Supporting Customer-Supplier Relationships: Requirements Engineering and Quality Assurance

Dagstuhl Seminar Report (02361)

Barbara Paech, Fh IESE

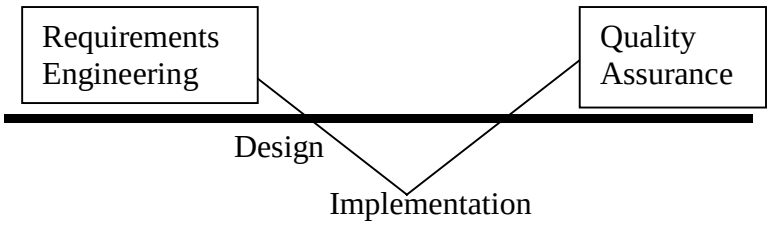
Seminar Organizers

Barbara Paech (Fh IESE) ,
David Parnas (McMasters University),
Jesse Poore (Univ. of Tennesse),
Dieter Rombach (Univ. of Kaiserslautern),
Rudolf van Megen (SQS Software Systems)

Table of Contents

1 Introduction

Increasingly, product engineers need to buy software components or to outsource part of their software development. For the cooperation with (external or internal) suppliers or for software procurement, the upper most level of the V-model, namely requirements engineering and quality assurance of the software product, are of utmost importance.



However, traditionally, requirements engineering and quality assurance are seen as separate activities carried out in quite different time frames during system development and through quite different people. Similarly, there is not much overlap in the corresponding research communities.

The purpose of this seminar was to bring together researchers and practitioners in the areas of requirements engineering and quality assurance such as inspection, testing and formal verification that are interested in a coherent support for software contracting. In the course of the seminar synergies and tradeoffs like the following have been discussed:

- How to support the communication between customer and supplier through elicitation and documentation of requirements?
- Which requirements documents can serve as the basis for software purchase?
- What quality assurance methods and products support the monitoring of the supplier?
- How to use quality assurance techniques for software product assessments?
- Can test models substitute a requirements specification as e.g. suggested by Extreme Programming (XP)?
- When to integrate quality assurance in the requirements engineering activities, e.g. what degree of stability is necessary for a re-

quirements specification to serve as a starting point for the specification of the tests, and when to involve quality assurance during requirements specification?
- How can different kinds of quality assurance products be derived from different kinds of requirements specifications, e.g. how to derive test cases from use cases?
- How to distribute effort between requirements specification and quality assurance, e.g. when should the customer require a formal specification or a requirements traceability model, when should the customer sacrifice requirements engineering activities for testing activities?
- How to combine different quality assurance techniques such as inspection, testing and formal verification for supplier monitoring?
- How to assure the quality of non-functional requirements?
- How can the experience gained in quality assurance, be used to improve the determination and documentation of requirements?

The discussions fostered the understanding of both communities and helped to stimulate technology transfer of existing methods into practice as well as research on integrated methods. By inviting both researchers and practitioners from different domains like telecommunication system, embedded systems, information systems or web applications, the identification of context factors for the success of integrated methods was supported.

The seminar was conducted as an open space. On the first day the participants collected the topics they wanted to discuss and present. Based on this, an agenda for the whole week was developed with plenary sessions and working sessions in parallel tracks. Over the course of the seminar the agenda was restructured based on the needs of the participants. Every day the participants assigned themselves to the parallel sessions. In the plenary sessions overview talks were given, summaries of the parallel tracks presented and discussed. This scheme ensured that the groups in each track were small enough for intensive discussions, but on the other hand every participant was informed about the overall results. In the final session the results were put together into a general picture of the pros and cons of the integration of requirements engineering and quality assurance.

This report collects the abstracts of the session chairs. In the last chapter a summary is provided which highlights the findings of the individual seminars.

