From: CAiSE 1995, LNCS 932 © Springer-Verlag Berlin Heidelberg 1995

Towards a Deeper Understanding of Quality in Requirements Engineering

John Krogstie Odd Ivar Lindland Guttorm Sindre

Faculty of Electrical Engineering and Computer Science
University of Trondheim, Norway

Abstract. The notion of quality in requirements specifications is poorly understood, and in most literature only bread and butter lists of useful properties have been provided. However, the recent frameworks of Lindland et al. and Pohl have tried to take a more systematic approach. In this paper, these two frameworks are reviewed and compared. Although they have different outlook, their deeper structures are not contradictory.

The paper also discusses shortcomings of the two frameworks and proposes extensions to the framework of Lindland et al. The extensions build on social construction theory and the resulting framework should contribute to understanding quality in requirements engineering and conceptual modelling.

Keywords: **Requirements engineering, conceptual modelling, quality, social construction**

1 Introduction

The notion of quality in requirements specifications is so far poorly understood. Software metrics [7] have mostly concentrated on the deliverables of the later phases, such as design and coding, or on detailed process metrics. Moreover, these efforts have concentrated far more on the issue of 'building the product right' than 'building the right product', whereas both should be covered to ensure quality from the user's point of view [2]. Previously proposed quality goals for conceptual models [6, 14, 22, 25] have included many useful aspects, but unfortunately in the form of unsystematic bread and butter lists. Two recent frameworks [17, 20] have attempted to take a more structured approach to understanding the problem. Still, both these need more development before they can result in concrete guidelines for the requirements engineering process. A useful first iteration is to compare the two frameworks and see if they fit together, and possibly unite and extend them.

The rest of the paper is structured as follows: Section 2 reviews and compares the two frameworks. Then, section 3 establishes an extended framework based on the comparison. Section 4 concludes the paper. The terminology used in the papers follows the one usually used in the areas of conceptual modelling and requirements engineering. One should be aware of that the use of many terms in these areas differs significantly from their use in for instance logic programming.

2 Review and Comparison

We will briefly present the main parts of the two frameworks, before performing a comparison between them.

2.1 Lindland/Sindre/Sølvberg's Framework

The main structure of this framework is illustrated in Figure 1. The basic idea is to evaluate the quality of models along three dimensions — syntax, semantics, and pragmatics — by comparing sets of statements. These sets are:

- $\mathcal{M}$, the model, i.e., the set of all the statements explicitly or implicitly made in the model. The explicit model $\mathcal{M}_E$, consists of the statements explicitly made, whereas the implicit model, $\mathcal{M}_I$, consist of the statements not made, but implied by the explicit ones.
- $\mathcal{L}$, the language, i.e., the set of all statements which are possible to make according to the vocabulary and grammar of the modelling languages used.
- $\mathcal{D}$, the domain, i.e., the set of all statements which would be correct and relevant about the problem at hand. Hence, notice that the term domain is used somewhat differently from the usual. Here, it means the 'ideal' model/solution to the problem.
- $\mathcal{I}$, the audience interpretation, i.e., the set of all statements which the audience (i.e., various actors in the modeling process) think that the model consists of.

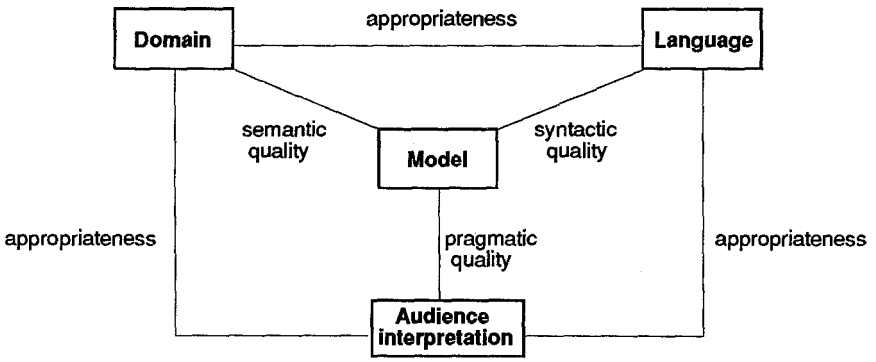

Fig. 1.: The framework by Lindland et al. (From [17])

