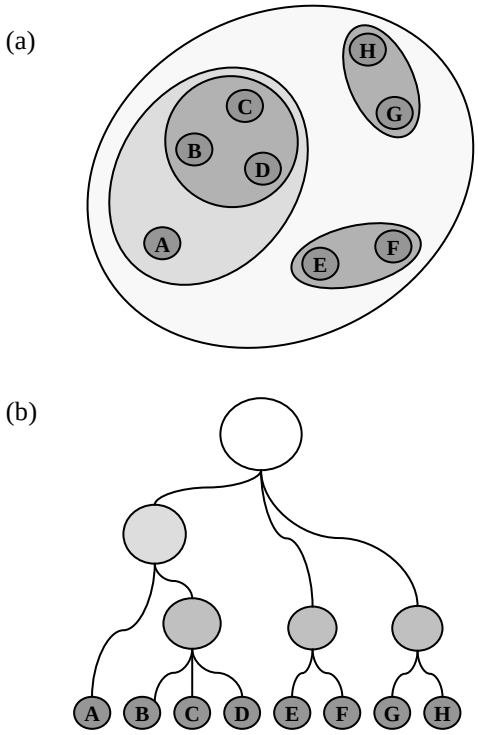

Figure 2. Eight requirements labeled A to H are clustered at several levels of within-cluster similarity. (a) Here, for illustrative purposes, the similarity between requirements corresponds to their distance. (b) The same hierarchical clustering can be depicted in a dendrogram.

measure of similarity between requirements can be computed. The requirements can then be input to a clustering algorithm, or in particular a *hierarchical clustering algorithm* (Johnson 1967), to obtain a hierarchy of clusters of similar requirements. These clusterings range from a single large cluster containing all requirements (low within-cluster similarity) down to many small clusters containing only one requirement each (high within-cluster similarity), as in Figure 2a. Such a hierarchy of clusterings can be visualised as a *dendrogram*, illustrated in Figure 2b.

Use-case-based decomposition is similar to the more traditional way of organising a design specification through *functional decomposition* (see for instance the 'structured analysis' method, described by for instance DeMarco, 1979; Yourdon, 1989), but offers a number of advantages over the latter.

First, the stativity and rigidity of a top-down functional decomposition does not fit the fluidity of the design situation – a problem that the usage-centered cognitive engineering was meant to resolve in the first place. Use-case-based decomposition, on the other hand, is dynamic and flexible, with the hierarchy automatically changing as requirements or use-cases are added or removed. It should be mentioned here that the more requirements and use-cases a design specification