2 Abstracts of the Plenary Talks

2.1 Plenary Talk: Formal Methods by Connie Heitmeyer, Naval Research Laboratory

To begin, a logician's definition of a formal method is contrasted with a computer scientist's definition. Then, a number of examples are presented to illustrate the benefits of applying formal methods to specifications and to other artifacts associated with software development. To illustrate the formal methods and techniques, the SCR (Software Cost Reduction) tabular notation and the SCR tools are introduced. Many different classes of formal methods are illustrated, including consistency checking, formality-based simulation, model checking, and mechanical theorem proving, by showing how they can be applied to real-world software systems. The talk concludes by showing how formal techniques may be used to automatically construct test cases from a formal specification. These test cases are designed to satisfy a selected coverage criterion.

2.2 Plenary Talk: Inspection by Filippo Lanubile, University of Bari

Software inspections are a software engineering best practice for improving software quality and reducing avoidable rework They were first introduced by Michael Fagan at IBM, based on experience from hardware engineering, with the main goal to find defects as close to their point of creation as possible. Since then, inspections have been applied in many variants by software industry and have been empirically investigated by many researchers.
This talk first introduces the characteristics that distinguish software inspections from testing and other types of peer reviews. It is shown how different goals for individual and team analysis result in different implementations of the inspection process. Then, reading techniques for individual analysis are presented, whose level of guidance ranges in prescriptiveness. Findings from empirical studies of scenario-based reading techniques are summarized and discussed. Finally, the talk presents some tools to support geographically-distributed inspection teams, in the context of global software development.

2.3 Plenary Talk: Requirements Engineering by Soren Lauesen, IT University

There are many ways to specify requirements. Some of them are easy to verify for testers and developers, others are easy to validate for the customer. (Validation means that the customer can check the requirements to see that if they are met, he will get what he needs.)

A major problem with traditional requirements is that they may be easy to verify, but they are almost impossible for the customer to validate.
Another problem is that they are not suited for tender processes or COTS because they go too far in the design-direction, so that some suppliers don't qualify although they have a system that actually would satisfy the customer needs. I will show ways to overcome these problems.

In my presentation I will show real-life examples of many kinds of requirements and their consequences for the project and the customer. I will also show how easy the various requirements are to verify during development and testing.

2.4 Plenary Talk: Testing by Dirk Meyerhoff, SQS

The NIST-study says that 60 billion $US are wasted every year on poor software quality.
In many industry software projects at least functional requirements are checked (tested). For non-functional requirements this is not the case.
Non-functional requirements of relevance are for example maintainability and portability of the code.
Today static analysis techniques do allow for checking many non-functional requirements. These techniques are very efficient in finding problem areas of the code.
Functional requirements are checked for example in functional and in integration testing.
There are methods around for test preparation, even tool supported.
A process requirement for the software development process today is traceability. In some industries like pharmaceutical, or in the safety critical domain this has been implemented for many years. We expect the other industries to follow this path. Traceability will be tool-supported completely in the near future.
Transparency of the testing process and its result is becoming more important. Up to date, transparency of the process can best be provided by putting all process information on the web, publishing it to management directly. This helps controlling project success.

3 Abstract of the Session Talks

3.1 User's Manual as a Requirements Specification by Daniel M. Berry, University of Waterloo

This talk argues that a user's manual makes an excellent, if not the best, software requirements specification. It discusses several lessons learned from experiences writing user's manuals as requirements specifications.

3.2 Requirements Engineering Lessons from House Building by Daniel M. Berry, University of Waterloo

Anyone who has built or remodeled a house and has developed or enhanced software must have noticed the similarity of these activities. This talk describes some lessons about requirements engineering I learned while being a customer in a house building and two house re-modelings. The biggest problem is to avoid very expensive requirements creep. The main lesson is the importance of the customer insisting on following a full requirements engineering process, including goal identification, requirements elicitation, analysis, and specification, and validation of the specification. A secondary lesson is that a customer has an important role in requirements engineering and he or she sometimes needs to learn that role.

3.3 Requirements Engineering Challenges in Multi-Functional Projects by Ole Fischer, Ericsson Eurolab Deutschland

In Multi-Functional Projects at Ericsson an overall solution consists of multiple components that are developed in-house by teams at different sites in Europe, or developed by external development partners or supplied as standard off-the-shelf products. For each component a Requirements Specification exists that is refined iteratively throughout the project phases, and used as key input to the processes design and test.
The following challenges are seen:

- a common notation for requirements suitable for different groups and purposes (technical, non-technical people, customers; validation, supplier selection, product design, test preparation, services design);
- synchronisation of the documents and people writing or using the various requirements specifications;
- determining when a requirement is "good enough" for the intended purpose;
- progress tracking and visibility of requirements fulfillment.

