IV. Synopsis

This is not the end. It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.
Winston Churchill

Now, this last part finishes this thesis. At first, a short summary of the achieved results of this research project is provided. This is then followed by a prospectus of possible further questions to continue research on, either improving the current features set of R2A or more general on the research topics of this thesis. At the end, the author tries to summarize the general conclusions to draw from this thesis.

IV.25 Summary of the Achieved Research Results

To achieve anything worthy to be called quality you will have
to do a good deal more than follow a drawing or specification,
whoever made them and however carefully.
There is a good and close parallel to music.
The quality of a performance depends
on the performers as much as on the score.
The performers are said to be interpreting the score,
but in fact they are adding intention of their own
to those of the composer, recognising that no score
in practice can fully express the intentions of the composer,
that it can never be more than an indication, a sketch;
and no designer can in practice ever produce more than a sketch
even though his drawing is dimensioned in thousandths
of an inch and his specification is as long as your arm.
[Py78; p.80]

In the following the main technical innovations achieved through PROVEtech: R2A (in the further called R2A) are summarized:

- *Hierarchic decomposition* of a system (or software) is an old idea in *SE* (see *structured analysis* and *design* [De78]). In UML based design, this view is seen as one besides many others with equal rights (see, e.g., the *view* concept *"4+1 View Model"* by Kruchten [Kr95]). UML does not prefer any *view* or

B. Turban, *Tool-Based Requirement Traceability between Requirement and Design Artifacts,*
DOI 10.1007/978-3-8348-2474-5_5, © Springer Fachmedien Wiesbaden 2013

make relations between *views* explicit. Instead, defining *views* and their relations are left open to *architecture documentation*. However, this leads to a more difficult understanding of a designed model since all *views* and elements are mixed up in one egalitarian repository (see fig. 15-3 in ch. III.15). Besides, the heterogeneous *view* concept of UML makes UML incompatible with other modeling techniques used in embedded development such as ETAS ASCET or Matlab, which only use one *hierarchic decomposition view*. As R2A makes only one necessary assumption: that a design must be made using a *hierarchic decomposition* (in fact a claim that can be called state-of-the-art), the approach should be compatible to any other computer tool-based design approach and even to HW or *computer aided design* (*CAD*). To include such a tool, only an interface implementation for R2A's modeling tool *variation point* connecting to the corresponding tool would be necessary. Development experiences within the R2A-project have shown that this is possible within a two to three person month's development effort.

- As shown in ch. III.16.2, the mechanism of coupling modeling tools in R2A is even capable to integrate models of different modeling tools in one integrated model. In this way, all achievements described below can also be used as an embracing method to generate an integrated model, crossing tool gaps between different modeling tools. This allows using specific modeling tools together in an integrated model. In this way, it is possible to employ the specific strengths of the specific tools in one integrated model.
- As not explicitly discussed yet, but the approach for *traceability* can be equally used to establish *traceability* between requirements and an *AM*, when, e.g., a UML-tool is used to create the *AM*.
- In the approach shown here, the *hierarchic decomposition* builds the spine of the complete model because each element of the *design model* gets explicitly included into the *abstraction hierarchy tree* and is extended to a so called *abstraction node* (*AN*) having extended semantics (cf fig. 15-2 in ch. III.15). To each *AN* further diagrams can be added as additional *views*. Through this way, the orientation of the designers is alleviated as at first navigation into the *abstraction hierarchy* to the desired element can take place (vertical direction). Starting from this, also navigation along the further attached views

is possible[267] (horizontal direction). However, the problem is still unsolved that some of the remaining views of a model may go *crosscutting* over different *abstraction layers* and *ANs*. At the moment, it is possible to add diagrams with such characteristics to all *ANs* touched by the view, but finding an even more consistent solution for this, is an open point for further research.