The primary sources for model quality are defined using the relationships between the model and the three other sets:

- *syntactic quality* is the degree of correspondence between model and language, i.e., the set of syntactic errors is $\mathcal{M} \setminus \mathcal{L}$.
- *semantic quality* is the degree of correspondence between model and domain. If $\mathcal{M} \setminus \mathcal{D} \neq \emptyset$ the model contains invalid statements; if $\mathcal{D} \setminus \mathcal{M} \neq \emptyset$ the model is incomplete. Since total validity and completeness are generally impossible, the notions of *feasible validity* and *feasible completeness* were introduced. Feasible validity is reached when the benefits of removing invalid statement from $\mathcal{M}$ are less than the drawbacks, whereas feasible completeness is reached when the benefits of adding new statements to $\mathcal{M}$ is less than the drawbacks. The term drawback is used instead of the more familiar term cost in an effort to cover both purely economic issues and factors like user preferences and ethics.
- *pragmatic quality* is the degree of correspondence between model and audience interpretation (i.e., the degree to which the model has been understood). If $\mathcal{I} \neq \mathcal{M}$, the comprehension of the model is not completely correct. Usually, it is neither necessary nor possible that the whole audience understand the entire conceptual model — instead each group in the audience should understand the part of the model which is relevant to them. *Feasible comprehension* was defined along the same lines as feasibility for validity and completeness.

In addition to these primary quality concerns, it is pointed out that correspondence between domain and language, between domain and audience interpretation, and between language and audience interpretation may affect the model quality indirectly. These relationships are all denoted *appropriateness* as shown in Figure 1.

It is also argued that previously proposed quality goals such as minimality, traceability, consistency, and unambiguity are subsumed by the four goals of syntactic correctness, validity, completeness, and comprehension, and a distinction is made between goals and means to reach these goals. For more details on this framework, the reader should consult [17]. The parts of the framework dealing with fault detection have been applied in connection with integrating the development and testing of object-oriented software [18].

2.2 Pohl's Framework

Pohl's framework [20] which is one of the results of the NATURE-project [12] defines three dimensions of requirements engineering:

- *the specification dimension* deals with the degree of requirements understanding. At the beginning of the process, this understanding is opaque. The desired output of the RE process is a complete system specification, where completeness is measured against some standard, guideline, or model.
- *the representation dimension* deals with the degree of formality. Various languages can be used in the process; informal ones such as natural language,

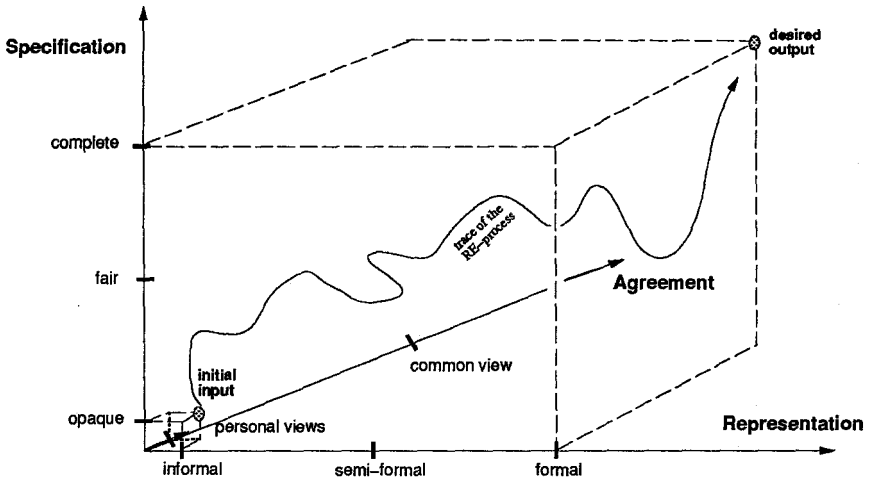

Fig. 2.: Pohl's framework (From [20])

semi-formal ones such as many graphical modelling languages, and formal ones (e.g., logic). At the beginning of the process, statements will usually be informal. Since formal representations allow reasoning and partial code-generation, these are more system-oriented. Hence, a transformation of informal requirements to a formal representation is desirable.