3.4 Traceability and Testing by Martins Gills, Riga Information Technology Institute and Thomas Zink, DaimlerChrysler

One of the problems that persists within many software development projects is their inability to see and to maintain a systematic relation between requirements, design items, tests, etc. Part of the problem is related to vaguely defined and maintained requirements. The same could be said about inconsistency between design and the code. Also, testing can only benefit from project settings where the up-to-date requirements are available, and all tests are based upon them. This all leads to the question of traceability. Traceability as a software development property has been explored for at least a decade, several requirements management and traceability tools have been developed. But still there exists a question of whether it is possible to introduce some traceability in projects that currently are not systematically organized.
In the workshop "Traceability and testing" Martin presented the results of an experiment that was carried out at Riga Information Technology Institute. A web-based traceability tool TraceIt was developed and applied in several projects. Each one was different in terms of the main task and the development process. For every project, a separate traceability model was defined.
Introduction of the tool helped to organize the requirements information. Also, completely all testing information was kept within the TraceIt database, including the whole problem tracking process. This was quite an unexpected result, and this experiment showed the possibility to integrate both traceability and problem/defect information.
Workshop's discussions touched the historical development of the traceability, as well as questions of the concept and notation of the proposed model. A more refined approach for the traceability model was proposed, and the questions may lead to further research and experiments.

3.5 Call for Consultants – A Role-Play by Martin Glinz, University of Zürich, and Andreas Spillner, Hochschule Bremen

We organized a plenary session on customer-supplier-relationships. We invented a role-play scenario where Martin played the role of a despaired software manager who had to deliver software to Andreas, who played the role of an equally despaired customer. The participants were divided according to their expertise into four groups, playing the roles of consulting companies. These were (1) ReQEx – the *requirements experts* company, (2) xQbT – the *extreme quality by testing* guys, (3) FMF – the *formal methods first* advocates, and (4) Fagan&Partners – the *inspectors* group. One participant volunteered to form a fifth company: Emotionics - the *satisfaction by emotion* expert.

In this role-play, Martin and Andreas decided to join forces to escape from their desperate situation and contacted the five consulting groups in order to get advice. Each consulting group had 20 minutes to prepare an improvement strategy containing both short-term and long-term elements. Then all the proposals were presented and discussed in an open hearing.
After a vivid discussion we took home a bunch of ideas about improving the satisfaction of both customer and supplier in software projects and the memory of a session that had been great fun for all participants.

3.6 Combination of Inspections and Testing Break-Off-Session/Workshop by Hans-Gerhard Gross and Maud Schlich, Fh IESE

The combination of inspections and testing may be seen from two different perspectives, empirically, and methodologically.

In the first case, we may either consider the optimization of testing through inspections, or the optimization of inspections through testing. It means we may be able to analyze defects that slipped through the inspections process and are caught during testing in order to change and improve our inspections process ("defect slipperage"). In fact, this has already been done and demonstrated. However, little is known of how to support or direct testing effort to determine so called "hot-spots" in the software that may be somehow identified during the inspections phase. Can we conduct experiments producing evidence that a forward prediction of this kind is actually feasible?

The discussion went along the following lines:

- We can optimize inspections through errors that are caught in testing and traced back to the inspected documents. This has already been done.
- Can inspections of requirements documents control or improve specification-based testing?
- Can inspections of code documents control or improve code-based testing?
- Inspections find many issues in a module, do we have to test this module less/more?
- Inspections find no issues in a module, do we have to test this module less/more?
- Can we use inspections to prioritize testing?
- Can we use inspection to predict the type of testing (testing method) with the highest probability of success.
- What is successful testing, what successful inspections? Can we exploit relations between inspections and testing in order to determine that? Can we use one to define the other?
- Can we define entry/exit criteria for requirements – inspections – testing?
- Does inspection actually reduce testing effort? Conduct an experiment to check the evidence, although the number of different affecting variables is very large in

an experiment. Therefore we need a good and stable overall development process; this leads to large scale and difficult experiments.
- Regression testing is a standard testing approach, but inspections are not really used to revisit changed systems. Can we define something like "regression inspection"?
- Which attributes (measures) do determine the relation between inspections and testing?

