The Three Dimensions of Requirements Engineering[+]

Klaus Pohl

Informatik V, RWTH-Aachen, Ahornstr. 55, 5100 Aachen
pohl@informatik.rwth-aachen.de

Abstract. Requirements engineering (RE) is perceived as an area of growing importance. Due to the increasing effort spent for research in this area many contributions to solve different problems within RE exist. The purpose of this paper is to identify the main goals to be reached during the requirements engineering process in order to develop a framework for RE. This framework consists of the three dimensions:

- the specification dimension
- the representation dimension
- the agreement dimension

Looking at the RE research using this framework, the different approaches can be classified and therefore their interrelationships become much clearer. Additionally the framework offers a first step towards a common understanding of RE.

1 Introduction

There is general agreement among software engineers and researchers that an early stage of the software development life cycle called *requirements engineering* exists. Furthermore requirements engineering (RE) is perceived as an area of growing importance. Due to the increasing effort spent for research in this area many contributions to solve different problems within RE exist. The purpose of this paper is to identify the main goals to be reached during the requirements engineering process in order to develop a framework for RE, the three dimensions of requirements engineering. Looking at the RE research using this framework the different approaches can be classified and therefore their interrelationships become much clearer. Additionally the framework offers a first step towards a common understanding of RE.

A first impression of the research subsumed under the term requirements engineering can be gained by looking at the topics (cf. table 1) of the first major international meeting on RE (International Symposium on RE 1993).

[+] This work was supported by ESPRIT Basic Research Action 6353 (NATURE) which is concerned with Novel Approaches to Theories Underlying Requirements Engineering and by the state Nordrhein-Westfalen, Germany.

- formal representation schemes and RE modelling
- descriptions of the RE process
- tools and environments to support RE
- requirements engineering methods;
- requirements analysis and validation;
- requirements elicitation, acquisition and formalization
- establishing traceability to requirements
- reuse and adaptation of requirements;
- intersections with AI, domain modelling and analysis
- intersections with computer-human-interaction and cognitive science;
- intersections with group and cooperative work
- intersections with systems engineering

Tab. 1. Topics of the First International Symposium on Requirements Engineering.

Even to understand the topics, the question *"What is requirements engineering?"* must be answered first. For example, before talking about tools and environments for supporting RE, a clear idea of the aim of RE (e.g., building a requirement specification as defined in IEEE STD 830–1984) and the problems to deal with, must be available. Also before looking at the intersections between RE and other research areas, a common understanding of RE must be gained first. But the topics illustrate, that RE is an interdisciplinary research area.

To get a more detailed view of the ongoing research, we give a brief overview of the RE literature. First, we focus on the research dealing with the *detection of requirements*. This includes the problems of requirements elicitation and capture as well as the problems of validation and verification of requirements (e.g., [11], [29], [30], [84], [64], [87]). To *represent requirements* formal specification languages (e.g., Z [92], VDM [8], [47], PAISLey [100]) and knowledge representation languages (e.g., RML [41], ERAE [45], TELOS [76], [55]) were proposed. They offer the advantage of automatic reasoning (e.g., [9], [73], [65], [62], [96]) but applying them to RE is not straight forward (e.g., [4], [46], [3], [28]). Moreover, they must be generated from, and integrated with, informal RE specifications (e.g., [41], [6], [57], [38], [34], [74], [59]).

During the RE process *different views* of the system to be built exist. Some work concerns view integration and viewpoint resolution (e.g., [63], [64], [31]). Others suggest to focus on the social and cognitive aspects of RE (e.g., [90], [40]), thus gaining a better specification. Methods of AI are also used to support the RE process (e.g., [1], [5], [65], [69], [58], [94], [86], [68]). The advantages of reusing specification for economical reasons as well as for avoiding errors were lined out (e.g., [7], [36], [66], [94], [67], [22], [16], [68]). Other research focuses on the *RE process* (e.g., [43], [17], [44], [53], [18], [80]). It was recognized, that the RE process must be traceable (e.g., [33]) and understandable. Therefore the recording of design rationale (e.g., [83], [88], [53]) and the integration of argumentation concepts into the RE area are proposed (e.g., [15], [85]). Generally speaking it can be said that methodologies for supporting RE that based on different representation formalisms exist, but do not tell the requirements engineer very clearly how to proceed (e.g., ER [13], SA [95], [98], JSD [12], object–oriented analysis [93], [79], [75], [14], conceptual modelling [77], F-ORM [22], PSL/PSA [89], SREM

[2], ASPIS [84], KBSA [19]). Also some classification of the methods were proposed (e.g., [101], [21]).

Even with the coarse classification of the literature made above the *main goals* and the *real problems* of RE are not visible. A first step into getting to the heart of RE is to distinguish between two kinds of problems:

- original requirements engineering problems and
- problems caused by approaches which try to solve the original problems.

Making the original RE problems and the goals to be reached during the process explicit provides the basis for classifying the research of the RE area and for guiding a RE process. In section 2, we consider the RE process at an abstract level. Looking at the *initial input* and the *desired output*, three main characteristics can be identified. These features lead to the three dimensions of requirements engineering which are the main contribution of this paper (section 3). In section 4 we look at the RE process within the three dimensions. Thus the goals to be reached by the RE process are recognized and the problems which occur during the process can be classified. A classification of computer support for requirements engineering is made in section 5. In section 6 our contributions are summarized.

2 The Requirements Engineering Process

McMenamin and Palmer [71] suggest to distinguish between the essence of a system and its incarnation. The essence is defined by all essential activities and data stores whereas the sum of people, phones, computer systems, offices, typewriters, pencils, rubbers and so forth that are used to implement the system are the incarnation (cf. [71], [98]). To get a clear idea of the essence of a system they assume that the system can be implemented using perfect internal technology. This assumption makes it easier to concentrate on the essence of the system instead of getting influenced by unnecessary side aspects. Therefore the essence of a system has to be clearly defined first; aspects which come from the use of imperfect technology are not considered. After this, the so gained *essential model* of the system is extended by actions and data stores based on the use of imperfect technology. In the following we use this approach to look at the RE process.