- *the agreement dimension* deals with the degree of agreement. The RE process has many stakeholders, and in the beginning each of these will have their personal views concerning the requirements to be made. The goal of the process is to reach agreement on the requirements. Detected conflicts must be solved through discussions among those affected.

The RE process can be characterized as an arbitrary curve within the cube spanned by these three dimensions, as illustrated in Figure 2. Pohl distinguishes between *original* RE problems, which are those caused by the three dimensions, and problems caused by approaches to solve the original problems, i.e., those related to methods, tools, social aspects, cognitive skills and economical constraints. Furthermore, the article discusses the computer support for RE in light of the three dimensions and discusses how the framework can be applied in analyzing RE methods, practise, problems, and process situations.

2.3 Overall comparison and critique

At first sight, the two frameworks may seem completely different. The terminology used is different. Lindland et al. [17] defines the quality of models (e.g., requirements specifications) according to the linguistic dimensions of syntax,

semantics, and pragmatics, Pohl's framework [20] identifies the goals of requirements engineering along the three dimensions of completeness, formality, and agreement.

Although the two frameworks have a quite different appearance, they are rather similar in their deeper structure. The following observations can be made:

- the *representation* dimension corresponds to the *syntactic* dimension, since both these deal with the relationship between the specification and the language(s) used. The main differences in this respect is that Pohl's framework discusses several languages, whereas Lindland's framework sees the language as one and just considers whether the specification is correct according to the rules of that language (which may be a union of several languages, formal and informal). It should also be noted that Pohl's framework regards a formal specification as a *goal.* Lindland's framework states that formality is a *mean* to reach a syntacticly correct specification, as well as higher semantic and pragmatic quality through consistency checking and model executions of different kinds.

- the *specification* dimension corresponds to the *semantic* dimension, since both these deal with the goal of completeness. A notable difference here is that Pohl sees completeness as the sole goal (possibly including validity?), whereas Lindland's framework also identifies the notions of validity and feasibility. The reason for this discrepancy seems to be a somewhat different use of the term completeness, where Pohl uses the term relative to some standard, whereas Lindland et al. uses it relative to the the set of all statements which would be correct and relevant about the problem at hand.

- the *agreement* dimension is related to the *pragmatic* dimension, since both these deal with the specification's relationship to the audience involved. The difference is that Pohl states the goal that the specification should be agreed upon, whereas Lindland et al. aim at letting the model be understood. In a way these goals are partly overlapping. Agreement without understanding is not very useful in a democratic process. On the other hand, using the semiotic levels described in the FRISCO-report [16], it seems more appropriate to put agreement into the social realm, thus going beyond the framework of Lindland et al.

Although both frameworks contribute to improving the understanding of quality issues in requirements engineering, they still have several shortcomings. For instance, in Pohl's framework it appears that a formal, agreed, and complete specification is the goal of the requirements engineering phase. Although we support this as desirable, we — as argued in [17] feel that such goals are unrealistic and we need mechanisms for discussing when the specification/model is *good enough.* The notion of feasibility that is included in Lindland's framework addresses this aspect. In Pohl's framework such mechanisms are only implicitly included through the adherence to standards which potentially include them.

We also feel it is problematic that a completely formal representation is a goal of the RE process. It is not always desirable that all the products of a requirement

specification process are formal. For instance, when developing a goal-hierarchy as used in, e.g., TEMPORA [23], it is not meaningful to formalise the high-level business goals, even if these are an important result of requirements engineering in order for the participants to understand and agree about the requirements to the information system. This kind of information is also of vital importance when the requirements to the information systems must be reevaluated during maintenance.

In Lindland's framework, on the other hand, the social aspect of agreement is currently not handled in a satisfactory way. Even if people understand the requirements, this does not mean that they will agree to them. When discussing agreement, the concept of domain as currently defined is also insufficient, since it represents some ideal knowledge about a particular problem, a knowledge not obtainable for the actors that are to agree.