In the second case, we consider the two activities of performing inspections and deriving and executing tests in a joint method or process. The question is whether we can save quality assurance (QA) effort by performing static and dynamic analysis in parallel and perform a single debug step that aims at resolving the issues that the two analysis techniques identify.

The discussions went along the following lines:

- QA cannot only concentrate on the product, also processes are very important. Fixed processes are essential in empirical validation because there are too many variables that need to be controlled.
- What are we looking at: testing to find defects or testing in order to demonstrate high quality? The emphasis of inspections and testing changes in a process according to such questions.
- Inspection of requirements documents plus simulation of requirements; inspection of code plus test execution of code: simulation may be regarded as a different perspective for perspective-based reading. This represents the user's view. However, simulation is only feasible with formal specifications.
- Can we add dynamic perspectives to inspection? Inspection of UML state chart and interaction diagrams?
- Inspections find defects: does testing find the same defects or other defects?
- Can we test a system that is not inspected yet? Should we not wait for the issues from inspections to be resolved before we start testing (traditional two phase approach)?
- Can the testers act as inspectors in early development phases, sometimes they already act as requirements engineers?
- Can we develop a process to inspect requirements according to testability criteria? Develop a checklist for features of requirements that support testability.

3.7 Working session on Customer-Supplier Relationship by Frank Houdek, Daimler-Chrysler and Maud Schlich, Fh IESE

At the beginning of this working session the moderators presented various variations of customer-supplier-relationships. Especially it became obvious that there may be various levels of customer involvement ranging from "over the fence" to "all together".

In the subsequent discussion an initial process model for a development process involving both parties was refined and discussed. Major outcomes of the discussion are:

- A significant portion of activities should be performed together, so the respective roles are assigned to people on the customer's and the supplier's side. This contradicts the often seen current practice that is more like "over the fence".
- No sharp borderline between customer and supplier can be identified. There are some activities that are agreed to be only the supplier's responsibility, e.g. designing and coding, but even providing test-beds, project management or test planning should be carried out together.
- The sketched process (depicted by means of activities, work products, tools and roles) had a real hot spot in the earlier phases. Requirements (both on the level of user demands and on the design level) are essential to a successful project.

Unfortunately no working session member advocated on formal methods so this view point is underrepresented in the result of this working session.

3.8 Early test preparation by R. van Megen, SQS

Coming from a short presentation about statistics why/how many projects still fail today, be out of time or cost, the work group discussed the basis for test preparation: the requirements.

On behalf of that we analyzed why requirements are often not written; some reasons: domain experts are not available, modern methodology are very abstract (not for specialists, but for users). A solution to solve that problem could be: have a team of specialists (in methodology like UML etc.) who do the specification work, and a domain experts, who know what they want and can be used for interviews and reviews of results. All agreed that we need a more or less formal specification that can also be understood by non-specialists.

The discussion about early test preparation began with analysis of why it is not usual to have appropriate test cases etc.: time to market, starting too late, missing knowledge about benefits. Why do we need test cases such early, i.e. immediately after specification?

It was discussed that even after providing inspections to the specification, these are not without errors. Test preparation in that case can be used to find business errors within the specification: finding these early (before coding) would give a good benefit, because such errors would only cost a small part what it would cost to find the same error during acceptance test execution.

What do we need to convince management about the benefits? We need some numbers from real projects, a proof of concept, and a champion who promotes this.

3.9 Integrated refinement of non-functional requirements by Antje von Knethen, Fh IESE

The goal of this workshop was to discuss how non-functional requirements, such as maintainability or performance, could be described quantitatively under consideration of different viewpoints (i.e., customer, supplier) and different abstraction levels (i.e., system, software). Non-functional requirements have to be investigated together with functional requirements and architectural decisions because there are different types of dependencies between them. Non-functional requirements constrain, for example, architectural decisions, whereas the architecture realizes functional and non-functional requirements and constrains functional requirements.