- Additionally, the *ANs* tree arisen through *hierarchic decomposition* also builds the spine for the structured approach for *requirements traceability* to design establishment. Differently to approaches, where requirements are simply added to a *design element* via direct linking, the R2A provides a complete new approach to the problem called the *requirement dribble process* (*RDP*). In this approach, developers at first do not need to know by which design solution a *requiremental item* (*RI*) will exactly be fulfilled. Instead a designer can at first assign an *RI* to *ANs*, where she roughly grasps that the *RI* may be fulfilled by. Then, when the designer's vision gets clearer about an *RI*, the designer may use the *dribble-down* and *dribble-up* actions to reallocate the *RI*. In this way, on one side Simon's idea about *stable intermediate forms* (ch. I.6.2.1) is supported, and on the other side the uncertainty and flexibility of the approach directly supports designers in their *knowing-in-action* phase.
- Through the support of a dedicated process for assigning *RIs*, it is ensured that each *RI* is adequately considered in the design process: If new *RIs* are assigned to an *AN* 'from above' (a higher-level *AN*), these *RIs* get highlighted in the *AN* by a bold font style Now, the designer of the *AN* must try to find an adequate solution for the newly assigned *RIs*. If the designer of this *AN* is again able to delegate these requirements to a sub *AN* of the design, then these *RIs* '*dribble down'* one level deeper to a sub *AN*, and the problem is solved for the corresponding *AN*. However, if the designer is not able to clearly delegate these *RIs* to any sub *AN*, then the *RIs* stick to this *AN* and are inherited to all lower-level sub *AN*s (marked 'gray' at these lower levels) indicating that all *AN*s must work together to fulfill these *RIs*. But, if the designer responsible for the *AN* realizes that these newly assigned *RIs* cannot be fulfilled in the current state of design, the designer is able to repel these *RIs* back to the higher-level *AN* (its origin) accompanied with a corresponding note. In this

[267] As described in the point before, the usual orientation within modeling tools is in most cases realized by a repository concept, where all items present in a model are shown (see fig. 15-3 in ch. III.15). This repository is not touched by R2A. On the contrary, R2A's *AN* concept with its representation in an *ANH* tree can be seen as a distillate of the most important information on the most important items and their relationships present in the modeling tool repository. Whereas, the *modeling tool repository* is more a dictionary containing all someway present items in a model.

case, the designer of the higher-level *AN* must take care for a solution under consideration of the created notes.

- As a dedicated goal of the *RDP*, the process set aims on principles leading to a way to find an allocation for a requirement at an *AN* at the lowest level of abstraction to ensure that a requirement is implemented as local as possible. In this way, the *impacts* of a later requirement change are also limited as local as possible.
- In the wake of this goal, the concept of the r*equirement influence scope* (*RIS*) has been developed. The concept includes that a requirement being associated to an *AN* also is propagated to the child *ANs* as "*inherited*" requirement. In this way, on one side pressure is imposed on the project team members to bring requirements to the most possible local level in the *ANH* tree. Thus, a later possible change of the requirement has only minimal *impact* (cf. ch. III.18.2.2). The *RIS* thus promotes a heuristic enforcing a design with emphasis on localization of the requirements. This heuristic is an essential part of the ideas behind the *RDP*.
- As a further plus, the history of the different requirement allocations and reallocations during the *RDP* are automatically tracked via *configuration management* features. In this way, the decision process of the designer taking the decision can be reconstructed later. This follows ideas of Gruber and Russell [GR96a] or Schneider [Sch06] to capture important information during performed action and extracting important *rationale* information later as a *by-product*.
- Some *traceability* research rather neglects the aspects about the process of *traceability* establishment (see, e.g., [Kn01a], [Kn01b]). In the author's view, however, this issue might be the central key problem of *traceability* since *traceability* faces a significant *benefit problem* (see ch. II.10.5) in a similar way as *RatMan* approaches do (ch. II.9.4.2). As a consequence, R2A's *traceability* mechanisms try to allow capturing *traceability* as a mere *by-product* of normal design activities, where designers can perceive direct benefit from recording *traceability* information. To achieve this goal, R2A at first allows capturing *traceability* by several possible *drag-and-drop* operations being performed easily and quickly as *by-product* of the decisions performed. Secondly, R2A directly shows the *RIs* assigned to each *AN*, thus directly giving designers benefit for their *traceability* work as this work is used to provide a sorted out view on the *RIs* important for the currently considered design aspect from the otherwise numerous manifold of *RIs* present in a *requirements specification*. Further, with the *RIS* and *RDP* concepts, design decisions about requirements allocation are automatically captured, following the principles of Simon's idea about *stable intermediate*