Looking at a process (e.g., the requirements engineering process) on a abstract level, its essence is transforming an input to a desired output. Assuming that the RE process can make use of perfect technology (perfect tools, no social conflicts, no cognitive limitations etc.) it is insignificant how the transformation is achieved. Let us focus on the output of the RE process first.

2.1 The Desired Output

There is no doubt, that at the end of requirements engineering a specification of the system to be built (at least for the current version of the system) must exist. This specification serves as a basis for the next phase within the software life cycle. Thus, as a first characteristic of the output of the RE process, a specification of the system can be identified. We don't focus on the details of the final specification at this point. It is

enough to keep in mind that the **complete specification**, as expected, is the basic result of the RE process.

If the system specification is expressed using e.g. natural language, different people may understand the same specification in different ways. This may lead to unexpected designs and implementations. To avoid different interpretation of a specification, more and more people suggest to use a formal language for representing the specification of the system. Additionally a formal language offers the possibility of reasoning support. So the result of the RE process should be expressed using a **formal language**.

But it is not enough to produce a specification expressed in a formal language. Assume that a functionality called *work control* is well defined and that there exists no problem in mapping this part of the specification into a design and an implementation later on. But within the requirements engineering team only a few people agree on this functionality promoted by the people which are responsible for cost control. The representatives of the users don't like this functionality at all. If no common agreement is reached during the RE phase, the problems caused by this must be solved later on. As experience has shown, more effort is needed to correct errors in the later phases of the software life cycle [11]. To avoid expensive error corrections all people involved in the RE process should end up on a **common agreement** on the final specification.

Summarizing the main characteristics of the *desired output* of the RE process are a complete system specification expressed using a formal language on which all people involved agree.

2.2 The Initial Input of the Process

At the beginning of the RE process the knowledge about the system is coarse. Some features of the system are obvious, whereas about others only vague imaginations exist. Therefore the understanding of the system and the specification which can be gained out of it is very **opaque**. Since people involved in the RE process have various roles (e.g., user representative, system developer, maintenance staff, financial officer) and different skills and knowledge, each of them has his own understanding of the system to be built. Especially at the beginning of the RE process many different visions of the system exist. They may have something in common, but this is not necessarily the case. Hence at the beginning of the RE process many **personal views** on the system exist and no common representation format is used to express the expectations. Each stakeholder uses his preferred representation format for expressing his personal view of the system. Some of them may just think about the system (representing the knowledge in brain-structures), others may make notes using natural language, or may draw pictures or graphics. Hence mainly **informal representations** are used at the beginning of the RE process.

Summarizing, at the beginning of the RE process opaque personal views of the system exist which are recorded using informal languages.

3 The Three Dimensions of Requirements Engineering

Looking at the brief description of the *initial input* and the *desired output*, three main goals of the RE process can be identified:

- improving an opaque system comprehension into a complete system specification;
- transforming informal knowledge into formal representations;
- gaining a common agreement on the specification out of the personal views;

Out of these goals, three dimensions of RE can be gained: **specification, representation** and **agreement** dimension. Within the three dimensions, the *initial input*, as well as the *desired output* can be characterized. This is shown in figure 1, where the *initial input* is characterized by *personal views*, *opaque system specification* and *informal representation* and the *desired output* by *common agreement*, *complete system specification* and *formal representation*. In the following the three dimensions are described.

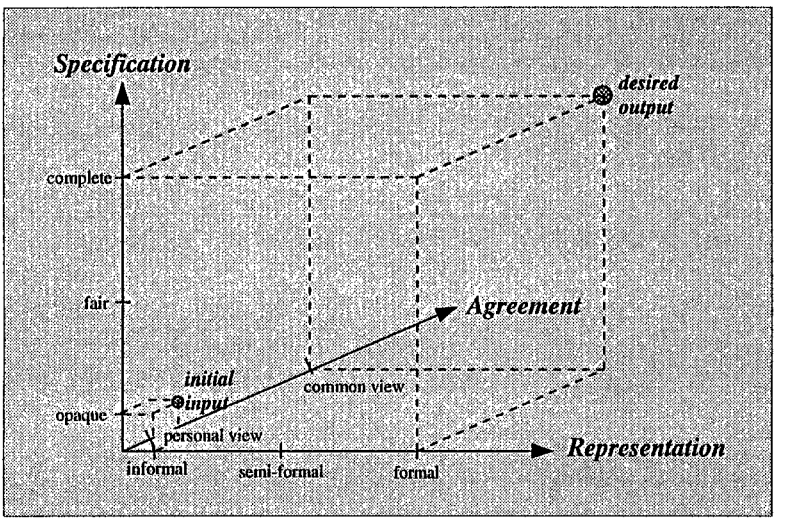

Fig. 1. The Three Dimensions of Requirements Engineering.

3.1 The Specification Dimension

The *specification* dimension deals with the degree of requirements understanding at a given time. At the beginning of the RE process the specification of the system and its environment is more or less opaque. This goes along with the vague imagination of the system at the early stage of the RE process. Focusing on this dimension, the aim of RE is to transform the operational need into a complete system specification through an iterative process of definition and validation (e.g., analysis, trade-off-studies, prototyping).

Several standards and guidelines describe how the final requirements specification should look like (e.g., IEEE Std. 830 [49], British Standard 6719, European Space Agency ESA PSS-05-0 [72]). In the following we briefly describe the properties a requirements specification should have. A more detailed description of the attributes of a requirements specification and an overview of existing standards and guidelines can be found in [25].

First of all, a requirement specification is supposed to state what a system should do and not how (cf. [20]). Additionally, the specification must be unambiguous, complete, verifiable, consistent, modifiable, traceable and usable during operations and maintenance (cf. [49] for a detailed description).

Secondly a differentiation between two kinds of requirements can be made:

- functional requirements
- non-functional requirements

The functional requirements specify what the software must do. According to IEEE 830, non-functional requirements can be further divided into *performance*, *design constraints*, *external interface* and *quality attributes*. Performance requirements deal with the execution time and computational accuracy. Design constraints are predefined designs imposed on the software development by the customer. External interface requirements define everything outside the subject of the system the software must deal with (e.g., constraints from other standards, hardware or people). With quality attributes the quality of the software to be reached is defined (cf. [61] for examples of quality attributes).