At first, we discussed the term "non-functional requirement" that is somehow ambiguous because it is used for various system properties. Therefore, research should determine the concrete non-functional requirements investigated. Then, we discussed how to handle different types of non-functional requirements, especially maintainability. Maintainability seems to be more difficult to measure than other types of non-functional requirements. Further on, we discussed how different precise descriptions of non-functional requirements might look. On the one hand, a non-functional requirement can be described more precisely without determining a certain solution (e.g., change of type x shall be performed in time y). Such a description is especially important for the customer. To get a more precise description of maintainability, a better understanding of the types of changes that may occur in the domain is required. Therefore, typical changes have to be anticipated and documented. Questionable was whether it is reasonable to document a certain time in which a type of change shall be performed because precise estimation of time is difficult. On the other hand, a non-functional requirement can be described more precisely with determining a certain solution (e.g., to ease change, each requirement shall be traceable to design and code). Such a description supports the supplier in achieving a non-functional requirement described by the customer.

3.10 Sequence-Based Specification and Model-Based Testing by Stacy Prowell, University of Tennesse

We began by discussing sequence-based specification as a systematic means for eliciting a formal, functional specification of external behavior from a set of requirements. First a system boundary is constructed, and stimuli identified. Then one enumerates sequences of stimuli, in order by length, and indicates the appropriate next response for each. If work stalls or becomes unproductive, an explicit sequence ab-

straction, in the form of a function, is constructed to change the view during enumeration. A state machine is automatically derived from the complete enumeration. The functional specification provides insight into the testing problem: it helps identify the test boundary, all inputs, and the valid sequences of inputs. The state machine constructed from the enumeration is modified to account for testing goals and to capture known or postulated statistical characteristics of use of the system, and becomes a usage model from which random and non-random tests are generated. Finally, we discussed tools being developed to support these techniques.

3.11 The Role of Emotion, Values, and Beliefs in the Construction of Innovative Work Realities by Isabel Ramos, Telepac and Daniel M. Berry, University of Waterloo

Traditional approaches to requirements elicitation stress systematic and rational analysis and representation of organizational context and system requirements. This talk argues that (1) for an organization, a software system implements a shared vision of a future work reality and
that (2) understanding the emotions, feelings, values, beliefs, and interests that drive organizational human action is needed in order to invent the requirements of such a software system. This talk debunks some myths about how organizations transform themselves through the adoption of Information and Communication Technology; describes the concepts of emotion, feeling, value, and belief; and presents some constructionist guidelines for the process of eliciting requirements for a software system that helps an organization to fundamentally change its work patterns.

Joint work with Joao A. Carvalho

3.12 The role and the impact of UML and other OMG standards, as MDA, in software development by Gianna Reggio, University of Genova

In this group we discussed the possible roles and value of the various standards proposed by the OMG (Object management Group) to support the development of software systems. As expected, they are not the ultimate solution, but they can be useful in some cases and less in other ones.
A sensible role for UML may be as a standard communication means characterized by an intuitive semantics, that does not need to be made more precise any further. A formal semantics of some parts, however, is needed to develop tools for checking properties, or for simulation.
The discussion pointed out another problem of the UML; precisely, the fact that its diagrams, such as state charts, take a lot of time to be prepared and are difficult to modify when changes are made to the modeled system. The Agile-modeling and extreme programming communities propose a light-weight use of the UML that may be satisfactory, consisting of only a few diagrams at design level which are automatically modified when the code is modified.

Development methods may overcome this weakness of the UML also by requiring to produce UML models in a well-defined and structured way (e.g., by linking use cases to the classes realizing them), so that such a structure may facilitate their evolution.
For what concerns MDA (Model Driven Architecture) the opinion was that it does not seem to become a widely adopted standard, but that however, it incorporates some valuable aspects, which should be considered with attention:

- Abstraction (it requires to first produces abstract, platform independent models, to be then transformed in specialized models, one for each used platform)
- The importance for a method to consider explicitly
  - the domain of the applications to be developed (as telecommunication, finance, healthcare,...)
  - the use of middleware platforms (as CORBA, Enterprise Java Beans, WEB services,...).