forms (ch. I.6.2.1). In the author's view, the principle of *stable intermediate forms* directly reflects how designers usually develop a design out of the requirements at hand. Thus, through *RIS* and *RDP*, traces can be directly captured according to their occurrence with special emphasis on flexibility and optimal adaption into a SPICE-conforming process landscape. These concepts can also be seen as an attempt to adapt *traceability* establishment activities to the way designers are thinking, thus preventing a *cognitive dissonance* for designers. In this way, R2A tries to be less *intrusive* to designers' thinking, thus supporting designers in both, their *knowing-in-action* and *reflection-in-action* phases (see Schön, ch. I.6.2.3).

- Significant parts of the research about *traceability* concentrate on proposals to use richer *traceability* models as a kind of *conceptual trace model* (cf. ch. II.10.4.2). In the author's view, these approaches provide important points. However, it is questionable whether their *formality* is not too complex (i.e. complicated; see footnote 80 (p.77)) for activities that should best be performed as a *by-product* (see ch. II.10.5 and ch. II.9.4.2). The research attempt shown here tries to integrate good ideas from these research attempts into a complete concept. As a result, a *requiremental items taxonomy* has been developed, distinguishing real *requirements* from the customer from *RIs* (*DCs* and *BRCs*) arising as consequences from design decisions.
- This – as a further result – also has sparked the idea to enhance R2A's *traceability* concepts by an *integrated decision model* for documenting *requiremental* and design-based decisions (cf. ch. III.20). Thus, the developed *decision model* is called *integrated* because it directly integrates information about design decisions into a *traceability* concept. Again following the idea that this additional information must be rather captured as a *by-product*, the *decision model* is construed as a *semi-formal model*, where the *formalisms* build a *skeleton* to easily sketch the basic information about a decision and to add more detailed information on demand. In this way, the *decision model* on one side addresses *benefit problems* encountered for *capturing rationale* (ch. II.9.4.2) but on the other side also allows capturing deeper *rationale* information for problems of rather *wicked* nature (cf. ch. I.6.2.2). As ch. III.20.4 has outlined, the decision model is also a good means for fulfilling demands on decision documentation, imposed by research about *architecture documentation* (e.g., cf. [Ha06], [CBB+03]), and much closer integrating the thus captured information with the *design model*.
- The *decision model's* concept of modeling conflict situations and consequences resembles to the *pattern* concept expressed by Alexander (ch. I.6.2.4). In fact, as ch. III.20.5.1 shows, R2A's *decision model* can be a decisive means to document the *rationale* behind *pattern* usages to directly integrate the *pattern*

concept with the *traceability* concept, to help to better include consequences of *pattern* usages with other decisions and to help to discover new *patterns*.

- As embedded design (but also other design) must often care for adequately managing *resource restrictions*, the R2A approach offers a second *decision model* to capture decisions about *resource restrictions* that can be managed as budget. The decision results are again expressed as new *RIs* assigned to *ANs*, expressing the need that an *AN* does not exceed the resource budget that was assigned to it. To cover this aspect, the *requiremental items taxonomy* has been extended by the *RI* called *BRC* expressing the budget character of the *RI* and the budgeting decision process.
- The arrangement of the *design models* in an *abstraction hierarchy tree* and the way requirements consideration in design can be handled by the *RDP*, suggest the conclusion that adequate mechanisms can help to significantly improve the flow of communication between project stakeholders. In R2A, this can be achieved by temporal decoupling (asynchronous temporal communication) of messages preventing, for example, that important information is not adequately propagated if the responsible developer is not present. Such a mechanism is intended to support goal-oriented creation of notes (cf. ch. III.17.2) for any entity present in the data model and actions performable on the entities. These notes allow sketching occurring problems with references to all affected model entities and propagating this information to concerned stakeholders.
- At the same time, these notes are included into the *history* function (cf. ch. III.17.5) in order to better enable later reconstruction of the incident's occurrence[268] (e.g., helpful during a SPICE assessment). In this context, further research attempts could enhance the mechanism described here by state-based notes (e.g., with the states: 'New', 'In work', 'Processes', or 'New solution'), or escalation paths in a consistent process-driven model.
- Additionally, all these concepts allow a high degree of flexibility to change *traceability* information again. This flexibility is also especially helpful to support *design refactorings*, where *traceability* information is also adapted correspondingly, thus preventing significant degradation of *traceability* information captured once.
- This directly segues to the next topic: *Traceability* is intended as means to manage requirement changes. Through the *graphical impact analysis* concept (see ch. III.22.1.1), R2A allows proposed requirement changes to be better predicted and helps to implement once decided requirement changes. An especially important point is to consistently infer and propagate all requirement