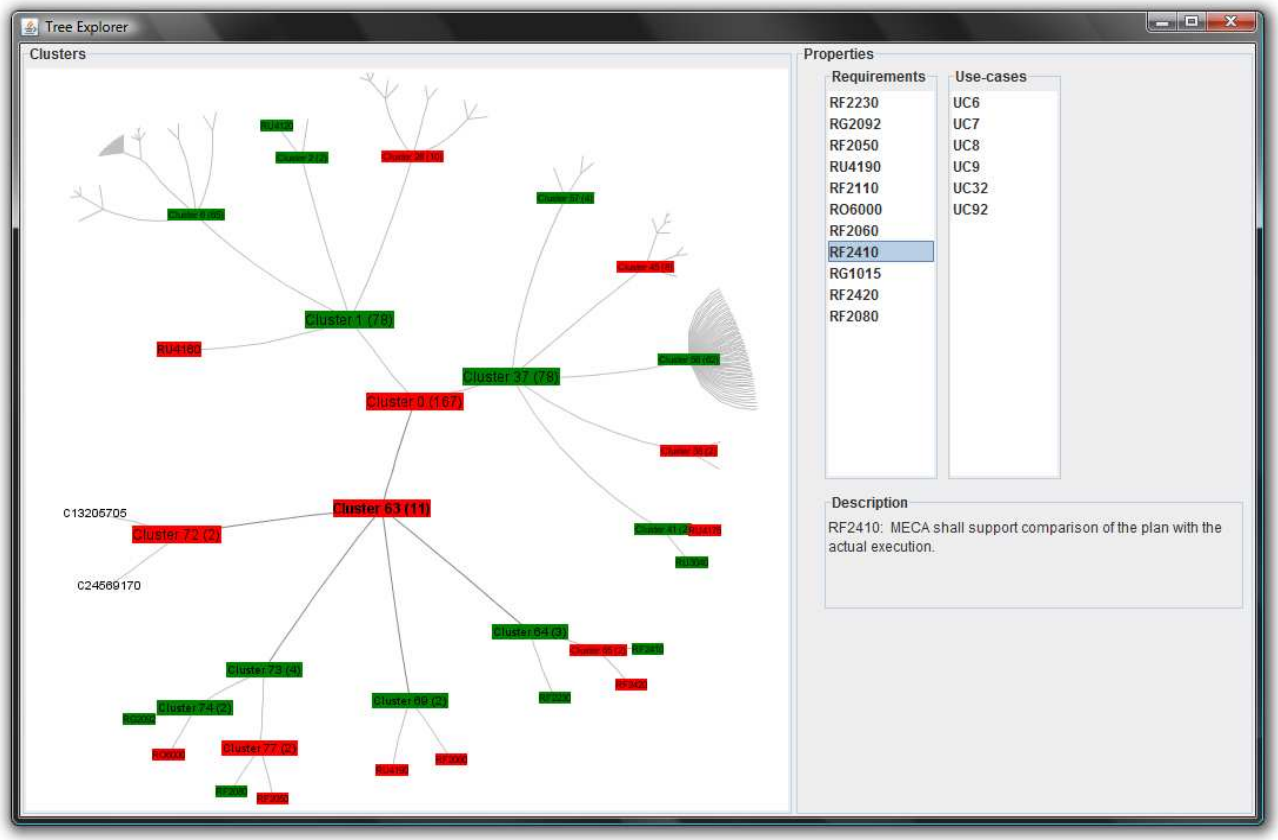

Figure 3. A screenshot of the hypothesis exploration tool (prototype).

contains, the more stable the use-case-based decomposition becomes.

Second, an advantage of use-case-based decomposition over functional decomposition is that it closely reflects the pragmatic experience of the user. The clusters will contain those combinations of requirements and the dependencies between them that are relevant for the user and which will provide a useful hypothesis for user-centered empirical evaluation.

Third, in the more traditional approach, a mismatch between the designers' envisioned decomposition and a functional hierarchy, or a disagreement between designers on the desired location of a requirement in the hierarchy, will block creativity and discussion. By employing use-case-based decomposition, on the other hand, such a mismatch will instead highlight which use-cases may be missing or inadequate. It will lead to a fruitful re-evaluation of the use-cases.

A Requirements Dendrogram for the Mission Execution Crew Assistant

We decomposed the MECA requirements baseline based on approximately 80 use-cases. We employed the conceptual clustering algorithm COBWEB (Fisher, 1987), as implemented in the *WEKA* workbench (http://sourceforge.net/projects/weka/; Witten and Frank, 2005), with the *acuity* parameter set to the default value and *cutoff* set to 0.09. In a nutshell, COBWEB constructs a hierarchy of clusterings by maximising the over-all *category-utility*, a measure based on conditional probabilities.

In our clustering runs so far, the requirements' applicability to use-cases has always been a binary attribute, to increase the efficiency of the manual use-case annotation process. With applicability as a binary attribute, we found that use-cases should be coupled to requirements rather liberally for the resulting dendrogram to be insightful and balanced. We suspect that using a continuous attribute instead could further improve the resulting clustering.

For the MECA project we have developed a prototype tool to explore and select hypotheses for empirical evaluation (Figure 3). Central to this tool is the requirements dendrogram: the hierarchy of clusters constructed bottom-up from the use-cases. The *HyperGraph* toolkit is used to visualise the requirements dendrogram (http://hypergraph.sourceforge.net/). A hyperbolic (i.e. fish-eye) view is a common tool for visualising large hierarchies, because it provides an intuitive way to focus on parts of the hierarchy while at the same time maintaining the overview. Selecting a cluster in the tree will reveal on the right the requirements contained in it and the use-cases to which they apply.