3 Framework extensions

This section aims at extending Lindland et al.'s framework in order to include some of the good aspects of Pohl's framework and also hopefully eliminate the inherent shortcomings of the current version of Lindland's framework.

The key area for improvement is related to the relationships between the domain, model, and audience interpretations and the introduction of the social goal of agreement.

3.1 Background on social construction

Since 'agreement' was not thoroughly discussed in [17], we will first introduce our ontological position for discussing the concept. This will also influence some of the other relationships in the framework.

We base our treatment of agreement on the idea that 'reality' is socially constructed [1], an idea which is the foundation of most of the current theoretical discussion within social sciences [5], and which has received increased attention in the information systems community [8, 16, 24]. For a constructivist, the relationship between 'reality' and models of this reality are subject to negotiation among the audience, and may be adapted from time to time. This is in contrast to a more traditional objectivistic ontology, where the relationship between 'reality' and models thereof is obvious.

The mechanisms of social construction in an organization can briefly be described as follows [9]: An organization consists of individual social actors that perceive the world in a way specific to them. The *local reality* is the way the individual perceives the world that s/he acts in. Whereas some of this local reality may be made explicit and talked about, a lot of what we know about the world is tacit. The term 'individual knowledge' is below restricted to the *explicit* local reality of an individual actor.

When social actors act, they *externalise* their local reality. The most important ways the social actors of an organization externalise their reality, are to

speak and to construct languages, artifacts, and institutions. What they do is to construct *organizational reality*: To make something that other actors have to relate to in their work. Finally, *internalisation* is the process of making sense of the institutions, artifacts, technology etc. in the organization, and making this organizational reality part of the individual local reality.

Whereas the development of a requirements specification based on a social actor's local reality is partly a process of externalisation of her/his reality, the process of developing conceptual models can also be looked upon as part of a sense-making process. The views of several actors are collected in a conceptual model and agreement about the validity of this is reached. It should also be noted that the ability and possibility for the different stakeholders to externalise their local reality will differ. Thus, in the words of Goguen one should think about requirements as " ...emergent, in the sense that they do not already exist, but rather emerge from interactions between the analyst and the client organization" [10].

In the framework of Lindland et al, 'reality' is represented by the domain, $\mathcal{D}$. The domain represents the perfect understanding of the problem. From the viewpoint of social construction, as well as the view of information systems engineering as a wicked problem [21], it can be questioned whether a perfect solution at all exists. This is not an important point, however, since the perfect solution is anyway stated to be unachievable. Hence, the domain $\mathcal{D}$ serves only as a useful conceptual fixpoint to make it easier to define quality terminology. To discuss the social aspects, the actors' understanding of the domain must be added to the framework, in the same sense as their understanding of the model was already introduced in the previous version of the framework.

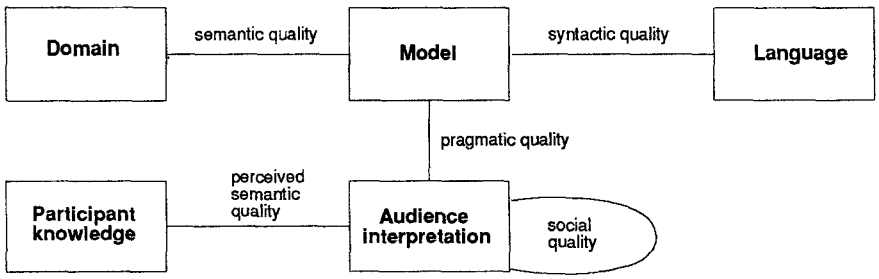

Fig. 3.: Extended framework

3.2 Extended framework

We are now ready to extend the framework of Lindland et al. The main concepts and their relationships are shown in Figure 3. The following sets are defined:

- $\mathcal{A}$, the audience, i.e., the union of the set of individual actors A_1,...,A_k the set of organizational social actors A_{k+1},...,A_n and the set of technical actors A_{n+1},...,A_m who needs to relate to the model. The individual social actors being members of the audience is called the *participants* of the modelling process. An organizational social actor is made up of several individuals. The audience consists of all who need to understand the model during the RE process. The participants are a subset of the stakeholders of the process of developing the new or improved information system, a stakeholder being someone who potentially stands to gain or lose in the process. Stakeholders typically include project managers, system developers and analysts, financers, maintainers, and future users.