[268] Up to now, results are often only discussed and tracked orally.

changes throughout the complete model. R2A achieves this through the consistency management mechanism described in ch. III.22.2.

- In many development projects, parts of a project are delivered by a supplier. Correspondingly, *supplier management* is an important task in these projects. One of the most essential issues addressed is that the requirements for the supplied parts must be formulated in a way that the supplied parts fit together with the other designed parts of the system. R2A allows generating a *requirements specification* directly for a part of a design (an *AN* or a sub tree in the *ANH*), which can then be used as *supplier requirements specification*. In this way, all information about requirements, design and decisions performed in a R2A project relevant for a supplier of a part can be directly propagated to the supplier without unintended information loss due to tool gaps or other potential breaks in the information chain (see ch. III.23.1).
- Last but not least to mention, the mechanisms of generating *requirements specifications* for parts of a design described in ch. III.23.1 can also be used to implement a direct and seamless information propagation for situations, where a project has several requirement and design processes at different layers of abstraction as it is demanded by SPICE. Even though the author himself rather prefers an integrated design process for the different layers as it is described by fig. 23-1 in ch. III.23.2, fig. 23-2 in ch. III.23.2 shows that R2A also has the potential to improve the information flow in cases the process demands of SPICE shall be fulfilled word for word. As the ch. III.23.3 shows, R2A can even be used to achieve a temporal decoupling of the development of the different requirement and design artifacts.

IV.26 Perspectives for Further Research

It's like deja-vu, all over again.
Yogi Berra

The current research results of the R2A-project also provide perspectives for possible further research. In the following, problems or ideas are outlined that may raise interesting research questions:

- The current solution of R2A has made a significant simplification concerning the *view* concept. In R2A, *views* are merely represented by one diagram. This does not consider more complex *views*. However, in design documentation theory, a *view* often consists of a set of diagrams that must be considered together. R2A currently only considers this fact by the *ANH*. This brought the

advantage that the user can more easily navigate in the model, but on the other side other *view's* being more complex than one diagram might be scattered over several *ANs* and the relationships between these diagrams may not be adequately surfaced by a model. Further research could concentrate on finding a solution, in which several diagrams could be integrated into a more complex *view* other than the *ANH view*, and where, however, the advantages of better model navigation as provided by the current R2A-solution are still present.

- In the context of the *RIS* (ch. III.18.2.2) and the *RDP* (ch. III.18.2.4), the author has only spoken from *RIs* 'assigned' to an *AN*. This leaves open space for interpretation of the concrete semantics of any relationship. In fact, different *traceability CTMs* (see ch. II.10.4.2.3) know different relationship types between *RIs* and design. The R2A-approach could be extended to allow designers to define a concrete semantics of a relationship. However, further research should then also ensure that this extension is not just leading to further complication without significant gains of value.
- In this context, a further interesting idea may be to have a relationship describing fuzziness concerning the kind of connection between an *RI* and an *AN*. Instead of 'assigned' relationships, currently describing the fact that an *AN* is directly influenced by an *RI*, there might exist relationships having notions like 'bordering' (the requirement is fulfilled nearby, thus the *RI* should be monitored whether it possibly has some influence), 'keep in mind' and 'I don't know, but might be important'. By such fuzzy relationships designers could identify connections, for which they 'feel' that there is a dependency they cannot describe rationally. This corresponds to Schön's observation that designers also work in a state of intuitive *knowing-in-action*, where they use *tacit knowledge* and thus cannot rationally explain their exact thoughts.
- Ch. III.18.2.2 describes a mechanism where the scope of a requirement is determined by the so-called *RIS*. When *RIs* are added to an *AN*, these *RIs* are automatically inherited to all child *ANs* of the *AN* in the *ANH*. In this way, developers are spurred to find the most local solutions for an *RI*. On the other side, effects of *nonfunctional RIs* can be made more transparent as than it is possible by other approaches. *Nonfunctional RIs* can be added to a very high-level *AN*, where they are inherited by large extents of *ANs*. Taming *nonfunctional RIs* is rather difficult. In the author's opinion, the *decision model* introduced in ch. III.20 proves very helpful as it allows documenting decisions about the taming strategies of *nonfunctional RIs,* allowing deriving more concrete *DCs* as decision consequences. Now, if this is thought through consequently, it may be possible that a *nonfunctional RI* is tamed by decisions, where more concrete *RIs* (*DCs* or *BRCs*) are derived. It should be