At the end, it was remarked that MDA and UML seem to be based on good principles, but quite badly developed. Perhaps, if people with a good formal background developed them, the results may have been better.

3.13 Views and (In)Consistency in Model Based Development by Bernhard Schätz, TU München

In concurrent engineering, as for example used in the automation domain, domain-specific models are built and changed throughout development by mechanical, electrical and software engineers.
Therefore, partiality as well as inconsistency of specifications must be supported by such a domain/ view-oriented development process to further aspects like reuse and concurrent development of systems. To treat partiality/(in)consistency, an integrated product model unifying thoses different views and models can be applied. For syntactic consistency, integration must take place on the conceptual level syntactically relating the basic modeling concepts used by the engineers, esuring, e.g., consistency of interfaces or establishing traceability. Furthermore, integration on the semantic level is needed to support integration of different views or partial specifications (like scenarios and state-based descriptions), to check their completeness or consistency, or to transform one kind of view into another kind (e.g., test cases out of state-based descriptions).
In the workshop we presented the AutoFocus CASE approach, addressing these issues of concurrent development.

4 Summary

The outcome of this seminar is summarized in the following. The main contribution of the seminar was a better understanding

- between the different communities: requirements engineering, formal methods, inspection and testing
- of practical problems of customer/supplier relationships (C/SR) and
- of the practical and scientific problems of integrating requirements engineering (RE) and quality assurance (QA).

In the closing session many participants emphasized that they will base future work on this understanding and work in particular on the synergies of these techniques. All participants agreed that an integrated RE and QA process would be of utmost practical value. In the following we first highlight the findings wrt. C/SR. Then we sketch an integrated RE and QA process which evolved during the course of the seminar. Finally, we list the most important research questions identified in the discussions.

4.1 Customer and Supplier

Many presentations and discussions of the seminar dealt with the integration of RE and QA without regard of the issue of C/SR. Thus, during the seminar the question arose whether there is anything specific to integration of RE and QA for C/SR. In particular, the participants from industry felt that C/SR issues are very important. Thus, the workshop by Frank Houdek and Maud Schlich aimed at identifying these specific issues (see also abstract 3.7).

First of all, it was agreed that C/SR are quite complex, but also quite diverse. Customers are interested to buy or procure software. Suppliers produce software from scratch or by adapting a product. The intensity of the relationship varies between joint development work and very formal settings where only management or lawyers of both sides interact.

Second, it was agreed that both, customers and suppliers, need to be involved in RE and QA activities. Thus, both of them have responsibility for the success of these activities. Also, it is equally important for them that this process is successful.

So, the following issues for RE and QS in C/SR-situation were identified:

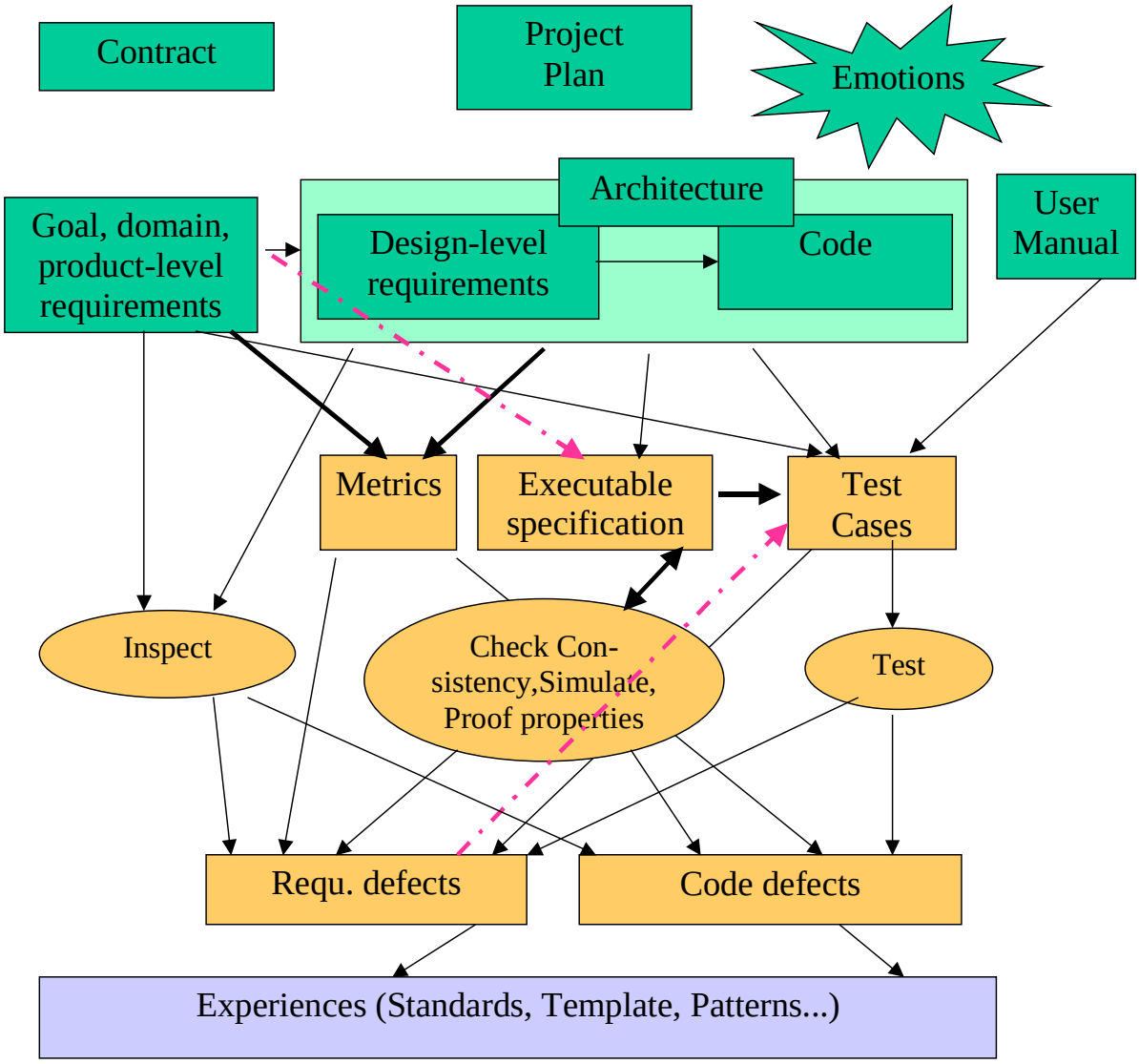
- The integrated RE and QA process must accommodate both, customer and suppliers, that means people with quite diverse backgrounds and interests. In addition,

C/SR-situations typically induce issues of geographical distribution, cultural and organization differences on the process.

- For C/SR projects an integrated RE & QA process is especially important, because these are exactly the activities where customer and supplier interact.

4.2 Integrated Process

The following picture shows the outline of an integrated RE and QA process.
Boxes represent artifacts, ovals represent activities. Arrows represent dependencies.



Roles involved in this process are not shown. It depends on the situation which role is involved in which activity.
In the upper half of the picture the RE and Design artifacts are shown. One important distinction here – which is too often not recognized when the RE and the formal method community interact – is that there are two kinds of requirements:

- Goal, domain and product-level requirements describe the effects of the software system in its environment as well as the features of the software system that induce these effects.
- Design-level requirements detail the product-level requirements such that developers exactly know what to build.