  A technical actor is typically a computer or computer program, which must "understand" part of the specification to automatically manipulate it. $\mathcal{A}$ is often evolving during the process of requirements engineering.

- $\mathcal{M}$, the model, i.e., the set of all statements explicitly or implicitly made in the model. At an early point of requirements engineering there may be one model for each participant, but usually fewer models which are the joint models of organizational actors exists. For each participant, the part of the model which is considered relevant for the actor can be seen as a projection of the total model, hence $\mathcal{M}$ can be divided into projections $\mathcal{M}^1, ..., \mathcal{M}^k$ corresponding to the involved participants $A_1, ..., A_k$. Generally, these projections will not be disjoint, but their union cover $\mathcal{M}$. The complete model will be evolving during the process of requirements engineering.

- $\mathcal{L}$, the language, i.e., the set of all statements that are possible to make according to the vocabulary and grammar of the modelling languages used. Several languages can be in use at the same time, corresponding to the sets $\mathcal{L}_1$,...,$\mathcal{L}_j$. A sub-language is related to the complete language by limitations on the vocabulary or on the set of allowed grammar rules or both.

  The set $\mathcal{L}$ can be divided into several subsets, e.g., $\mathcal{L}_I$, $\mathcal{L}_S$, and $\mathcal{L}_F$ for the informal, semi-formal and formal parts of the language, respectively. A language with formal syntax is termed semi-formal, whereas a language which also has formal semantics, is termed formal. Note that this does not imply that the language has a semantics based on formal logic.

- $\mathcal{D}$, the domain, i.e., the set of all statements which would be correct and relevant about the problem at hand. $\mathcal{D}$ denotes the ideal knowledge about the problem. The domain evolves during the requirements engineering process.

- $\mathcal{I}$, the audience interpretation, i.e., the set of all statements which the audience thinks that a model consists of. Various parts of the model will be of interest to various participants. Just like the model is projected into $cal M^1, ..., \mathcal{M}_k$ above, its interpretation can be projected into $\mathcal{I}_1, ..., \mathcal{I}_k$ according to the interests of the participants.

- $\mathcal{K}$, the knowledge of the participants, i.e., the union of the sets of statements $\mathcal{K}_1$,...,$\mathcal{K}_k$, one for each individual social actor in the audience. The set $\mathcal{K}_i$ contains all possible statements that would be correct and relevant for addressing the problem at hand according to the knowledge of the actor A_i. $\mathcal{K}_i$ is a subset of the explicit internal reality of the social actor $\mathcal{K}^i$. $\mathcal{K}^i$ is also evolving during requirements engineering. $\mathcal{M}_i$ is an externalisation of $\mathcal{K}_i$ and is a model made on the basis of the knowledge of the individual actor. Even if the internal reality of each individual will always differ to a certain degree, the explicit internal reality concerning a constrained area might be equal, especially within groups of social actors [9, 19].

With this new framework in place, we have an increased potential for discussing specification quality. The primary goal for semantic quality is a correspondence between the model and the domain, but this correspondence can neither be established nor checked directly: to build the model, one has to go through the audience's knowledge of the domain, and to check the model one has to compare this with the audience's interpretation of the model. Hence, what we do observe at quality control is not the actual semantic quality of the model, but a perceived semantic quality based on comparison of the two imperfect interpretations.

Syntactic quality Syntactic quality is the correspondence between the specification and the language. The goal is syntactic correctness, $\mathcal{M} \setminus \mathcal{L} = \emptyset$, or for a given externalization, $\mathcal{M}_i \setminus \mathcal{L} = \emptyset$. Typical means to ensure syntactic quality is *formal syntax*, i.e., that the language is parsable by a technical actor in the audience, and the modeling activity to perform this is termed syntax checking.