considered whether a feature may be helpful to specify that a decision or several decisions together completely tame an *RI*. It must be analyzed whether it would be a logical consequence that an *RI* tamed by one or more decisions may lose its inheritance status to lower-level *ANs* (lose its *RIS*) because its influence would rather take effect through the decisions and the effects of the *RIs* resulting as consequences. Such considerations, however, must also consider that such an effect may not be realized for any decisions, but it would rather be necessary to mark certain decisions as the taming decisions of an *RI* leading to the deactivation of the *RI's RIS*. In this way, it is questionable to a certain degree whether such a feature brings significant extra value to designers or whether it just implies a further *complication* to the design (see footnote 80 (p.77)). In the case of the latter, the author would recommend leaving out the question, even though it might be slightly more logical than the current solution.

- At the moment, *consistency checking* is a rather neglected topic of this research even though rudimentary *consistency checking* can be provided by the *rule engine*. Analyses on what *consistency reporting* is necessary for users could be performed. A further problem may arise with the fact that R2A rather relies on heuristics such as the *RDP* (ch. III.18.2.4) and the *decision models*. These heuristics imply a certain *non-linearity*. As here traces cannot be followed so directly, this could make *consistency checking* more difficult. For example, usually *consistency checking* mechanisms rely on checking whether all requirements are someway associated to a *design model*. If this is the case, it is assumed that the requirement is adequately considered in the design process. The author, however, is rather skeptical towards the real expressiveness of such rather simple checks. With R2A, however, such simple checks are even not possible because the *RDP* heuristics allows assigning requirements to *design elements* not being the final destination of the requirement. Instead, the requirement assignment can change with *dribble-down* or *dribble-up* operations in order to support design decision-making. In this case, a requirement can only be seen as adequately considered after the requirement has reached its final destination. The situation can get even more complicated when a requirement is part of documented decisions. Here, e.g., the question arises whether a requirement can only be seen as adequately considered when all consequential items of all decisions involved have reached their final destination. In the author's view such a developed *consistency checking* mechanism would provide significantly more fine-grained information than current *consistency checking* approaches and thus provide even stronger expressiveness. But, because all effects of such a

consistency checking heuristics are not yet analyzed, this point is rather an open question for further research.

- Pinheiro indicates that for capturing *nonfunctional traces*, *hypermedia (multimedia) systems* could provide significant support ([Pi04; p.104-105], see ch. II.10.4.2.2). The approach proposed here offers possibilities to tackle *nonfunctional traces* via the integrated *decision model* (cf. ch. III.20). It would be possible to couple the *decision model* approaches with *rationale* tools like Compendium, supporting *rationale capturing* on the fly as well as with other media objects such as tape or video recordings of meetings, in which the corresponding decisions are discussed.
- A design process is also driven by other documents such as meeting protocols, review protocols or documentation of the used COTS[269]-components. In the author's opinion, it will never be possible to integrate all documents important for development into one tool solution. Correspondingly, it should be possible to have a *hyperlink* concept to give developers the freedom to link to further documentation, someway not manageable in R2A. As projects usually use *configuration management* tools to manage versions of all documents in a project, it may be interesting to integrate R2A with configuration management systems via the standardized CVS[270]-interface.
- In *issue tracking* (i.e. *change management*) systems open issues (e.g., problems or bugs) can be managed. A direct connection of R2A to issue tracking systems could help to make influences of issues transparent, because often issues beyond requirements or requirement changes exist having influences on design decisions. The exact way of integration should be analyzed by further research. However, a starting point for integration could be to shape the integration in a similar way as the integration of *REM-tools* has been made: A continuous synchronization process cares to have all issues in an accurate state in R2A and these issues are then treated analogously to *requiremental items*. As the description to arrow '1.' of fig. 20-8 (see ch. III.20.4) describes, a better integration with change management tools might help to solve information backlashes to requirements occurring during design and especially during processes of discovering *rationale* in decision processes. However, it must be noticed that issues are slightly different to requirements because only certain issues may be interesting for design and

