An Integrated Decision Model For Efficient Requirement Traceability In SPICE Compliant Development

Bernhard Turban[1], Markus Kucera[2],
Athanassios Tsakpinis[2], Christian Wolff[3]

[1]Electronic Systems Engineering,
Micron Electronic Devices AG, Neutraubling, Germany
bernhard.turban@micron-ag.com

[2] Competence Center SE,
University of Applied Sciences, Regensburg, Germany
{markus.kucera,athanassios.tsakpinis}@informatik.fh-regensburg.de

[3] Institute for Media, Information and Cultural Studies
Regensburg University Germany
christian.wolff@sprachlit.uni-regensburg.de

Abstract *— Requirement traceability ensures that (SW-)products meet their requirements and additionally makes the estimation of the consequences of requirement changes possible. It is especially difficult to establish at the transition from requirements specification to its provision in the design, because design processes represent creative and complex transfers of mostly unique problem constellations into a sustainable solution (so-called Wicked Problems). At first, this article searches for symptoms of the problem in analyzing the process model of ISO 12207, the foundation of SPICE or CMMi. This analysis mainly serves the derivation of a concept for the integrated extension of today's traceability models with the aspect of documented design decisions. In the context of current approaches in Rationale Management, our concept proofs as sustainable solution that supports „heavyweight" prescriptive approaches as well as „lightweight" pragmatic approaches and – moreover – shows interdependencies between both kinds.*

1 Introduction

In the development of safety-critical embedded real-time systems, aspects of safety and reliability have critical importance (cf. [1], ISO 61508). Due to this reason, issues as control and improvement of software processes (cf. ISO 15504 SPICE) are of high significance. In the course of establishing SPICE at *Micron Electronic Devices AG*, we have discovered several interesting results in this area. This paper presents parts of our results concerning requirement traceability from requirements specifications to system as well as software designs. Currently, these results are being implemented in a SE tool prototype.

For the development of safety critical embedded systems, traceable and consistent elaboration of requirements throughout all development cycles (especially the design phases) are mandatory. However, today's document-heavy approaches face problems with redundancy and synchronization of different stakeholders' views. To handle these issues, our approach concentrates on maintaining one consistent view of all requirements between all stakeholders.

In the following design phases, the involved stakeholders and artefacts - of the different engineering disciplines (Systems engineering, HW, SW) - shall be connected by a light-weight model and tool based approach.

The core of this approach is a decision-model, linking requirements, design problems and design together. As a result, new constraints on the solution space can be identified and used similar to requirements. Whereas former traceability approaches regarded decisions as valuable side information, our decisions get directly integrated in the classical traceability information forming traceability chains of decisions through the design processes. As a side effect, the approach addresses several problems in rationale management and encourages direct communication between the stakeholders.

We start with a short description of the state of the art in *traceability* (Chapter 2). Chapter 3 –a kind of insertion- introduces the *layered process model of SPICE* and shows connections between the process model and traceability issues. This builds ground for Chapter 4 which introduces our integrated decision model that helps to improve currently used traceability models. Chapter 5 shows connections between our decision model and previous approaches in the area of *Rationale Management*. In Chapter 6, the effects of such a traceability model are discussed. At the end, Chapter 7 shows an example which contrasts our approach to currently used practices. For better understanding of the first theoretical chapters, it is recommended to first take a short look at Chapter 7.

2 State-of-the-Art in requirement traceability

Requirements management, i.e. the activity of organizing, administrating and supervising requirements during the whole development process, and *Traceability* are mandatory actions to fulfil exigencies imposed by Software Engineering standards like SPICE[1] (*Software Process Improvement and Capability dEtermination*, cf. [2]) or CMMi (*Capability Maturity Model Integration*, cf. [3]).

Traceability means "comprehensible documentation of requirements, decisions and their interdependencies to all produced information/artefacts from project start to project end" ([4; p.407]). Between artefacts or respectively models of different development processes emerging structural interruptions – semantic gaps ([5], [6], [7; p.138f]) – endanger a project's consistency and the common understanding of its stakeholders. Traceability relationships are intended to close these gaps. Paech et al. [8] indicate that traceability in relation to the design of artefacts is typically seen as a set of bidirectional relationships between requirements and their *fulfilling* design entities (cf. [9]). Figure 1 schematically shows today's usual method of relating requirements to design elements by <<Satisfy>>-links[2] (cf. [10], [11]).

1 Furthermore the article concentrates on SPICE, but the claims are equally valid for CMMi, since both process models are based on the process model of ISO 12207.