Beside this classification of requirements a distinction between *vital* requirements and *desirable* requirements should be made (cf. British Standard 6719 [48]). *Vital* requirements must be completely accomplished by the system, whereas *desirable* requirements may be relaxed and need not be met within the stated limits. Some standards propose to include *costs* and *schedule* information in the requirements specification (e.g., British Standard 6719) whereas other separate them from requirements engineering (e.g., IEEE Statement of Work). Additionally many proposals for validation and verification of system specification were made (e.g., [11], [99], [10], [35], [25], [64]).

Summarizing the first main goal of RE, as identified by many researchers, is to built a requirements specification, according to the standard and/or guideline used. The degree of the specification (opaque to complete) is captured by the *specification* dimension.

3.2 The Representation Dimension

The *representation* dimension copes with the different representations (informal and formal languages, graphics, sounds etc.) used for expressing knowledge about the system. Within RE there are three categories of representations. The first category includes all informal representations, such as arbitrary graphics, natural language, descriptions by examples, sounds and animations. The second category subsumes the semi-formal languages such as SA-diagrams, ER-diagrams, SADT etc. The third category covers formal languages such as specification languages (e.g., VDM [8], Z [92]) or knowledge representation languages (e.g. ERAE [45], Telos [76]).

Each of these categories offers some unique advantages. Informal representations like natural language are user-oriented. They are well known, since they are used in daily life. The expressive power offered by *informal representation* is very high and all kinds of requirements freedom are available (e.g., ambiguity, inconsistency, contradictory; cf. [4], [28] for more detail). *Semi-formal representations* like SA or ER diagrams are based on a structured graphical visualization of the system. The representations are clear and provide a good overview of the system ("one picture says more than a thousand words"). Additionally they are widely used within industry as a quasi-standard. In contrast to informal representation the semi-formal representation come with formally defined semantics, which could be used for reasoning. But the formal defined semantic of semi-formal languages is very poor, so still most of the represented knowledge has no formal meaning. *Formal representation languages* have a richer well defined semantic.

Therefore reasoning about most of the represented knowledge is possible. Even code can be (partially) automatically generated out of a them. So formal representation languages are more system oriented.

The use of a particular representation language has two main reasons. The first reason for using a special language is simply personal preference. Due to the advantages of each representation class, different people prefer different representations. For example the system user may like natural language, whereas the system specialist may prefers formal representation. The second reason for using a particular language is the current state of the specification. At the beginning of the RE process normally informal languages are used, whereas at the end specifications are often represented using formal languages. Hence the RE process must assure, that out of the informal requirements a formal specification is achieved. Since different representation languages are used within the RE process in parallel, they must additionally be kept consistent. Suppose that a requirement was expressed using natural language by the customer. Out of this requirement, a formal specification was built by the system specialist. If, for example, the informal requirement is revised, it must be assured that the formal representation of the specification is modified accordingly.

The representation language used does not imply if a specification is vague or precise. Hence the *representation* dimension is orthogonal to the *specification* dimension. A vague imagination of the system can be expressed using a natural language, but also using a formal representation language. Also concrete (formally defined) ideas can obviously be represented using a formal representation language, but they can also be exactly described using natural language (e.g., lawyers try to do so). Looking at the specification `'the age of Carl is 10 years'` and on a formal specification, e.g., using first order logic, `'age (Carl, 10, years)'` no difference can be recognized. Whereas the vague specification *'Carl is young'* is also vague if it is represented in first order logic `'young (Carl)'`. Hence the difference between the two specifications, vague versus precise, remains the same, independent of the representation language used.

Summarizing, during the RE process different representation languages are used. At the beginning of the process the knowledge about the system is expressed using informal representations, whereas at the end of RE the specification must also be formally represented.

The second main goal of the RE process is threefold. First, different representations must be offered. Second, the transformation between the representations (e.g., informal to semi-formal, informal to formal) must be supported. Third, the different representations must be kept consistent.

3.3 The Agreement Dimension

The third dimension deals with the degree of agreement reached on a specification. At the beginning of the RE process each person involved has its own personal view of the system. Of course few requirements may be shared among the team, but many requirements exist only within personal views of the people, e.g., stemming from the various roles the people have (system analyst, manager, user, developer etc.). In the following the expression *common system specification* is used for the system specification on which the RE team has agreed.

The RE process tries to increase the *common system specification*. But still requirements exist on which none or only partial agreement was reached. Let's focus on a simple example. Assume, that a library system is currently specified by an RE team. An agreement was gained, that data about the real world object *'book'* must be stored. Each stakeholder defines (from his point of view) the properties of the object *'book'*. The user defines the properties *'book-title, author-name, year'* using natural language. The system analyst additionally defines the properties *'book-id, status-of-book (loaned | available | defect | stolen | ordered)'* using a formal representation language and the specification of the librarian consists of the properties *'names of authors, keywords, classification-no., location,...'*. Therefore, the need for storing information about the object book belongs to the *common system specification*, whereas at the same time the properties to be stored are pertained by the personal views. In addition the coexistent specifications are expressed using different representation languages.

Different views of the same system have positive effects on the RE process. First, they provide a good basis for requirements elicitation (e.g., [64]). Second, the examination of the differences resulting from them can be used as a way of assisting in the early validation of requirements. Hence having different views enables the team to detect additional requirements. Moreover, if contrasting requirements were stated, conflicts can be detected and therefore become explicit.

It is important to recognize that the integration of different views at the representation level (e.g., integrating formally represented views into a comprehensive view) and the agreement on the integrated view among the people involved in the process are two separate actions. The fact, that a view was formally integrated has nothing to do with the agreement which exists on this view. A detected conflict must be solved through communication among people. Of course this communication has the aim of attaining an agreement (solving the conflict), but as a side effect additional unknown arguments (requirements) could be detected (cf. [15], [85]). Support for conflict resolution can be found in the area of computer supported cooperative work (e.g., [97], [42], [15]). Additionally support can be offered through different representations, e.g., by providing informal knowledge for explanation of formal representations, by offering graphical representation for overview of the system, or by automated detection of differences between formal specifications.