The first kind of requirements is the basis for the discussions with the users and customers, the second kind is the basis for the developers. Customers must approve both, since only they can assure that design-level requirements detail the other kinds.
The arrows between these artifacts indicate activities to develop code from the requirements. To ease change and reuse, it is important that the resulting dependencies between the different artifacts are traced.
In the lower half of the picture the QA artifacts and activities are shown. The outcome of the activities is requirements and code defects. The arrows indicate the inputs and outputs of these activities as well as transformations between the artifacts (where the activities are not explicitly shown). A solid arrow indicates that there are powerful automation tools to support the activity delivering the result.
One important point here is that all QA activities can be used to identify requirements defects and code defects[1].
So inspections can be used to find requirements defects and code defects (and of course also design defects, not shown here).
Similarly, metrics can be applied to requirements artifacts and code artifacts. However, typically they will only be applied to design-level requirements, because at this level there is an emphasis on formal properties.
The development of test cases (out of the product-level requirements or the design-level requirements) helps to identify requirements defects. . If the user manual is developed early, it is also possible to identify test cases from the user manual.
Powerful automatic analysis of the requirements documents is only possible, if requirements are transformed into an executable specification. There is evidence from several industrial applications that this transformation is worth the effort when starting from design-level requirements. It is an open question whether an executable specification can be directly developed from product-level requirements (therefore the arrow is highlighted). The problem is that product-level requirements are often not detailed enough and not stable enough. If there is a formal specification of the requirements as well as of the realization (code), then the analysis tools can also reveal code defects. Furthermore, it is possible to automatically derive test cases from executable requirements specifications. In this case the executable specification is used as an oracle for the outcome of the test cases
Defects detected during testing are either code defects (when the code does not conform to the specified requirements and the specified requirements are correct) or requirements defects (when the code does not conform to the specified requirements, but to the customer's needs). One can use the evidence of requirements defects identified early to tailor the set of test cases to specific problems expected in the code. However, so far this step has not been exploredFinally, the analysis of the defects helps to make experiences explicit. These can be captured in standards, templates and patterns and serve to improve following projects.

[1] Note that for the purpose of this discussion it is not necessary to distinguish between failures and defects.

Altogether, this picture reveals many possibilities for QA of RE through inspection, metrics, formal methods and testing. There was general agreement that all possibilities are worthwhile in some contexts. The major open questions concern the selection of the right technique depending on the context and effort savings through synergies of these techniques. These questions are detailed in the following section.

4.3 Research Issues

The seminar lead to a common understanding of the possibilities of integrating RE and QA techniques. At the same time it revealed many open issues wrt. the problems and benefits of such an integration - in particular, given the cost and time restrictions of industrial projects

- **Documentation effort**: In practice, it is not possible to specify requirements completely. However, QA can only work on the basis of requirements documents. Thus, one needs guidance on where to focus the requirements engineering effort. Here experiences from previous QA-activities (e.g. defects classes) can help.

- **Technique selection**: It is not possible in general to execute all QA activities on all artifacts. Thus, one needs guidance when to select which technique. So for example, it would be nice to guide the selection depending on the application domain (e.g. formal specification seems to be more suitable for embedded systems than for information systems) or on the project context (e.g. size of the project, people involved) or on the classes of defects investigated. Clearly, the technique has to be understandable by the participants. One important prerequisite for creating such guidance is data about the effort and defect detection rates of the different techniques.

- **Synergies between QA-techniques**:
  There are three ways to achieve synergies

  - Through substitution: this means that one QA-activity is carried out instead of another one. So, for example one could substitute a certain set of tests through inspection. (e.g. specific defects only through inspection).
  - Through preparation: this means that one QA-activity is used to alleviate another technique. So, for example, automatic derivation of test cases from formal specification eases the testing process.
  - Through leveraging of QA results: This means that analysis of the defects found through one technique is used to guide the application of another technique in the same project. For example, defect rates from requirements inspections can

be used to focus testing.

So far, in industry mostly testing is used and the possibilities of inspections,metrics and formal methods are not sufficiently explored. Thus, the preparation or leveraging of one activity through another one is not explored. The participants agreed that there is a high potential for improvement.

•Synergies between RE and QA:
There are four ways to achieve synergies

- Through substitution: this means that a RE artifact or activity is substituted by a QA artifact or activity. For example, instead of requirements documents, test cases are written. This is one measure now advocated in Extreme Programming. In this case the test cases serve as the basis for the development. Through preparation: this means that during RE activities are performed and artifacts created that are especially suited to perform QA activities. For example, use cases ease the derivation of test cases. Through leveraging of QA results: This means that analysis of the defects found through QA is used to guide the RE activities of the next project. For example, defect rates from requirements inspections can be used to improve the elicitation or specification activities. Through mixed teams: This means that QA people are involved early in RE activities, e.g. as inspectors. Another example is to let requirements engineers write the test cases so that they get to know the level of details needed by the testers. Again, so far in industry these synergies are rarely explored and there is a high potential for improvement.