2 Lately, the trend arises to use SysML ([12]) for these relations.

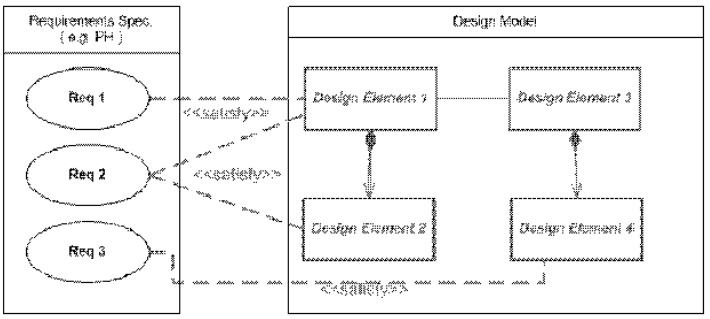

Figure 1: Today's practice of establishing traceability via <<Satisfy>>-links

Paech et al. [8] show that these relationships can be of a far more complex nature (cf. Fig.2). By restraining the solution space, *non-functional requirements (NFR)* restrain *functional requirements (FR)* and *architectural decisions (AD)*. On the other hand, NFRs are realised by FRs and ADs, whereas FRs are realized and restrained by ADs.

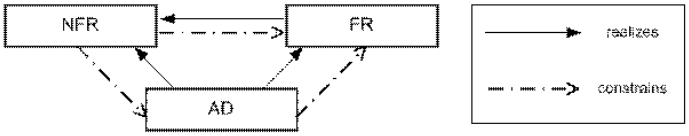

Figure 2: Relationships between non-functional (NFR), functional requirements (FR) and achitectural decisions (AD) according to [8].

3 The process artefact model of SPICE

The SPICE process model uses the standardised process model of ISO 12207 (cf. [2]). In the following, we focus on the processes ENG.2-ENG.5 of SPICE. These processes form a layer model of different levels of abstraction in which problem space descriptions (requirement view: ENG.2, ENG.4) alternate with solution space descriptions (designs: ENG.3, ENG.5) (cf. [13; p.113f], [14]):

- ENG.2: derives from the user requirements specification a general system requirements specification (SYS-RS).
- ENG.3: uses the SYS-RS to create a high-level systems design with the prior emphasis on HW-SW-partitioning.
- ENG.4: the software requirements specification (SW-RS) derives from ENG.2 and ENG.3.
- ENG.5: uses the SW-RS for the design of the SW architecture.

SPICE-oriented traceability models require a continuous link chain between the artefacts of ENG.2, ENG.3, ENG.4, ENG.5 to ensure the consistency of the entire model (cf. [15], [6]).

As our research has shown, this clear separation is a more or less metaphoric one (cf. [16]) providing orientation aid for the developers as process models do. However, in practical terms, such a clear separation is mostly not viable ([2; p.103], [13; p.114]).

Especially the proclaimed *specification of SW requirements* should be dealt with cautiously in our application domain, since a really separate SW-specification[3] faces the following problems:

- Often, requirements on HW and SW are strongly interwoven (cf. [2; p.104]).
- In many cases, SW functionality is already clearly demanded in the user requirements specification (ENG.1). Thus, if applying such a clear separation, those requirements must be taken over into the SYS_RS (ENG.2) and SW_RS (ENG.4) causing additional efforts and redundancies (cf. [18]).
- Other requirement types exist not attributable to either HW or SW (e.g. project management, quality management, mechanical construction). Alternatively, in current requirements management tools like DOORS®, a HW-SW-partitioning of requirements is also viable using an attribute (proposed values: System, HW, SW, construction, management).
- Generally, linking of different artefacts is a time consuming, unproductive and error-prone administrative work that should be minimized.

Thus, concentrating on more pragmatic views of the agility scene (cf. [19]), we propose to merely concentrate on *one* dependable, consistent artefact (cf. DRY-principle (*don't repeat yourself*) in [18]) to store all contractually obligatory requirements as one common view / interface to synchronize the views of all stakeholders in the project.

At least two dedicated exceptions exist that should be dealt with on their own and will not be part of the further discussion below:

- *Complex Systems (System of Systems)*: If complex systems can be divided into relatively independent subsystems (with exactly definable interfaces), then the subsystem specifications should be separated.
- For development parts delegated to subcontractors the interface and context of these must be deeply analysed and defined.

4 Relationship between layered process model and traceability

This chapter is concerned with the core of above described process layer model. In refining the previously described metaphoric idea of the process layer model, we come to the proposal of our approach.

The actual core of this approach is the following: System design has high impact on its SW design by raising new "requirements" in addition to the pristine requirements of the stakeholders (e.g. in the automotive sector, SW-design must be subordinated under constraints of extremely cost-optimized HW components. At the moment, SPICE neglects these critical connections between HW and SW).