Summarizing, the *agreement* dimension is as important as the *representation* and *specification* dimension. We have pointed out that several specifications expressed in different representation formats may exist at the same time. Further we showed, that the coexistence of different views has positive effects on the RE process. Thus, allowing different views and supporting the evolution form the personal views to a common agreement on the final specification is the third main goal of RE.

4 The RE Process within the Three Dimensions

Looking at the RE process within the three dimension, the aim of the RE process can be stated as getting from the *initial input* to the *desired output*. So the trace of the RE process is an arbitrary curve within the cube spanned by the three dimensions (cf. figure 2).

The *initial input* is characterized as opaque personal views of the system represented using informal languages, whereas the *desired output* is characterized as formally represented, complete system specification on which agreement was gained (cf. section 2 for details). The main goals of the RE process can be sketched as follow (cf. section 3 for details):

- develop a complete system specification out of a opaque system understanding
- providing integrated representations and support the transformation between them
- accomplish a common agreement on the final specification allowing personal views.

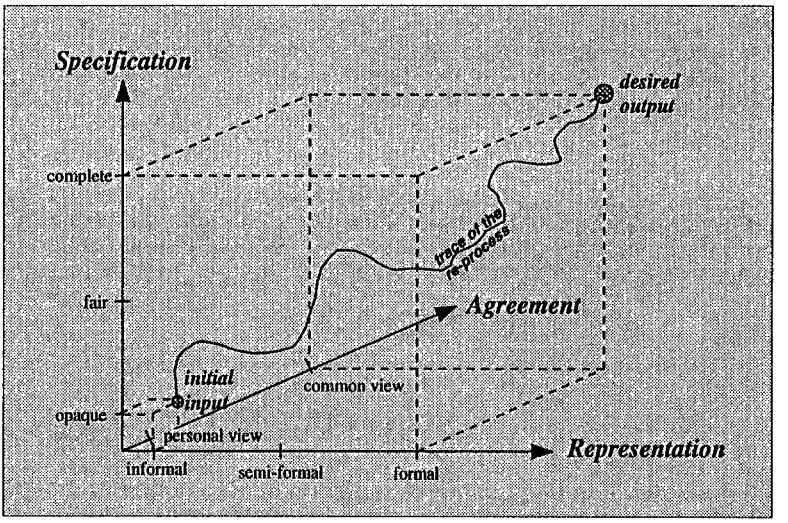

Fig. 2. The RE process within the three dimensions.

Getting from the *initial input* to the *desired output* is an interactive process consisting of different actions. An action can of course affect more than one dimension; improving one dimension often lead to a step back in another dimension.

The transformation of informally represented knowledge into a formal specification is a good example of an action (transformation step) affecting all three dimensions. An improvement within the *representation* dimension is gained, since informal knowledge is transformed into a formal representation. But during the formalization a contradiction within the formal representation may be detected by automated reasoning. This leads to a communication within the RE team to gain an agreement about the conflict (improvement of the *agreement* dimension), but additionally as a side effect a new requirement was noticed. The integration of the requirement as well as the agreement about the contradiction lead to an improvement of the *specification* dimension. The original action, transforming informally represented knowledge into a formal representation causes the execution of other actions and therefore affects all three dimensions.

This view of the RE process can not only be applied for the overall system specification. Also the evolution of each individual requirement can be covered by the three dimensions. A specific requirement can be represented within different specifications (personal views), each of these views can be represented using different representations

and the specific requirement can be well understood by a part of the RE team, whereas the other part may have still only vague ideas about it. Hence, the three dimensions and the view of the RE process as an interactive transformation process consisting of actions also helps to understand the RE process at a microscopic level.

Since the RE process takes place in the *'normal'* world, the result of the RE process is influenced by various factors. All of them can have both positive and negative influence on the RE process. We identified five main factors influencing the RE process:

- *Methods and Methodologies:* The process is influenced by the methods and methodologies used for guiding the process. Of course using another method during the process can lead to different results, since they focus on different things. If e.g., structured analysis was used, the final formal specification can be totally different in comparison to a specification gained by using object oriented analysis.
- *Tools:* The final specification depends on the tools used during the process. If e.g., a reasoning tool for formal representations was used, inconsistencies can be detected, which otherwise could be still in the final specification.
- *Social Aspects:* The social environment of the RE team affects their working results. If e.g., there are conflicts between the different persons, they work more ineffectively; if the people feel fine at work, the output of the work is much better.
- *Cognitive Skills:* People have different cognitive skills. If very bright people are involved in the RE process, the final specification is usually better.
- *Economical constraints:* Economical constraints limit the resources (people, money, tools, etc.) which can be used during the RE process. It's not always true, that with more resources a better result can be gained, but if the available resources are low a certain limit, the output of the process gets less quality.

Discussing these influences in detail is beyond the scope of this paper. But it should be clear, that these are not unique to the RE process. Most of the existing processes, e.g., the production processes, are influenced by these factors.

For these reasons it is necessary to distinguish between problems which are *original* RE problems and those problems which are caused by one of the five influences mentioned above. The problem of keeping SA-diagrams, ER-diagrams as well as the data-dictionary consistent is an example for a problem caused by one of the five influences mentioned above (methods). Another example is the problem of motivating people (social aspects). *Original* RE problems are all the problems which are caused by the three dimensions. Hence requirements capture, elicitation of requirements, transformations between different representations, integration of different views are examples for *original* RE problems.

5 Computer Support for Requirements Engineering

Traditional CAD/CASE systems have often neglected that computer support for any engineering activity must be based on an understanding of the process. In this section we use the framework presented is this paper to characterize the kinds of computer support that could be useful for RE. We distinguish between computer support for improving the result of the RE process in one of the three RE dimensions, for guiding the process of RE and for easing the influences on the process.

5.1 Specification Dimension

Getting to a deeper understanding of the system and therefore to a better system specification can mainly be supported by three different kind of approaches.

First, generic knowledge (domain knowledge) can be used to improve the specification of the system. There exist generic knowledge which is valid within a particular domain, e.g., banking systems, but also domain knowledge which is valid within many domains, e.g., stock control. It was demonstrated by many research contributions that the use of *domain knowledge* has positive effects on the RE task (e.g., [5], [1], [36], [84], [65], [86], [68], [51]).