Semantic quality For the semantic quality of the complete model $\mathcal{M}$, no major changes are necessary to the previous version of the framework. [17] defines two goals, feasible validity and feasible completeness.

Discussing perceived semantic quality, we get the following:

- *Perceived validity* of the model projection: $\mathcal{I}_i \setminus \mathcal{K}_i = \emptyset$.
- *Perceived completeness* of the model projection: $\mathcal{K}_i \setminus \mathcal{I}_i = \emptyset$.

The perceived semantic quality can change, for better or for worse, either as a result of changes in (the understanding of) the model, or as a result of changes in the knowledge about the domain. Notice that one way the knowledge of the actor can change, is through the internalization of another sub-model. Internalisation can be expressed crudely as a mapping between the sets of statements, being part of the explicit internal reality of an actor.

$$INT : \mathcal{K}_i \rightarrow (\mathcal{K}_i \cup \mathcal{N}) \subset \mathcal{M}_j \setminus (\mathcal{O} \subset \mathcal{K}_i) \cdot \tag{1}$$

$i \neq j, \mathcal{O} \cap \mathcal{N} = \emptyset, \mathcal{K}_i \setminus \mathcal{N} = \mathcal{K}_i$

$\mathcal{N}$ and $\mathcal{O}$ above is sets of statements. $\mathcal{O}$ might be empty giving a monotonous growth of $\mathcal{K}_i$. If $\mathcal{O}$ is not empty there is a non-monotonous growth of $\mathcal{K}_i$.

Pragmatic quality Pragmatic quality can be defined largely the same way as before, the goal being comprehension, i.e. that the model is understood, not its understandability. [17] also defined this on behalf of various participant groups, since each such group will usually only be interested in a part of the model. Similarly, we can define individual comprehension: $\mathcal{I}_i = \mathcal{M}^i$, as the goal that the participant A_i understands the relevant part of the model.

For total comprehension, one must thus have $(\forall i, i \in [1...k])\ \mathcal{I}_i = \mathcal{M}^i$, i.e., that every participant understands the relevant part of $\mathcal{M}$.

Total comprehension is also an unrealistic goal. Hence it is interesting to define feasible comprehension as the situation where comprehension can still be improved, but the drawbacks of doing this exceeds the benefits. This has been done in [17].

That a model is understood from the technical actor's point of view, means that $(\forall i, i \in [n+1...m])\mathcal{I}_i = \mathcal{M}^i$, thus all statements that are relevant to the technical actor to be able to perform code generation, simulation, etc. is comprehended by this actor. In this sense, formality can be looked upon as being a pragmatic goal, formal syntax and formal semantics are means for achieving pragmatic quality. This illustrates that pragmatic quality is dependant on the different actors. This also applies to social actors. Whereas some individuals from the outset are used to formal languages, and a formal specification in fact will be best for them also for comprehension (regardless of execution etc.), other individuals will find a mix of formal and informal statements to be more comprehensive, even if the set of statements in the model is in fact redundant.

Some of the means to achieve pragmatic quality have been identified earlier, namely formality, executability, expressive economy and aesthetics. The corresponding modelling activities are inspection, visualization, filtering, diagram layout, paraphrasing, explanation, execution, animation, and simulation. Another important activity is training the participants in the syntax and semantics of the modelling languages used.

Social quality Inspired by Pohl, we set up the goal for social quality as *agreement.* However, this is not straightforward to define. Four kinds of agreement can be identified, according distinctions along two orthogonal dimensions:

- agreement in knowledge vs. agreement in model interpretation.
- relative agreement vs. absolute agreement

Agreement in model interpretation will usually be a more limited demand than agreement in knowledge, since the former one means that the actors agree about what (they think) is stated in the model, whereas there may still be many issues they disagree about which have not been stated in the model so far, even if it might be regarded as relevant for one of the actors.

Relative agreement means that the various projections are consistent — hence, there may be many statements in the projection of one actor that are not present in that of another, as long as they do not contradict each other. Absolute agreement, on the other hand, means that all projections are the same.