However, one issue in SW requirements which might benefit from more intensive discussion is their negotiability. "Real requirements" are forming the contractual basis between the stakeholders -particularly with the customer. Occurring changes must be harmonized with the customer via a *Change Control Board* (CCB). Whereas, for "require-

3 Boehm points out, that the separation between Systems and SW engineering has been a historical and artificial one ([17]).

ments" to change with the origin of the definitions of the design, it is possible to search for an project internal solution first, before escalating the issue to a CCB is considered.

Thus, both kinds of requirements should be strictly separated in their notion. We use the following taxonomy (Fig. 3):

- *Requirements* are directly allocated to the SYS-RS, since they concern the legal agreement between customer and contractor.
- „Requirements" derived from requirements or designs are called *DesignContraints.*
- *Requirements* and *DesignConstraints* have similar qualities and structure. Thus, we use the term *RequirementalItem (RI)* for both items.

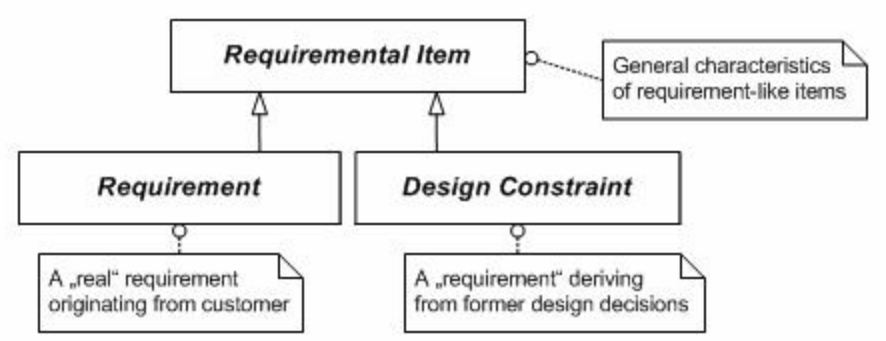

Figure 3: Requiremental items taxonomy

Requirements have to refer to their origin (cf. [7], [4]). This relation should apply to all RIs. The origin of *DesignConstraints* lies in previously made design decisions solving the conflicts/forces between RIs and/or architectural items constraining the broader more abstract solution space to a more concrete one.

These considerations suggest the inclusion of a decision model in the traceability information (cf. Fig.4[4]) helping to document the origin of new *DesignConstraints* (this especially helps to make the HW's influence on SW more transparent (cf. [20; p.415])) in a lightweight and need-oriented way.

The conventional scheme of relating requirements to realizing model elements is extended by a dialog allowing the capture of documented decisions. In this dialog, elements of the requirement model and the design model which are conflicting/ causing a problem can be chosen. Equally, diagrams describing aspects of the conflicting situation shall be attached as additional information (<<*documenting diagrams*>>).

Furthermore the decision can be specified on demand via a text component. The text component accepts unstructured text, but -when needed- can give adequate templates to support the decision documentation. A possible way to structure -the user should choose

4 Fig.3 shows this concept extending today's traceability models by an explicit decision model. The diagram sketches a concrete situation, where a conflict between two requirements (Req_1, Req_2) and two UML model elements (Class1, Class2) is resolved by a design decision resulting in two new DesignContraints (DesConstraint1, DesConstraint2).

these freely- is given in Fig.4 with the decision's attributes *assumptions*, *rationales* and *solution specification*.

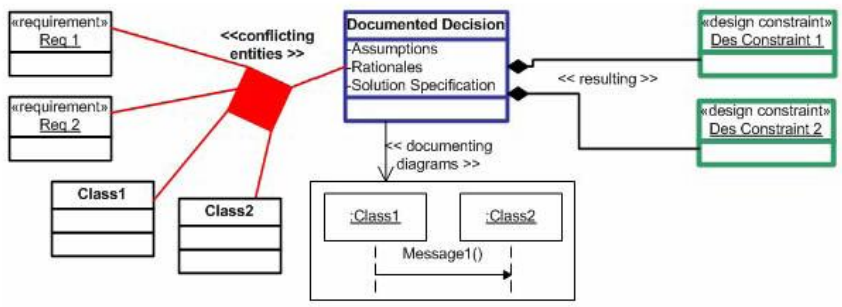

Figure 4: Documented decisions build the connection between requirements, design elements and resulting design constraints.

5 Relation to previous approaches of rationale management

The decision model presented here is strongly connected to the research area called *rationale management* (RM), since both deal with decisions during SE processes. In classical RM, the focus lies on documenting, recovering, further usage and reuse of justifications (= *rationale*) behind design decisions. RM mainly targets on the information about the "Why" of design decisions in order to alleviate the knowledge transfer of decision makers to other involved stakeholders.