---

269 Commercial Off The Shelf

270 *Concurrent Versioning System*: This interface is an international standard for integrating configuration management systems with other environments such as programming IDEs.

should therefore be synchronized in R2A. This means a filter must distinguish the architecturally significant issues from the insignificant issues.

- *Impact analysis* (*IA*) approaches as [Ha99] propose combining a tracing approach with a kind of *dependency analysis* approach using the relationships in a model. In this way, effort to capture *traceability* information is reduced by using the model relationships present in a model. As ch. II.10.6.2 has shown, however, such approaches often lead to the so-called *fan-out* effect [Al03] because models contain manifold relationships having other purposes leading to many unnecessary traces. Correspondingly, the R2A approach rather concentrates on achieving more exact results by allowing establishing dedicated more *fine-grained traceability* information as a *by-product* of usual development effort, thus reducing efforts for *traceability*. However, on the other side, dependency information in the model can be valuable to indicate other possible *impacts* resulting from interconnections within the model. Thus, it may be possible to combine the current R2A approach with a dependency approach automatically analyzing all other relationships created in the *design model*. To avoid the *fan-out* effect, R2A's *IA* could show these *impacts* identified from *dependency analysis* with a different iconification (as it is already done for distinguishing *direct impacts* from *indirect impacts* derived from decisions or *inherited impacts* derived from the *RIS*) to distinguish them from *impacts* derived from captured *traceability* information within R2A. Further, it could be possible that this additional *dependency analysis* can be activated or deactivated for *IA*. In this way, designers could have additional support for identifying possible *impacts* from interconnections within the model but also ignore the information if they feel it is not helpful.
- Ch. II.10.2 further indicates that with *model-driven development* methods and tools a new problem arises concerning *traceability*: As code then often is generated from models, some requirements are not necessarily implemented through the models but by setting parameters or choosing specific model transformation procedures over other procedures [AIE07]. This means that *traceability* tools should also need to map requirements to parameter choices or transformation procedures of the modeling tool. Currently in R2A, *traceability* to these items could be achieved by a documented decision, where the requirements are in the conflicts section and the resulting section contains *DCs* with the chosen parameter settings or transformation procedures. Further research, however, could also try to find more adequate support by R2A for this tracing problem.
- Another research direction may be to integrate a metrics approach with R2A. Ch. III.20.5.3 indicates that architectural evaluation and identification of

neuralgic points can be supported by combining R2A with metrics. Further research could evaluate the potential of the ideas about metrics sketched in ch. III.20.5.3.

$$CEF(\mathrm{Re}q) = \sum (AN_{direct} * Level_{\mathrm{AN}} + coeff_{inher} * \sum ANs_{inher} + coeff_{Dec} * \sum Decs_{\mathrm{Re}q}) \quad (26.1)$$

$$Level_{ANH} = coeff_{lvl}^{(LowestLevel - Level_{\mathrm{Re}q})} \quad \text{where} \quad coeff_{lvl} > 1 \quad (26.2)$$

$$Avrg(CEF) = \frac{\sum (CEF(\mathrm{Re}q))}{Count_{\mathrm{Re}qs}} \quad (26.3)$$