Second, the reuse of specific knowledge can lead to a better system specification. Reusing requirements specification of already existing systems leads to better insight of the systems behavior and avoids misspecifications. If the requirements specification of an existing system is not available it can be gained through reverse engineering (e.g., [36], [7], [56], [16]). For both using generic and specific knowledge during the RE process, support for retrieving suitable knowledge must be offered, e.g. using similarity based search approaches (e.g., [39], [16], [91]).

Third, the current specification of the system can be improved by applying techniques for requirements validation. Validating a software specification was characterized by Boehm as *"Am I building the right product"* [11]. During the validation errors and gaps within the current specification can be detected. This leads to a correct specification of already known requirements (correcting the errors) or the detection of new requirements (filling the gaps, e.g.,[11], [30], [87]).

5.2 Representation Dimension

Within the *representation* dimensions the support which can be offered is twofold.

First, due to certain strengths and weaknesses of the different representation formats the use of informal representation (e.g., natural language, graphics), semi-formal (e.g., ER, SA) and formal representation languages (e.g., VDM, Z, TELOS, ERAE) must be possible. For keeping the knowledge, expressed in the different representation formats, consistent, the different representations must be integrated. The relationship between formal and informal representations is much less understood. But hypertext offers a opportunity to structure informal requirements and to relate them to formal approaches (e.g., [60], [15], [85], [59]).

Second, the transformation between informal, semi-formal and formal representations must be supported. On one side, support for automated derivation of formal specifications out of informal descriptions has to be offered (e.g., [41], [57], [74], [34], [87]). On the other side, the transformation process must be supported by offering requirements freedom within the formal representation language. Formally specifications have traditionally been expected to be complete, consistent and unambiguous. However, during the initial definition and revision of formal requirements, they are typically fragmented, contradictory, incomplete, inconsistent and ambiguous. Furthermore the expressions may include various levels of abstractions (concrete, examples, general properties etc.). Since formal requirements are built out of non-formal, the acquisition process must allow many freedoms (cf. [3], [28], [51]).

5.3 Agreement Dimension

There was not much research done in supporting the *agreement* dimension within the area of requirements engineering. Nevertheless, three kinds of essential assistance for the *agreement* dimension can be identified.

First, as pointed out in section 3.3, different views of the system exist during the RE process. Even within formal languages it must be possible, that different views and different specifications exist in parallel. Also the different views and specifications must be maintained during the RE process.

Second, support for detecting dissimilarities and inconsistencies between the different views must be offered. Additionally the integration of different views must be supported by appropriated tools. Contradictions for example can be made explicit through automatic reasoning and of course the work out of a solution can be supported. Viewpoint resolution and view integration are two good examples for such support (e.g., [64], [31]).

Third, as mentioned in section 3.3, an agreement can only be gained through communication among the involved people. Hence supporting the communications, conversations, coordination and collaboration between people as well as decision support leads to better and possibly faster agreements. Research done in the CSCW area can contribute basic solutions for this (e.g., [97], [42], [32], [27], [70]).

5.4 Process Modeling

To support the overall RE process a suitable process model must be developed for guiding the RE process within the three dimensions.

According to Dowson [26], process models can be classified in three categories: activity-oriented, product-oriented and decision-oriented models. From the viewpoint of requirements engineering, only the last category appears to be partially appropriate. It is probably difficult to impossible to write down a realistic state-transition diagram (to cite a popular activity-oriented model) that adequately describes what has to happen or actually happens in RE. But relying on the pure object history is also insufficient. Even the decision-based approach (e.g., [52], [88], [82]) offer only limited hints when and how to decide on what. The central aspect of the process model for RE is therefore that it makes the notion of situation (in which to decide) explicit and relates it to the broader question of context handling (e.g., [80]).

Using the three dimensions, for each action a prediction, how the specification will change after the actions was applied, can be made. For example for validation at least a prediction can be made, that after the validation, the *specification* dimension will be improved. Within the NATURE [50] project it is assumed, that the basic building block of any process can be modelled as a triplet <*situation, decision, action*> [43]. A process model based on this assumption for supporting the RE process within the three dimensions is currently under development.

The last two feature, to be mentioned here, is the importance of quality orientation and process improvement (cf. [53], [80] for more information about quality and improvement oriented process models). It was recognized within the mechanical engineering community, that it is insufficient to correct the missing quality of a product after the fact it was produced. Quality must be produced in the first place. Therefore quality

oriented process models are necessary. Especially in rapidly changing areas, like software production, it is very important to have evolving and quality oriented process models.

5.5 Easing the Influences on RE

As identified in section 4 five main influences on RE exist. *Social aspects, cognitive skills* and *economical constraints* are basic influences on the process. In contrast, *methods and methodologies* as well as *tools* are designed to support the process within the three dimensions, but also to ease the basic influences on the process (social aspects, cognitive skills and economical constraints). For designing appropriate methods, methodologies or tools knowledge gained within other research area can be used, e.g., management methods (e.g., TQM [23], [78]), organizational measures (e.g. value-added chains [81]).

Beside the task of building suitable methods and tools the need for recording of process knowledge was recognized to make the development process of software and specifications traceable (e.g., [24], [88], [54], [53]). Informal, semi-formal as well as formal knowledge must be recorded, and therefore interrelated. Hypertext is supposed to offer a solution for the integration of different representation (e.g., [37], [6], [38]).

6 Conclusions

In this paper we introduced a framework for requirements engineering (RE). First we focused on the essence of the RE process. We characterized the *'initial input'* of the RE process as opaque personal views at the system expressed using informal representation languages. The *'desired output'* was sketched as a complete system specification expressed using formal languages on which an agreement was reached. Based on this characterization the three main goals of RE were identified:

- gaining a complete system specification out of the opaque views available at the beginning of the process, according to the standard and/or guideline used,
- offering different representation formats, supporting the transformation between the representation (e.g., informal to semi-formal, informal to formal) and keeping the various representations consistent,
- allowing various views and supporting the evolution form personal views to common agreement on the finial specification.