Since different participants often have their expertise in different fields, relative agreement is a more useful concept than absolute agreement. On the other hand, the different actors must have the *possibility* to agree on something, i.e. the parts of the model which are relevant to them have to overlap to some extent.

However, it is not given that all participants will come to agreement. Few decisions are taken in society under complete agreement, and those that are are not necessarily good, due to e.g group-think. To answer this we introduce *feasible agreement*:

Feasible agreement: A situation of feasible comprehension where inconsistencies between statements in the different $\mathcal{I}_i$ are resolved by choosing one of the alternatives when the benefits of doing this is less than the drawbacks of working out agreement.

The pragmatic goal of comprehension is looked upon as a social mean. This because agreement without comprehension is not very useful, at least not when having democratic ideals. Obviously if someone is trying to manipulate a situation, agreement without comprehension is useful. The area of *model monopoly* [3] is related to this.

Some activities for achieving feasible agreement are:

- Viewpoint analysis [15]: This includes techniques for comparing two or more models and find the discrepancies.
- Conflict resolution: Specific techniques for this can be found in the area of computer supported cooperative work, see [4, 11] where argumentation systems are presented.
- Model merging: Merging two potentially inconsistent models into one consistent one.

The above activities can be done either manually, semi-automatically or automatically, for semi-automatic or automatic support, formal syntax and semantics are again useful. In addition is it useful to be able to represent inconcistency and disagreement directly in the model, and not only have to compare separate models.

4 Concluding Remarks

This paper has reviewed and compared two recent frameworks for disussing quality of requirement specifications: the framework of Lindland et al. in [17] and Pohl's framework in [20]. The comparision has shown that the frameworks have different appearences and uses different terminology, but the deeper structures of the frameworks are quite similar.

The main objective of the paper has been to push our understanding of quality aspects in requirements engineering one step further. The comparison of the two frameworks has been useful in that respect. In particular, the concept of agreement in Pohl's framework has inspired us to look deeper into the social process of building a specification.

In contrast to the previous version of the framework of Lindland et al. we are now able to discuss the quality of models where different social actors are developing their submodels based on individual domain knowledge. Furthermore, the process of merging different viewpoints is defined as contributing to social quality. Here, agreement among the actors is the major goal.

Table 1 shows an overview of the goals and means of the extended framework. The overview is based on a similar one in [17], but has been extended as discussed above.

Quality types	*Goals*	*Means*	
		Model properties	Activities
Syntactic q.	Synt. correctness	Formal syntax	Syntax checking
Semantic q.	Feasible validity Feasible compl.	Formal semantics Modifiability	Consistency checking Statement insertion Statement deletion
Perceived sem.q.	Perceived validity Perceived compl.		Statement insertion Statement deletion Audience training
Pragmatic q.	Feasible compr.	Expressive economy Aesthetics Executability	Inspection Visualization Filtering Diagram layout Paraphrasing Explanation Audience training Execution Animation Simulation
Social q.	Feasible agreement	Conflict modelling	Viewpoint analysis Conflict resolution Model merging

Table 1.: Framework for model quality

Although the framework contributes to our understanding of quality issues with respect to requirement engineering, the contribution so far lies on a high level of abstraction. There are several interesting paths for further work by which the framework can be refined to become more directly useful for requirements engineering practitioners. Among others, the follow areas need further exploration:

- *development of further product metrics:* In the current framework quality goals are mainly defined as the degree of correspondence between various sets. Future work should concentrate on developing quantitative metrics so that the quality of the model, audience, and the domain knowledge can be more explicitly assessed. Some initial efforts in this direction are reported in [13].
- *development of process guidelines:* The framework gives an overview of decisions that will have to be made in the requirements engineering phase. Further work should result in guidelines that practitioners may use directly in concrete projects.

Since semantic, pragmatic and social quality are in practice immeasurable, process heuristics may be a more interesting issue to pursue than product metrics.