The MECA design specification is organised according to an OWL/RDF ontology (Breebaart *et al.* 2009), accessible to all parties involved through a web-

interface. The ontology defines concepts like 'crew member' and 'equipment', as well as important meta-concepts like 'requirement', 'use-case' and 'claim'. We regard the tool presented above as an extension, coupling a knowledge base with a dynamic, use-case-based visualisation of the requirements baseline.

SELECTING A HYPOTHESIS FOR EMPIRICAL REQUIREMENTS EVALUATION

So far we have mainly described how a design specification is best organised. Central are the ideas to view sets of requirements, with the corresponding justifying claims, as hypotheses, and to decompose the requirements baseline based on use-cases. In the current section we show how both ideas are combined in a proposed methodology for selecting hypotheses with high utility for empirical evaluation.

Criteria for a Useful Hypothesis

When is a design specification finished? Although an empirical hypothesis can never be undisputedly proven (because a counterexample may always be discovered), hypotheses concerning parts of the design specification can be made more plausible through experimentation. Determining when some part of the design specification is finished amounts to estimating the *plausibility* of the corresponding hypothesis. There comes a point at which to accept it as true and enter the implementation stage (for pointers regarding this decision, see e.g. Zave and Jackson, 1997).

Before that happens, it is useful to keep track of which pieces of the design specification have been tested together and whether the outcome confirmed the hypothesis, in order to avoid testing an already quite plausible hypothesis. With such bookkeeping, the plausibility of a hypothesis can be computed automatically. Looking again at the tool in Figure 3, nodes in the hyperbolic tree, representing clusters or hypotheses, are coloured red (not plausible yet) or green (plausible), based on their estimated plausibility.

Testability and Empirical Value

Not only do hypotheses differ in their established plausibility, they also differ in their *testability* (whether the hypothesis is easy to test) and *empirical value* (whether testing the hypothesis has any influence on the plausibility of the design specification as a whole). A requirements baseline can be decomposed into exponentially many different chunks, each with a corresponding hypothesis. Determining which of these countless hypotheses are suitable for an evaluation experiment is a difficult problem.

Two rules of thumb can help to optimise testability and empirical value. First, the larger the *number* of requirements in a hypothesis, the more closely it resembles the design specification as a whole and hence the higher its empirical value, but the lower its testability. A hypothesis that concerns the entire requirements baseline carries maximal empirical value, but it is also the hardest to test – after all, one has to implement or simulate the entire system.

Second, *related* requirements are often subject to various interdependencies, the inclusion of which in a hypothesis greatly increases its empirical value. Relatedness also increases a hypothesis' testability, because related requirements apply in similar situations and can be tested under similar experimental conditions. This is especially the case when relatedness is estimated based on the use-cases in which requirements occur, rather than for instance on functional grounds.

Both rules of thumb are present already in the dendrogram resulting from a use-case-based decomposition of the requirements baseline (previous section). The number of requirements is incorporated roughly in the vertical dimension in the hierarchy, with the sets higher up in the dendrogram containing more requirements than the sets at the bottom. The relatedness between requirements is responsible for grouping some requirements together while keeping others separated and is fundamental to the dendrogram. A use-case-based decomposition of the requirements baseline thus allows the designer to focus on parts of the design specification that, as hypotheses, will optimise empirical value and testability.

A Simple Procedure for Selecting Hypotheses

Given a requirements dendrogram, possibly incorporated in a tool like the prototype presented above, the methodology for selecting an optimal hypothesis can be sketched as follows:

1. Select an underevaluated hypothesis (i.e. a red cluster in the dendrogram in Figure 3).
2. If the hypothesis is very difficult to test given the resources available (money, time, participants, software), move down in the dendrogram to its sub-hypotheses until an underevaluated hypothesis is encountered that is testable.
3. If the hypothesis is very easy to test given the resources available, consider moving up in the dendrogram until a maximally challenging hypothesis is encountered.

The testability of hypotheses higher up in the dendrogram increases gradually as each of its sub-hypotheses are made plausible, because only the interaction of its sub-hypotheses remains to be investigated. This enables the designer to gradually test larger and larger parts of the requirements baseline without introducing too many variables at once.