- A further metric possibly interesting to evaluate could help to determine the changeability of an *RI*. As indicated by ch. III.18.2.2 and ch. III.18.2.4, a *RI* should be as local as possible. A *change effort factor* (*CEF*) metric could measure the locality of a requirement by calculating the directly assigned *ANs* in relation to the hierarchical level in the *ANH* (is it high or low in the hierarchy?), the number of *ANs* where the *RI* has been inherited to and the number of decisions the *RI* is involved in. Formula (26.1) sketches a possible measurement formula for the *CEF* metric. The formula uses a level factor[271] calculating (formula (26.2)) a factor to determine the hierarchy level dependent complexity of each directly assigned *AN*. In this way, the metric could help to estimate the effort for changing a *RI*. From a higher perspective, this metric could also be used to create a metric to evaluate an *architecture* according to the average changeability of requirements. The average changeability could be calculated by the sum over the changeability of all requirements divided through the number of requirements (e.g., formula (26.3). Here, it is to mention that the metrics as proposed here are just rough sketches. Further research could deal with how to adapt parameters (different 'coeff' variables) in the sketched formulas to achieve distinctive, meaningful results. Afterward, the metrics need to be evaluated in several practical projects to get measuring scales for the practical meaning of the measured metrics.
- As described in ch. I.7.4, *verification criteria* for design artifacts must be defined and these must be made traceable [MHD+07; p.225ff]. At the

[271] Through the level factor with its level coefficient, the complexity of the *design model* is taken into account because the coefficient grows exponentially with the number of abstraction levels present in a design. When, e.g., a design grows by new abstraction levels, requirements added to higher level (resp. more abstract) *abstraction levels* lead to a significantly higher CEF (assumed a corresponding adequate value for the coefficient is chosen). In this way, the author assumes that the 'Avrg(CEF)' function also grows stronger for designs having more *abstraction levels*.

moment, this can be achieved in R2A via using the notes mechanism (ch. III.17.2). Such notes are only added to the assigned R2A-items and nowhere stored centrally, which leads to an unstructured approach with no complete overview about present *verification criteria.* Further research could try to find a better solution, where easiness of usage and usability should play a central role.

- The R2A approach introduced here leaves one major field of problems concerning development of automotive systems and software untouched: Ch. I.2.3 indicates that buyers of cars can select hundreds of different options of their car individually, where also different options are connected with each other. This, however, implies that the different ECUs employed in a car can significantly vary between different cars and that in different cars individual *variants* of the ECUs must communicate with each other. As HW costs are significant constraints, different ECU variants also have different HW assemblies. Nevertheless, all different ECU *variants* and the different ECUs with their *variants* in interplay must fulfill their requirements, especially all *safety-related* issues. This together implies significant higher complexity than if all ECUs had only one fixed version. In *SysEng* and *SE* theory, management strategies for this complexity are called *variation management*. Hull et al. [HJD02; p.180-183] show that managing variation implies significant higher complexity concerning *variants*, *version management* and *change management* of requirements in connection with their *traceability* (see also [Si98], [BP06], [PR09; p.141f]) because the different *variants* must fulfill partially different requirements and the valid requirements must be – despite the *variation* – consistent to each other. In other words, *version baselines* and *change management* must in principle be performed and managed individually for each *variant* [HJD02; p.180-183]. On the other side, *variation management* issues also impose high influence on *SW architecture* and design theory (e.g., cf. [PBG04; ch. 10]), because decisions about strategies for handling the *variation* at the *variation points* ([PBG04; p.276], [Si98]; also cf. ch. III.16.1) significantly influence design[272]. As R2A also has its two major involvements in *REM* and design issues, R2A has potential to

[272] As an example, it must be determined whether a *variation* can be simply handled through a configuration parameter or whether the *variation* requires significantly more complex mechanisms to be integrated into design considerations (e.g., flexibility needs for a *variation point* can also lead to the decision that significant parts of the application must be created through the *abstract factory pattern* in order to allow activation of different component implementations according to the *variation* need).

improve *variation management*. However, to find suitable features, further research is needed.

IV.27 Conclusions

After you find the gold, there's still the job
of picking out your particular nuggets.
[BT04; p.147]

Now, finally, the reader has reached the end of this thesis. The author hopes that this thesis could provide valuable information to the reader so that he considered it worth, while reading it.