However, existing approaches could not succeed in practice [22], even though documenting design decisions is regularly called for in literature (cf. IEEE 1471:2000, [23], [24], [25], [26], [27], [28]) and practitioners acknowledge the importance of this type of documentation ([29]). Diverse causes for this negligence have been identified, but the problem of *capturing* the rationale seems to be the main obstacle (cf. [22], [30]):

1. Most approaches are highly *intrusive* (bothersome and interfering) to the design process with extra effort for capturing ([31], [30]).
2. The approaches tend to have negative impact on the decision process, since not all (aspects of) decisions can be rationally justified, but arise from intuitive considerations (Fischer and Schön's „Theory of Reflective Practice" [32], [22]) basing on diffuse experiences (tacit knowledge [33]) (cf. [22], [30], [34]).
3. Decisions must be made despite of unclear circumstances and it is impossible to include all relevant information (*bounded rationality* [30]). Thus satisfactory solutions must be found although problem knowledge is clearly limited ([35]).
4. *Grudins Principle* [31] suggests that collaborative systems fail, if the invested value is not returned to the information bearers ([22], [36]).

The problem mentioned in Point 1 implies that in any case *not all* decisions can be treated exhaustively. For example, Clement, Bass et al. only refer to the documentation of the *most important* decisions ([23], [24]). Booch ([28]) gives another lead by dividing decisions into *strategic* (i.e. with striking impact on architecture, mostly made on the early stage of a project) and *tactical* (i.e. locally limited impact on the architecture).

In this context, strategic decisions must/should be thought through carefully and should – if possible – be made on explicit rationale grounding. For this relatively small fraction, the investment in more intensive analyses is highly valuable, as discussed by most approaches on rational management ([37], [23], [24], [38]). These issues may be analyzed in a prescriptive schema as IBIS ([39]), or the *Rationale Model* of Ramesh and Jarke ([37]), or REMAP ([40]), or Clement and Bass ([23], [24]). Our decision model (see Fig.3) supports this by additionally allowing to define a project individual template for the textual description component of the decision (here shortly sketched by the bullets "Assumptions", "Rationales" and "Solution Specification").

On the other hand, Booch also demands that *tactical* decisions should be documented. At that time, Booch thought both kinds would disclose itself by applying adequate modeling. Today's experiences show that such modeling just documents the *How*, but not the *Why* of decisions.

In our opinion, the developers should at least get the possibility to document decisions *on demand*, but considering aspects mentioned in Point 2 and 3, the intrusion on the development process must be minimized ([36], [30], [22], [34]). Therefore, the proposed decision model provides a minimal notational framework to identify the conflicting elements (*requiremental* and *design*) andn to derive in the resulting consequences as *DesignContraints*. Thus, the conflicting elements span an area of conflict, automatically documenting the basic rationale behind a decision.

In that case, however, the model is minimalistic and of a purely descriptive nature. Any further users of such minimalistically documented decisions must at first derive the actual knowledge about the decision on their own. But at least the fact that the context (the conflicting items and the results of the decision as *DesignConstraints*) for each decision is present provides evidence to later users: They can infer that a decision has been made consciously and first clues are given for refinding the rationale (cf. [41]).

In that way, not all decisions can be reconstructed. Since the tool discussed here shall also automatically record metadata like the author(s) of a decision, the decision's later user (*rationale seeker*) can thus consult the author(s) about unclear aspects. Additionally to tool usage, a *process rule* shall prescribe that the information seeker must document the results of this *decision recovery* in the decision's textual description to further improvement of the decision's documentation.

This procedure – inspired by Schneider ([36]: „Put as little extra burden as possible on the bearer of rationale") helps to cope with the problem in point four (see above), because by deferring the documentation work to the inexperienced rationale seeker, the experienced know-how bearers are disburdened significantly from communication/documentation work. As a positive side-effect, the transferred knowledge gets tightened in the rationale seeker during his documentation work.

On the other side, only unclear decisions will go through this further rationale request and documentation process. Therefore, the approach indirectly minimizes the documentation overhead by orienting on the selective information need of the further decision seekers.

Thus, concerning RM, our approach tries to balance and connect descriptive pragmatism and structured prescriptive methodologies. RM is not our central issue, but this chapter shows that requirement traceability and RM are very closely related to each other and complement one another.

A further general problem of RM not yet discussed here is the retrieval of documented decisions. Horner and Atwood ([30]) argue that fixed schemes – in contrast to unstructured text – offer better possibilities in indexing for retrieval. The following chapter shows how the retrieval problem can be avoided through usage of the gathered traceability information of this approach.

6 Effects on the traceability model

The idea of including decisions into the traceability models has already been proposed by Ramesh with his REMAP-tool([40]). In a later empirical study on traceability, he and Jarke ([37]) detected a real need by experienced users. Therefore they include a separate traceability submodel (*rationale submodel*) for decisions which is oriented on the former works with REMAP.