CONCLUSION

Partitioning a large design specification for piecewise evaluation can be difficult due to the many interdependencies between requirements, claims and use-cases. In this paper, we have proposed a method to improve systematicity of piecewise evaluation in Situated Cognitive Engineering in two ways. First, each set of requirements with a set of justifying claims is regarded as a hypothesis concerning the adequacy of the justification. Such hypotheses should be tested by comparing different refinements of the requirements

baseline. Second, whether a hypothesis is worth testing depends on the number of requirements, the relatedness of its requirements (which can be estimated based on use-cases), and its established plausibility. Applying the method yields a use-case-based decomposition of the design specification. We have developed a tool to support the method.

We believe the proposed method is an asset for the development of a wide range of cognitive systems, such as naval ships design (Neerincx, *et al.*, 2008), patient self-care supervision (Blanson Henkemans, *et al.*, 2007), negotiation support (Hindriks and Jonker, 2008) and Mars surface operations (Clancey, Lee and Sierhuis, 2001).

We have identified a number of interesting directions for future work. We intend to investigate different ways to annotate requirements for clustering (other than use-case-based). We have gained some initial experience with testability-based decomposition by annotating all requirements with the experimental conditions under which they could be evaluated (in a virtual or real environment, with human participants or artificial agents, with or without a measure for cognitive task load, etc.). We believe that such criteria can provide a valuable addition to our method. Another way to improve the requirements clustering method could be to enrich the annotations of use-cases and requirements by referring to a dedicated ontology and use these annotations as a basis for clustering. Natural language processing methods such as Latent Semantic Analysis could be employed to automate part of the annotation process.

Another direction for future research is to compare functional decomposition with use-case-based decomposition. We have already explained that a use-case-based decomposition approach better fits the dynamicity of the design context, helps to refine use-cases and reveals important interdependencies that would otherwise have gone unnoticed. Another important difference concerns the lack of cluster labels in use-case-based decomposition, which are usually defined from the start in functional decomposition. Two questions could address this difference. First, would meaningful cluster labels increase insight in the hierarchy and, for instance, promote communication? Second, if desired, could such labels be derived automatically from the data? So far, automatically labeling clusters based on the descriptions of use-cases to which they apply has yielded promising results.

Finally, an important topic for future work is the link between evaluation and refinement and the role of a use-case-based decomposition therein. Use-case-based decomposition results in a hierarchy that from top to bottom roughly reflects the routes of refinement, leading from general to increasingly specific requirements. Clusters higher up in the hierarchy not only contain more general requirements, in a way they can also be *identified* with such high-level requirements. For that reason we expect that the proposed method could be useful for identifying areas that need refinement and areas that have been prematurely refined.

ACKNOWLEDGEMENTS

MECA is a development funded by the European Space Agency (contract numbers 19149/05/NL/JA and 21947/08/NL/ST). Project partners are TNO Human Factors (NL), Science & Technology BV (NL), OK-Systems (E) and EADS-Astrium (D). Website can be found at http://www.crewassistant.com.

REFERENCES

Blanson Henkemans, O.A., Bonacina, S., Cappiello, N., Van der Mast, C., Neerincx, M.A., Pinciroli, F. (2007). A hybrid multi-agent system architecture for distributed supervision of chronic patients in the e-health setting. In *Euromedia 2007*, pp. 119-124, Eurosis-ETI, Ghent.

Breebaart, L., Bos, A., Grant, T., Olmedo Soler, A., Neerincx, M.A., Smets, N.J.J.M., *et al*. (2009). The MECA Project - Ontology-Based Data Portability for Space Missions, in *Space Mission Challenges for Information Technology*, pp. 193-200.

Carroll, J.M. (2000). Five reasons for scenario-based design. *Interacting with Computers* 13.1: pp. 43-60. Elsevier Science B.V.

Carroll, J.M., Rosson, M.B. (2003). Design Rationale as Theory. In Carroll, J.M. (ed.), *HCI Models, Theories and Frameworks*, pp. 431-461, Elsevier Science, San Fransisco.