Out of these, the three dimensions of RE were gained:

- *specification,*
- *representation and*
- *agreement dimension*

Looking at RE using these three dimensions we identified the main tasks and goals to be reached within each dimension during the RE process. But RE is not only driven by its goals, it is also influenced by the environment. We identified five main factors influencing requirements engineering: methods and methodologies, tools, social aspects, cognitive skills and economical constraints. Accordingly existing research and computer support was briefly sketched by distinguishing between computer support for improving

the specification in one of the three RE dimension, for guiding the process of RE and for easing the influences on RE.

Within the NATURE project this framework is used for classifying RE problems and for making process guidance possible. The framework itself should be seen as a first attempt to accomplish a common understanding of RE within the community. It should serve as a basis for discussing research topics and identifying the main problems of RE.

Acknowledgments

I am indebted to Stephan Jacobs and Matthias Jarke for many fruitful comments on an earlier version of this paper. Additionally I am grateful to John Mylopolous and many colleagues within the NATURE project for discussions which have positively influenced this paper.

References

1. B. Adelson and E. Soloway. The Role of Domain Experience in Software Design. *IEEE Transaction on Software Engineering*, 11(11), 1985.
2. Mack W. Alford. Software Requirements Engineering Methodology (SREM) at the age of two. In *4th Int. Computer Software & Applications Conference, New York*, pages 866–874. IEEE, 1980.
3. R. Balzer. Tolerating Inconsistency. In *Int. Conference on Software Engineering*, pages 158–165, Austin, Texas, 1991.
4. R. Balzer, N. Goldman, and D. Wile. Informality in program specifications. *IEEE Transactions on Software Engineering*, 4(2):94–103, 1978.
5. D.R. Barstow. Domain Specific Automatic Programming. *IEEE Transaction on Software Engineering*, 11(11), 1985.
6. James Bigelow. Hypertext and CASE. *IEEE Software*, pages 23–27, March 1988.
7. T. Biggerstaff and R. Richter. Reusability Framework, Assesment and Directions. *IEEE Transaction on Software Engineering*, 13(2), 1987.
8. D. Bjoerner and C.B. Jones. *VDM'87 VDM-A Formal Method at Work*. LNCS 252, Springer Verlag, 1988.
9. Alexander Borgida, Sol Greenspan, and John Mylopoulos. Knowledge Representation as the Basis for Requirements Specifications. *Computer*, 18(4):82–91, April 1985.
10. Marilyn Bush. Improving Software Quality: The use of Formal Inspections at the Jet Propulsion Laboratory . In *Proc. of the 12th Int. Conf. on Software Engineering, March 26–30, Nice, France*, pages 196–199, 1990.
11. B.W.Boehm. Verifying and Validating Software Requirements and Design Specifications. *IEEE Software*, 1(1):75–88, January 1984.
12. John R. Camaron. An Overview of JSD. *IEEE Transaction on Software Engineering*, 12(2):222–240, February 1986.
13. P.P.S. Chen. The Entity-Relationship Approach: Towards a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), 1976.
14. Peter Coad and Edward Yourdon. *Object Oriented Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

15. J. Conklin and M. J. Begeman. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *ACM Transaction on Office Information Systems*, 6(4):303–331, 1988.
16. P. Constantopoulos, M. Jarke, J. Mylopoulos, and Y. Vassiliou. Software Information Base: A server for reuse. ESPRIT project ITHACA, Heraklion, Crete, ICS-FORTH, 1991.
17. B. Curtis, H Krasner, and N. Iscoe. Field Study of the Software Design Process for Large Systems. *Communication of the ACM*, 33(11):1268–1287, 1988.
18. Bill Curtis, Marc I. Kellner, and Jim Over. Process Modelling. *Communications of the ACM*, 35(9):75–90, September 1992.
19. A. Czuchry and D. Harris. KBSA: A New Paradigm for Requirements Engineering. *IEEE Expert*, 3(4):21–35, 1988.
20. Alan M. David. The Analysis and Specification of Systems and Software Requirements. In Thayer R.H. and M. Dorfman, editors, *Systems and Software Requirements Engineering*, pages 119–134. IEEE Computer Society Press — Tutorial, 1990.
21. Alan M. Davids. A Comparison of Techniques for the Specification of External System Behavior. *Communications of the ACM*, 31(9):1098–1115, 1988.
22. V. de Antonellis, B. Pernici, and P. Samarati. F-ORM Method: Methodology for reusing Specifications. *ITHACA Journal*, (14):1–24, 1991.
23. W. E. Deming. *Out of the Crisis*. Massachusetts Instituite of Technology, Center for Advanced Engineering Study, Cambridge, 1986.
24. V. Dhar and M. Jarke. Dependency Directed Reasoning and Learning in System Maintenance Support. *IEEE Transactions on Software Engineering*, 14(2):211–228, 1988.
25. Merlin Dorfman and Richard H. Thayer. *Standards, Guidelines and Examples on System and Software Requirements Engineering*. IEEE Computer Society Press – Tutorial, 1990.
26. M. Dowson. Iteration in the Software Process. In *Proceedings 9th Int. Conf. on Software Engineering*, April 1987.
27. C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some Issues and Experience. *Communication of the ACM*, 34(1):38–58, 1991.
28. M. S. Feather and S. Fickas. Coping with Requirements Freedom. In *Proceedings of the International Workshop on the Development of Intelligent Information Systems*, pages 42–46, Niagara-on-the-Lake, Ontario, Canada, April 1991.
29. S. Fickas. Automating analysis: An example. In *Proceedings of the 4th International Workshop Software Specification and Design*, pages 58–67, Washington, DC, April 1987.
30. S. Fickas and P. Nagarajan. Critiquing Software Specifications. *IEEE Software*, pages 37–47, November 1988.
31. A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A Framework for Integration Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering*, 1(2), May 1992.
32. Gerhard Fischer, Raymond McCall, and Anders Morch. JANUS: Integrating Hypertext with a Knowledge-based Design Environment. In *Proceedings of Hypertext '89, November 5–8, Pittsburgh, Pennsylvania*, pages 105–117, 1989.
33. R.F. Flynn and D. Dorfmann. The Automated Requirements Traceability System (ARTS): An Experience of Eight Year. In Thayer R.H. and M. Dorfman, editors, *Systems and Software Requirements Engineering*, pages 423–438. IEEE Computer Society Press — Tutorial, 1990.
34. Martin D. Fraser, Kuldeep Kumar, and Vijay K. Vaishnavi. Informal and Formal Requirements Specification Languages Bridging the Gap. *IEEE Transactions on Software Engineering*, 17(5):454–466, May 1991.
35. Daniel P. Freeman and Gerald M. Weinberg. *Handbook of Walkthroughs, Inspections and Technical Reviews*. Dorset House Publishing, New York, 1990.
36. P. Freemann, editor. *Software reusability*. IEEE Press – Tutorial, 1987.