The decision model being proposed here has been inspired by the *rationale submodel,* but in our view Ramesh and Jarke's ([37]) solution lacks making concrete proposals for implementation and thus, the RM component appears loosely connected to the other traceability submodels. Besides, the *rationale submodel* (orienting on REMAP) extends IBIS ([39]), which is a prescriptive and intrusive method (cf. [42; p. 202ff]).

In contrast, our decision model directly fits into the schema for traceability to the design. In that way, a semi-formal model has evolved which provides easy handling and which has the following characteristics:

- A constellation (combination) of requirements and design elements leads to conflicts.
- Decisions do not directly influence dedicated design objects, but they bear *Design-Constraints* that can be flexibly assigned to design elements during the project.
- All other important information for documenting a decision can be added on demand as unstructured descriptive text.
- For important strategic decisions a template can provide prescriptive elements to assure these decisions have been made thoroughly.

The usage of the decision model has effects on existing traceability models. The traceability model described in Fig.1 is extended to a model briefly sketched in Fig.5.

Since design elements influence the decision process as well, the requirement dimension migrates to a close coupling with the design. Simple *<<satisfy>>* relationships can occur next to (as *Req.1* maps to *DesignElement1*), more complex traceability networks may occur. Thus, e.g. *Req.2* only impacts the design by the decisions *Dec.1* and *Dec.2*.

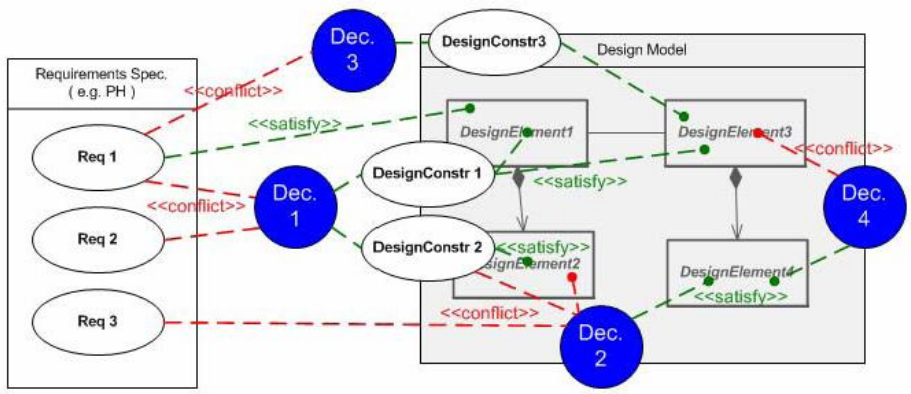

Figure 5: The newly emerging more detailed traceability information scheme.

Dec.2[5] arises from the conflict situation of *Req.3*, *DesignConstr.2* and *DesignElement2*, whereas *Dec.3* is only derived from requirement Req.1 (which then corresponds to a <<Derive>>-relationship as described in [5;p.33]). Consequently, *DesignElements* alone should also be able to invoke a decision (*Req1→Dec.3→DesignConstr3*).

With adequate tool support, these traceability relationships as indicated in Fig.5 could be visualized as a traceability tree. A kind of browser should support:

- Detailed impact analyses: Starting with a *starting impact set*, all subsequent paths would firstly be classified as *impacted*. During the following detailed check, the tool should allow to take out paths identified as none-relevant and to add paths detected as relevant.
- An adequate context for the simple retrieval of documented decisions.

7 Example

To contrast the differences between current approaches and our proposal, we use an example system requirements specification (SYS-RS) with three requirements causing a problem encountered by the Micron AG by one of its projects:

- Req.1: An external watchdog component must monitor the system.
- Req.2: Parametric data must be changeably by the customer during operation.
- Req.3: Parametric data must be stored in EEPROM.

7.1 Current practice

In current practice, the system design determines that the system will include a micro controller (controller), an external watchdog component and an external EEPROM (cf. Fig.6).

5 *Dec.2* is directly mapped to *DesignElement4*. This may also be possible, when no further information for understanding the decision is needed.