The main topic of interest has been *requirements traceability* between requirements and design artifacts in the development of *safety-critical systems*. As this thesis – hopefully – has shown, manifold factors must be considered, because the topic *traceability* is *cross-cutting* through research theories of *embedded systems development*, *systems engineering*, *software engineering*, *requirements engineering* and *management*, *design* theory and process standards for *safety-critical systems*. Despite all promising effects ascribed to *traceability* over the last two decades, the *traceability* concept did not broadly succeed in practice except for development organizations using process standards such as SPICE or CMMI, where, in most cases, *safety-critical systems* may be in the focus of development.

A reason may be the significantly higher effort and costs involved to make all requirements traceable throughout the complete development endeavor. Most probably, the effort and costs can only be justified, when issues of *safety* or *security* are involved. On the other side, costs will only be such a decisive factor if they are not outweighed by the benefits. This seems to be a core issue of the *traceability problem*.

Further, the thesis has shown that *requirements traceability* between requirements and design is especially *wicked* because this involves crossing a two-fold gap: First, different tools are used for *requirements specification* and design that make it necessary to bridge them. Secondly, a transition from requirements to design means a transition from a problem description to a solution description, involving a substantial, *non-linear gap* that is usually mentally bridged by designers but is difficult to cope with an ordinary link concept usually employed by *traceability* methods.

When analyzing different design theories, the author found out that design must rather be seen as a continuous decision process, where only parts of the decisions can be rationally describable by designers, but other extensive parts

arise by intuitive usage of *tacit knowledge*, cook-booky *heuristics*, and creativity. As the author has tried to show, exactly this "*tacit dimension*" [Po66] may be the major obstacle for valuable *traceability* information concerning design, because it infers the *non-linearity* in the relations between requirements and design and also hinders designers to make the transition process rationally explicable.

As a consequence, the author has invented a new tool solution called PROVEtech:R2A, aiming to narrow the twofold gap between requirements and design to a degree that *traceability* endeavors bring a real benefit to development.

To achieve this, PROVEtech:R2A has been developed to allow establishing *traceability* as a *by-product* of designers' usual development activities. Through this, additional benefit shall be provided to designers as an incentive to establish valuable *traceability* information. One of these benefits is that recorded *traceability* information can be directly used to improve communication and collaboration between designers. The tool further orients itself on the view of *design as a sequence of decisions*. Correspondingly, R2A allows recording traces of the decisions made. This starts with automatically recording traces of decisions about simple requirement allocation and design structure building (e.g., see ch. III.15) and continues by providing two different *decision models* allowing designers to document *rationale* information on more complex decisions.

Besides all these considerations, one further, very important, consideration has been that such a tool must also be integrated into a process landscape compatible with *process models* for *safety-critical systems*. This thesis has shown that this is in principle the case. As a further very important plus, the thesis identified major drawbacks of these *process models*, involving unnecessary redundancy concerning process transitions from requirements to design. The author could identify the underlying core idea that also design processes spark new "requirements" as consequences from decisions taken earlier. Once having identified this idea, the author could develop a *taxonomy of requiremental items*, where requirements originating from demands of the customer could be distinguished from *design constraints* originating from taken design decisions.

As it has further turned out, the first *decision model* could be used as a means to transform processes in a way that the original ideas of the *process models* were preserved, but unnecessary redundancy could be avoided. The *decision model*, allowing modeling conflict situations of requirements and then deriving consequences as new *design constraints*, can be seen as a new major extension of current *traceability* linking concepts by a more complex *traceability* concept that allows a better bridging of the gaps in a complex design decision process, leading to the *non-linear gap* between requirements and design. As a further major plus, the four major design theories introduced in this thesis could be adequately

weaved together with theories about *traceability* and *rationale management*, forming a tool set of supportive actions for designers.

Through significant research and development funding by the support program IUK-Bayern of the bavarian ministry of economics, it has been possible to develop PROVEtech:R2A to a solution now commercially available at the MBtech Group. First practical experiences at the MBtech Group are promising that the solution provides significant support for designers at their daily practical design work. In the meantime, through the coupling of the tool PROVEtech:TA (a solution of the MBtech Group for test automation) the usage context of PROVEtech:R2A has been even enlarged to a means for also bridging the gaps between a test specification and automatically executable testing code.