37. Pankaj K. Garg and Walt Scacchi. On Designing Intelligent Hypertext Systems for Information Management in Software Engineering. In *Proceedings of Hypertext '87, November 13–15, Chapel Hill, North Carolina*, pages 409–432, 1987.
38. Pankaj K. Garg and Walt Scacchi. A Hypertext System to Manage Software Life-Cycle Documents. *IEEE Software*, pages 90–98, May 1990.
39. D. Gentner. Structure Mapping: A Theoretical Framework for Analogy. *Cognitive Science*, 5:121–152, 1983.
40. Joseph A. Goguen, Marina Jirotka, and Matthew J. Bickerton. Research on Requirements Capture and Analysis. Technical report, Oxford University Computing Laboratory, Centre for Requirements and Foundations, December 1991.
41. S.J. Greenspan. *Requirements Modeling: A Knowledge Representation Approach to Software Requirements Definition*. PhD thesis, Dept. of Computer Science, University of Toronto, 1984.
42. I. Greif, editor. *Readings in Computer-Supported Cooperative Work*. Morgan Kaufmann, 1988.
43. George Grosz and Colette Roland. Using artificial intelligence techniques to formalize the information system design process. In *Proc. Int. Conf. Databases and expert Systems Applications*, pages 374–380, 1990.
44. R. Guidon and B. Curtis. Control of cognitive process during software design: What tools are needed? In E. Soloway, D. Frye, and S.B. Sheppard, editors, *Proc. of CHI '88 Conference: Human Factors in Computer Systems*, pages 263–269. ACM Press NY, 1991.
45. J. Hagelstein. Declarative Approach to Information Systems Requirements. *Knowledge Base Systems*, 1(4):211–220, 1988.
46. Anthony Hall. Seven Myths of Formal Methods. *IEEE Software*, (9):11–19, September 1990.
47. C.A.R. Hoare. International Conference on VDM and Z. LNCS 428, Springer Verlag, 1990.
48. IEEE. *Standards, Guidelines, and Examples on System and Software Requirements Engineering*. IEEE Computer Society Press – Tutorial, 1990.
49. IEEE. IEEE Std. 830–1984. In *IEEE Software Engineering Standards Collection*. IEEE, New York, 1991.
50. Matthias Jarke, Janis Bubenko, Colette Rolland, Allistair Sutcliffe, and Yannis Vassiliou. Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis. In *Proceedings of the 1th Int. Symposium of Requirements Engineering*, San Diego, CA, 1993. to appear.
51. Matthias Jarke, Stephan Jacobs, and Klaus Pohl et. al. Requirements Engineering: An Integrated View of Representation, Process and Domain. In *submitted to: ECSE '93*, 1993.
52. Matthias Jarke, Manfred Jeusfeld, and Thomas Rose. A Software Process Data Model for Knowledge Engineering in Information Systems. *Information Systems*, 15(1):85–116, 1990.
53. Matthias Jarke and Klaus Pohl. Information System Quality and Quality Information Systems. In *Proceedings of the IFIP 8.2 Working Conference on the Impact of Computer-Supported Techniques on Information Systems Development*, 1992.
54. Matthias Jarke and T. Rose. Specification Management with CAD^0. In P. Loucopoulos and R. Zicari, editors, *Conceptual Modeling Databases, and CASE*, 1991.
55. Manfred Jeusfeld. *Änderungskontrolle in deduktiven Objektbanken*. INFIX Pub, Bad Honnef, Germany, 1992.
56. P. Johannesson and K. Kalman. A Method for Translating Relational Schemas into Conceptual Schemas. In *8th Int. Conf. on Entity-Relationship Approach*, pages 279–294, 1989.
57. W. Lewis Johnson. Deriving Specifications from Requirements. In *Proceedings of the 10th International Conference on Software Engineering*, pages 428–438, Singapore, April 1988.
58. W. Lewis Johnson and Martin Feather. Building An Evolution Transformation Library. In *Proceedings of the 12th International Conference on Software Engineering*, pages 428–438, Nice, France, March 1990.