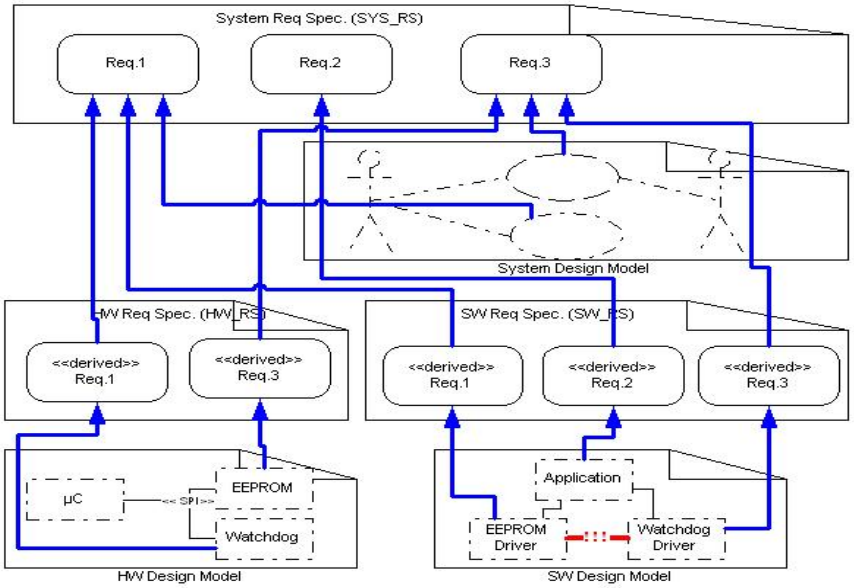

Figure 6: The example in previous approaches

The HW requirements specification (HW-RS) derived from the SYS-RS and system design again contains *Req.1* and *Req.3* linking back (Fig.6: fat blue arrows) to the SYS-RS. The detailed HW-design determines that watchdog and EEPROM will share the connection pins to the controller by an SPI[6]-communication interface, because other connected components have already used up all remaining pins of the controller. Req.1 gets linked to the watchdog symbol and Req.3 to the EEPROM symbol in the HW design.

The SW requirements specification (SW-RS) contains Req.1, Req.2 und Req.3 linking back to the SYS-RS.

During SW design, the architect discovers the potential resource conflict in the shared usage of one SPI for EEPROM and watchdog. Since driving the EEPROM is very time intensive and triggering the watchdog is very time critical, the architect rates this combination as risk, but changes of the HW are rejected due to higher costs. The solution for this conflict, the EEPROM and watchdog drivers must be „artificially“ coupled to implement a cooperative handshake[7] solution (Fig.6: red association with "!!!").

The solution implies that the planned original standard drivers of a supplier must be adapted internally. In the further progress of the project, these adaptions caused extra effords not traceable to its background.

7.2 The new approach

The system design is done similar to Chapter 7.1 (Fig.7). The SYS-RS contains an attribute that allows a SW-HW partitioning. Req.1 and Req.3 are marked as relevant for HW and SW, Req.2 only for SW.

6 http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

7 When triggering needed soon, the watchdog requests the EEPROM-module, which handles preempting its task in a secure state.

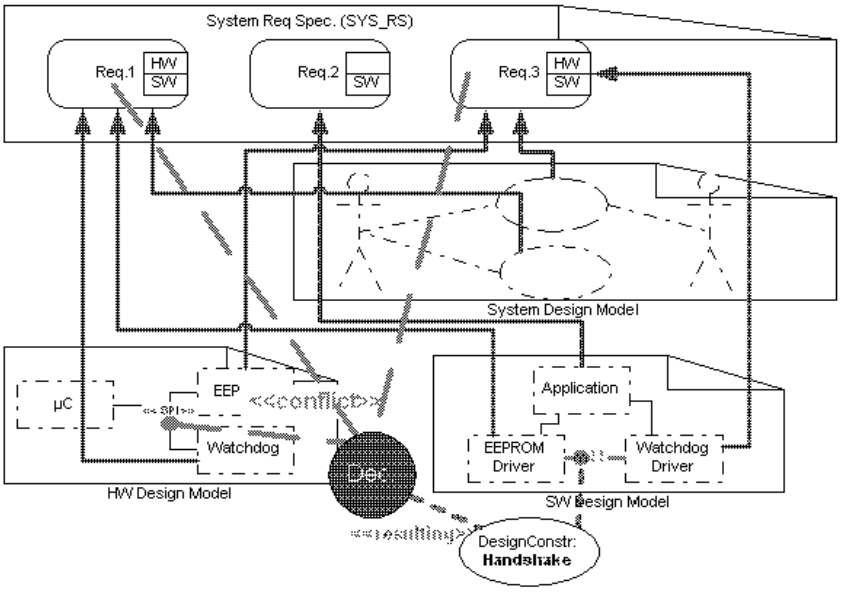

Figure 7: The example the new way

The HW-RS is not directly applied, since the relevant HW requirements are marked in the SYS-RS. The HW design is done similar to Chapter 7.1 and linked to the Req.1 and Req.3 in the SYS-RS.

The SW-RS is not applied, since the relevant SW requirements are marked in the SYS-RS. The SW design will be developed from the SYS-RS and the system design model. The architect discovers the same problem concerning watchdog and EEPROM. He opens a decision wizard and marks Req.1 and Req.3 as conflicting and links to the HW-design diagram that documents the conflict. As further rationale, the architect textually documents „synchronization conflict at SPI between time intensive EEPROM application and time critical watchdog application“. A further click helps the architect to put the conflict into the risk list.