References

1. P. Berger and T. Luckmann. *The Social Construction of Reality: A Treatise in the Sociology of Knowledge.* Penguin, 1966.
2. B. W. Boehm. Verifying and validating software requirements and design specifications. *IEEE Software*, 1:75–88, 1984.
3. S. Bråten. *Dialogens vilkår i datasamfunnet (In Norwegian).* Universitetsforlaget, 1983.
4. J. Conklin and M. J. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4):303–331, 1988.
5. B. Dahlbom. The idea that reality is socialy constructed. In Floyd et al. [8], pages 101–126.
6. A. M. Davis. *Software Requirements Analysis & Specification.* Prentice-Hall, 1990.
7. N. E. Fenton, editor. *Software Metrics — A Rigorous Approach.* Chapman & Hall, 1991.
8. C. Floyd, H. Züllighoven, R. Budde, and R. Keil-Slawik, editors. *Software Development and Reality Construction.* Springer Verlag, 1991.
9. R. Gjersvik. *The Construction of Information Systems in Organization: An Action Research Project on Technology, Organizational Closure, Reflection, and Change.* PhD thesis, ORAL, NTH, Trondheim, Norway, 1993.
10. J. Goguen. Requirements engineering: Reconciliation of technical and social issues. Technical report, Centre for Requirementss and Foundations, Oxford University, Cambridge, England, 1992.
11. U. Hahn, M. Jarke, and T. Rose. Group work in software projects: Integrated conceptual models and collaboration tools. In S. Gibbs and A. A. Verrijn-Stuart, editors, *Multi-User Interfaces and Applications: Proceedings of the IFIP WG 8.4 Conference on Multi-User Interfaces and Applications*, pages 83–102. North-Holland, 1990.

12. M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, and Y. Vassiliou. Theories underlying requirements engineering: An overview of NATURE at genesis. In *Proceedings of the IEEE International Symposium on Requirements Engineering (RE'93)*, pages 19–31, 1993.
13. J. Krogstie, O. I. Lindland, and G. Sindre. Defining quality aspects for conceptual models. In E. D. Falkenberg et al., editor, *Information Systems Concepts, Proc. ISCO3, Marburg, Germany.* North-Holland, 1995.
14. C. H. Kung. An analysis of three conceptual models with time perspective. In Olle et al., editor, *Information Systems Design Methodologies: A Feature Analysis*, pages 141–168. North-Holland, 1983.
15. J. C. S. P. Leite and P. A. Freeman. Requirements validation through viewpoint resolution. *IEEE Transactions on Software Engineering*, 17(12):1253–1269, December 1991.
16. P. Lindgren ed. A framework of information systems concepts. Technical Report Interrim report, FRISCO, May 1990.
17. O. I. Lindland, G. Sindre, and A. Sølvberg. Understanding quality in conceptual modelling. *IEEE Software*, pages 42–49, April 1994.
18. J. D. McGregor and T. D.. Korson. Integrated object-oriented testing and development processes. *Communications of the ACM*, 37(9), 1994.
19. J. W. Orlikowski and D. C. Gash. Technological frames: Making sense of information technology in organizations. *ACM Transactions on Information Systems*, 12(2):174–207, 1994.
20. K. Pohl. The three dimensions of requirements engineering: A framework and its applications. *Information Systems*, 19(3):243–258, April 1994.
21. H. Rittel. On the planning crisis: Systems analysis of the first and second generations. *Bedriftsøkonomen*, (8), 1972.
22. G. C. Roman. A taxonomy of current issues in requirements engineering. *IEEE Computer*, pages 14–22, April 1985.
23. A. H. Seltveit. An abstraction-based rule approach to large-scale information systems development. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proceedings of the 5th International Conference on Advanced Information Systems Engineering (CAiSE'93)*, pages 328–351, Paris, France, June 8-11 1993. Springer Verlag.
24. J. Siddiqi. Challenging universal truths of requirements engineering. *IEEE Software*, pages 18–19, March 1994.
25. R. T. Yeh, P. Zave, A. P. Conn, and G. E. Cole Jr. Software requirements: New directions and perspectives. In C. Vick and C. Ramamoorthy, editors, *Handbook of Software Engineering*, pages 519–543. Van Nostrand Reinhold, 1984.