59. W. Lewis Johnson, Martin. S. Feather, and David. R. Harris. Representation and Presentation of Requirements Knowledge. *IEEE Transactions on Software Engineering*, 18(10), October 1992.
60. W. Lewis Johnson and David R. Harris. The ARIES Project. In *Proceedings 5th KBSA Conference*, pages 121–131, Liverpool, N.Y., 1990.
61. S. E. Keller, L. G. Kahn, and R. B.Panara. Specifying Software Quality Requirements with Metric. In Thayer R.H. and M. Dorfman, editors, *Systems and Software Requirements Engineering*, pages 145–163. IEEE Computer Society Press — Tutorial, 1990.
62. Manolis Koubarakis, John Mylopoulos, Martin Stanley, and Matthias Jarke. Telos: A Knowledge Representation Language for Requirements Modelling. Technical Report KRR-TR-89–1, Department of Computer Science, University of Toronto, 1989.
63. Julio Cesar S. P. Leite. Viewpoint Analysis: A Case Study. In *Proceedings of the 5th International Workshop on Software and Design*, pages 111–119, Pittsburgh, PA, 1989.
64. Julio Cesar S. P. Leite and Peter A. Freeman. Requirements Validation Through Viewpoint Resolution. *IEEE Transactions on Software Engineering*, 17(12):1253–1269, December 1991.
65. P. Loucopoulos and R. Champion. Knowledge-Based Approach to Requirements Engineering Using Method and Domain Knowledge. *Knowledge-Based Systems*, 1(3), 1988.
66. M.D. Lubars and M.T. Harandi. Knowledge-Based Software Design Using Design Schemas. In *Proceedings 9th Int. Conf. on Software Engineering*, April 1987.
67. Neil Maiden. Analogy as a Paradigm for Specification Reuse. *Software Engineering Journal*, 1991.
68. Neil Maiden. *Analogical specification Reuse during Requirements Analysis*. PhD thesis, City University London, 1992.
69. M. Mannino and V. Tseng. Inferring Database Requirements from Examples in Forms. In *Int. Conf. on Entity-Relationship Approach*, pages 391–405. Elsevier Publishers B.V. (North-Holland), 1989.
70. David Marca and Geoffrey Bock. *Groupware: Software for Computer-Supported Cooperative Work*. IEEE Computer Society Press, Los Alamitos, CA, 1992.
71. Stephen M. McMenamin and John F. Palmer. *Essential System Analysis*. Yourdon Press, Prentice Hall, Englewood Cliffs, NJ 07632, 1984.
72. Richard H. Thayer Merlin Dorfman, editor. *Standards, Guidelines, and Examples on System and Software Requirements Engineering*, chapter ESA Software Engineering Standards, pages 101–120. IEEE Computer Society Press Tutorial, 1990.
73. Bertrand Meyer. On Formalism in Specifications. *IEEE Software*, pages 6–26, January 1985.
74. Kanth Miriyala and Mehdi T. Harandi. Automatic Derivation of Formal Software Specifications Form Informal Descriptions. *IEEE Transactions on Software Engineering*, 17(10):1126–1142, October 1991.
75. David E. Monarchi and Gretchen I. Puhr. A Research Typology for Object-Oriented Analysis and Design. *Communications of the ACM*, 35(9):35–47, September 1992.
76. John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: Representing Knowledge about Information Systems. *Transactions on Information Systems*, 8(4):325–362, 1990.
77. John Mylopoulos and Hector J. Levesque. *On Conceptual Modelling*. Springer Verlag, 1986.
78. J. S. Oakland. Total Quality Management. In *Proceedings 2nd Int. Conf. on Total Quality Management*, pages 3–17. Cotswold Press Ltd., 1989.
79. Barbara Pernici. Requirements Specifications for Object Oriented Systems. *ITHACA Journal*, (8):43–63, January 1991.
80. Klaus Pohl and Matthias Jarke. Quality Information Systems: Repository Support for Evolving Process Models. Technical report, RWTH Aachen, Informatik-Berichte 37–92, 1992.

81. M. Porter. *Competitive Advantage*. Free Press, New York, 1985.
82. C. Potts. A Generic Model for Representing Design Methods. In *Proceedings 11th International Conference on Software Engineering*, 1989.
83. C. Potts and G. Bruns. Recording the Reasons for Design Decisions. In *Proceedings 10th International Conference on Software Engineering*, 1988.
84. P. Paolo Puncello, Piero Torrigiani, Francesco Pietri, Riccardo Burlon, Bruno Cardile, and Mirella Conti. ASPIS: A Knowledge-Based CASE Environment. *IEEE Software*, pages 58–65, March 1988.
85. B. Ramesh and V. Dhar. Process-Knowledge Based Group Support in Requirements Engineering. *IEEE Transactions on Software Engineering*, 18(6), 1992.
86. Howard B. Reubenstein and Richard C. Waters. The Requirements Apprentice: Automated Assistance for Requirements Acquisition. *IEEE Transactions on Software Engineering*, 17(3):226–240, March 1991.
87. C. Rolland and C. Proix. A Natural Language Approach for Requirements Engineering. In *Proceedings of the 4th International Conference on Advanced Information Systems Engineering*, LNCS 593, 1992.
88. T. Rose, M. Jarke, M. Gocek, C.G. Maltzahn, and H.W. Nissen. A Decision-based Configuration Process Environment. *Special Issue on Software Process Support, IEE Software Engineering Journal*, 6(5):332–346, 1991.
89. H.H. Sayani. PSL/PSA at the Age of Fifteen. In Thayer R.H. and M. Dorfman, editors, *Systems and Software Requirements Engineering*, pages 403–417. IEEE Computer Society Press — Tutorial, 1990.
90. Walt Scacchi. Managing Software Engineering Projects: A Social Analysis. *IEEE Transaction on Software Engineering*, 10(1):49–59, 1984.
91. G. Spanoudakis and P. Constantopoulos. Similarity for Analogical Software Reuse. In *Proc. ERCIM Workshop on Methods and Tools for Software Reuse*, Heraklion, Crete, 1992.
92. J.M. Spivey. An introduction to Z and formal specifications. *Software Engineering Journal*, 4(1):40–50, 1990.
93. Alistair Sutcliffe. Object Oriented Systems Analysis: The Abstract Question. In *Proc. IFIP WG 8.1 Conf. The Object Oriented Approach in Information Systems*, Quebec City, Canada, 1991.
94. Alistair Sutcliffe and Neil Maiden. Software reuseability: Delivering Productivity gains or short cuts. In *Proceedings INTERACT*, pages 948–956. North-Holland, 1990.
95. C.P. Svoboda. Structured Analysis. In Thayer R.H. and M. Dorfman, editors, *Systems and Software Requirements Engineering*, pages 218–227. IEEE Computer Society Press — Tutorial, 1990.
96. Jeanette M. Wing. A Specifier's Introduction to Formal Methods. *Computer*, (9):8–24, September 1990.
97. T. Winograd and F. Flores. *Understanding Computers and Cognition: A New Foundation for Design*. Ablex Norwood, NJ, 1986.
98. Edward Yourdon. *Modern Structured Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
99. Edward Yourdon. *Structured Walkthroughs*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
100. Pamala Zave. An Insider's Evaluation of PAISLey. *IEEE Transaction on Software Engineering*, 17(3):212–225, March 1991.
101. Pamela Zave. A Comparison of the Major Approaches to Software Specification and Design. In Thayer R.H. and M. Dorfman, editors, *Systems and Software Requirements Engineering*, pages 197–199. IEEE Computer Society Press — Tutorial, 1990.