As resulting DesignConstraint, the architect sketches the cooperative handshake and links the DesignConstraint to the EEPROM and watchdog design elements in the SW design.

Our implementation follows the ideas described in Chapter 7.1. In the further project progress necessary changes are detected early by impact analyses and the additional costs can be compared to the cost savings of the rejected HW change.

The artefacts HW-RS and SW-RS not realized can be generated out of the model, on demand by summing up all requirements related to the corresponding design (HW design model for the HW-RS, SW design model for the SW-RS).

8 Summary and prospective

This article shows the interdependencies between the SPICE layered process model, requirements, traceability, designs and decisions with special attention on as low redundancy in the traceability information as possible. We plea for a strict separation between contractual mandatory requirements (real requirements) and requirements resulting from former design decisions (design constraints).

Furthermore we try to show in how far decisions are the link between requirements, designs and from those derived DesignConstraints. This closely connects and synchronizes approaches in requirement traceability and rationale management to lift synergic effects.

These remarks suggest, as has already been noted in the literature ([43], [6], [8], [44], [37], [11]), that the influence of requirements on design processes -and vice versa- is only insufficiently modeled by bidirectional linkages.

Therefore, a cooperation between Micron AG, the Competence Center for Software Engineering of the University of Applied Sciences Regensburg and the Media Computing Group of the University of Regensburg work on a prototype system that enhances currently used traceability approaches by the solutions sketched above. The main goal is to find a solution to document traceability information in a lightweight fashion and, as a by-product, during design processes (cf. [45]). The decision model for deriving further design constraints discussed here will be a key step towards this goal.

References

[1] O. Benediktsson, R. Hunter and A.D. McGettrick. Processes for Software in Safety Critical Systems. In *Software Process: Improvement and Practice*, volume 6 issue 1, p. 47-62, 2001

[2] K. Hörmann, L. Dittmann, B. Hindel and M. Müller. *SPICE in der Praxis, Interpretationshilfe für Anwender und Assessoren,* dpunkt Verlag, Heidelberg, 2006.

[3] R. Kneuper. *CMMI. Verbesserung von Softwareprozessen mit Capability Maturity Model Integration. Volume 2,* dpunkt Verlag, Heidelberg, 2006.

[4] Ch. Rupp. *Requirements-Engineering und –Management, Volume 2,* Hanser, München, 2002.

[5] M. Lindvall. *A study of traceability in object-oriented systems development.* Licenciate thesis, Linköping University, Institute of Technology, Sweden 1994.

[6] A. von Knethen. *Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems,* Fraunhofer IRB Verlag, Stuttgart, 2001.

[7] Ch. Ebert. *Systematisches Requirements Management,* dpunkt, Heidelberg, 2005.

[8] B. Paech, A. Dutoit, D. Kerkow and A. von Knethen. *Functional requirements, non-functional requirements, and architecture should not be separated* - A position paper, REFSQEssen, 2002.

[9] O. Gotel, O., A. Finkelstein. An Analysis of the Requirements Traceability Problem. *ICRE'94.*, pp. 94–101, 1994.

[10] M. Hause. *Successfully Managing An Incremental Real-Time Project; Part Three: Requirements Managemen*t. 1999, http://www.artisansw.com; (Access 10/2005).

[11] S. Allderidge. A Use Case Driven Safety Critical Programme; *INCOSE UK Spring Symposium* 2003. http://www.artisansw.com/pdflibrary/DS&S_INCOSE_2003.pdf, (Access 11/2005).

[12] Object Management Group (ed.). Systems Modeling Language Specification. *OMG adopted specification, Mai 2006*, http://www.sysml.org/docs/specs/OMGSysML-FAS-06-05-04.pdf, (Access 10/2006).

[13] K. Pohl and E. Sikora. Requirements Engineering für eingebettete Systeme. In *[20],* P.101–140, 2005.

[14] B. Nuseibeh. Weaving together requirements and architectures. *IEEE Computer*, 34 (3) p.115-117, 2001.

[15] J. Dick and J. Chard. *The Systems Engineering Sandwich: Combining requirements, models and design.* 2004. http://www.telelogic.com/corp/download/index.cfm?id=3640; (Access 06/2006).

[16] O. Nierstrasz. Software Evolution as the Key to Productivity. In *A. Knapp,M. Wirsing and S. Balsamo (Eds.) Radical Innovations of Software and Systems Engineering in the Future*, LNCS, vol. 2941, Springer-Verlag, 2004, P. 274-282.