Clancey, W. J., Lee, P., Sierhuis, M. (2001). Empirical Requirements Analysis for Mars Surface Operations Using the Flashline Mars Arctic Research Station, in *Proceedings of the Fourteenth International FLAIRS Conference*, AAAI Press, pp. 24-26.

Cockburn, A. (2001). *Writing Effective Use Cases*. Addison-Wesley.

DeMarco, T. (1979). Structured Analysis and System Specification. Prentice Hall.

Fisher, D.H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning* 2: pp.139–172.

Hindriks, K.V., Jonker, C.M., Creating Human-Machine Synergy in Negotiation Support Systems: Towards the Pocket Negotiator (2008). In W.-P. Brinkman (ed.), *Proc. of the 1st Int. Working Conference on Human Factors and Computational Models in Negotiation*, pp. 47-54, ACM, New York.

Hollnagel, E., and Woods, D.D. (1983). Cognitive systems engineering: New wine in new bottles. *International Journal of Man-Machine Studies*, 18, 583-600.

Johnson, S.C. (1967). Hierarchical Clustering Schemes. *Psychometrika* **2,** pp. 241-254.

Kelley, J.F. (1983). An empirical methodology for writing user-friendly natural language computer applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp.193-196, Boston, Ma.

Lamping, J., Rao, R., and Pirolli, P. (1995). A focus+context technique based on hyperbolic geometry for visualising large hierarchies. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp.401-408, Colorado, United States.

Neerincx, M.A. (2008). The Mission Execution Crew Assistant: Improving Human-Machine Team Resilience for Long Duration Missions. IAC2008.

Neerincx, M.A. & Lindenberg, J. (2008). Situated cognitive engineering for complex task environments. In: Schraagen, J.M.C., Militello, L., Ormerod, T., & Lipshitz, R. (eds.), *Naturalistic Decision Making and Macrocognition*, pp. 373-390. Aldershot, UK: Ashgate Publishing Limited.

Neerincx, M.A., Te Brake, G.M., Van de Ven, J.G.M., Arciszewski, H.F.R., De Greef, T.E., and Lindenberg, J. (2008). Situated cognitive engineering: Developing adaptive track handling support for naval command and control centers. In B. Patrick, C. Gilles and L. Philippe (eds.), *Third Internation Conference on Human-Centered Processes*, pp. 3-20. TELECOM Bretagne, France.

Norman, D.A. (1986). Cognitive Engineering. In D.A. Norman, and S.W. Draper (eds.), *User-Centered System Design: New perspectives on human-computer interaction*, pp. 31-62. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rauterberg, M., Neerincx, M.A., Tuyls, K., Loon, J. van (2008). Entertainment computing in the orbit. In: Ciancarini, P., Nakatsu, R., Rauterberg, M., Roccetti, M. (Eds.). *IFIP International Federation for Information Processing* **279**:pp. 59–70. Boston: Springer.

Rasmussen, J. (1986). *Information processing and human-machine interaction: an approach to cognitive engineering*. Amsterdam, Elsevier.

Rosson, M. B., Carroll, J. M., 2002. *Usability Engineering, scenario-based development of human-computer interaction*. Academic Press, San Diego.

Rosson, M. B., Carroll, J. M., 2008. Scenario-based Design. In Sears, A., Jack, J. A. (eds.), *The Human Computer Interaction Handbook*, pp. 1041-1060, Taylor & Francis Group, New York.

Smets, N.J.J.M., Abbing, M.S., Neerincx, M.A., Lindenberg, J, and Oostendorp, H. van (2008). Gamebased evaluation of personalized support for astronauts in long duration missions. *Acta Astronautica* **66** (5-6), pp. 810-820.

Witten, I.A., Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco.

Woods, David D. (1998), Designs are Hypotheses about How Artifacts Shape Cognition and Collaboration. *Ergonomics*, 41, 168—173.

Yourdon, E. (1989). *Modern Structured Analysis*, Yourdon Press Computing Series.

Zave, P., and Jackson, M. (1997). Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology*, Vol. 6, No. 1, pp.1-30.