[17] B. Boehm. *The Future of Software and Systems Engineering Processes.* 2005. http://sunset.usc.edu/publications/TECHRPTS/2005/usccse2005-507/usccse2005-507.pdf (Access 10/2006).

[18] A. Hunt and D. Thomas. *The pragmatic programmer*. Hanser, München 2003.

[19] B. Boehm and R. Turner. *Balancing Agility and Discipline- A guide for the perplexed.* Addison Wesley 2003.

[20] P. Liggesmeyer, P. and D. Rombach (Eds.): *Software Engineering eingebetteter Systeme Grundlagen - Methodik – Anwendungen*. Volume 1., Elsevier, München, 2005.

[21] A. Dutoit, A., R. McCall, I. Mistrik and B. Paech (Eds.). *Rationale Management in Software Engineering*. Springer, Berlin 2006.

[22] A. Dutoit, R. McCall, I. Mistrik and B. Paech. Rationale Management in Software Engineering: Concepts and Techniques. In *[21]*, p 1-48, 2006.

[23] P. Clements, F. Bachmann and L. Bass. *Documenting SW Architectures-Views and Beyond*. Addison Wesley 2002.

[24] L. Bass, P. Clements, R.L. Nord and J. Stafford. Capturing and Using Rationale for a Software Architecture. In *[21]*, 2006.

[25] M. Riebisch. Prozess der Architektur- und Komponentenentwicklung. In *R. Reusser, W. Hasselbring (Eds.): Handbuch der Software-Architektur*, dpunkt, Heidelberg, p.65-88, 2006.

[26] T. Posch, K. Birken and M. Gerdom. *Basiswissen Softwarearchitektur- Verstehen, entwerfen, bewerten und dokumentieren.* dpunkt, Heidelberg, 2004.

[27] M. Gerdom, T. Posch. *Pragmatische Software-Architektur für Automotive Systeme.* OBJEKTspektrum, 05/2004, p.64fff, 2004.

[28] G. Booch. *Object-Oriented Analysis and Design with Applications.* 2nd Ed., Addison Wesley, Santa Clara, 1994.

[29] A. Tang, M. Ali Babar, I. Gorton and J. Han. A Survey of the Use and Documentation of Architecture Design Rationale, In *Proc. of 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, 2005

[30] J. Horner, M.E. Atwood. Effective Design Rationale: Understanding the Barriers. In *[21]*, p.72-90, 2006.

[31] J. Grudin. Evaluating opportunities for design capture. In *T.P. Moran, J.M. Carrol (Eds.). Design Rationale: Concepts, Techniques, and Use.* Erlbaum Associates, Mahwah NJ, 1996.

[32] D. Schön. *The reflective practitioner. How professionals think in action.* Temple Smith, London, 1983.

[33] M. Polanyi. *The Tacit Dimension.* Doubleday, New York 1966.

[34] F. Shipman III and C. Marshall. Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems. *Computing Support Cooperative Work,* 8 (4), p.333-352, 1999

[35] M. Lehman and J. Fernández-Ramil. The Role and Impact of Assumptions in Software Engineering and its Products. In *[21],* p.313-328, 2006.

[36] K. Schneider. Rationale as a By-Product. In *[21]*, p. 91-109, 2006.

[37] B. Ramesh, M. Jarke. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering*, Vol. 27, No. 1, 2001.

[38] Tyree, J.; Akerman, A.: Architecture Decisions: Demystifying Architecture. *IEEE SW*, vol. 22(2), p.19-27, 2005.

[39] W. Kunz, H. Rittel. Issues as elements of information systems. *Working Paper 131, Center for Urban and Regional Development*, University of California, Berkeley, 1970.

[40] B. Ramesh and V. Dhar. Supporting Systems Development by Capturing Deliberations During Requirements Engineering. *IEEE Transactions on Software Engineering*. Vol. 18, No. 6, JUNE 1992.

[41] R. Roeller, P. Lago, H. van Vliet. Recovering architectural assumptions. *The Journal of Systems and Software*, 79, 2006.

[42] P. Louridas, P. Loucopoulos. A Generic Model for Reflective Design. *ACM Transactions on Software Engineering and Methodology*, Vol. 9, No. 2, April 2000, 199–237.

[43] A. von Knethen. A Trace Model for System Requirements Changes on Embedded Systems. In *Proc. of 4th International Workshop on Principles of SW Evolution.* Sept. 2001.

[44] R. Pettit. Lessons Learned Applying UML in Embedded Software Systems Design. *Second IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*. Wien, 2004.

[45] K. Pohl. PRO-ART: A Process Centered Requirements Engineering Environment. In M. Jarke, C. Roland, A. Sutcliffe and R. Dömges (Eds.)): *The NATURE of Requirements Engineering*, Shaker Verlag, p. 255-278, 1999.