THREE ASPECTS OF INTELLIGENT COOPERATION IN THE QUALITY CYCLE[1]

MATTHIAS JARKE , MANFRED A. JEUSFELD, PETER SZCZURKO

Informatik V, RWTH Aachen

Ahornstr. 55, 52056 Aachen, Germany

The WibQuS project investigates distributed computer and communication support for Total Quality Management in industrial organizations. An interdisciplinary study of this task reveals three aspects of intelligent cooperation required from such a system which we call conceptual, technical, and social integration. We use the repository standard IRDS to characterize these aspects. A solution concept derived from this characterization employs advanced conceptual modeling techniques to derive interoperability among technical subsystems and coordination technology for the human subsystem semi-automatically while taking important social factors into account.

Keywords: quality management, CAQ, TQM, quality cycle, repository standard IRDS, intelligent cooperative IS, evolution

1. Introduction

Product and service quality have become decisive competitive factors in modern industry [4]. It has been recognized that quality and productivity need not be competing goals, because much productivity is lost through rework of poor-quality products or even recall actions after product delivery. However, the potential synergy between quality and productivity can be realized only if quality is produced rather than controlled into the product. This requires a concept of Total Quality Management [13] in which the voice of the customer reaches even the most distant corners of the organization and its affiliates [6].

---

[1] This paper was presented as a keynote talk at the First International Conference on Intelligent and Cooperative Information Systems, Rotterdam, May 1993.

The voice of the customer changes as frequently as your competition´s reaction to it. Information technology can open up additional channels for the necessary communication, coordination, and collaboration of the organizational reactions to such new opportunities or threats. This paper makes the claim that intelligent and cooperative information systems, integrating humans and computerized components in an effective manner, can contribute significantly to this aim. This claim is the result of experiences in a major interdisciplinary research effort on computer-supported Total Quality Management, called WibQuS ("Wissensbasierte Systeme in der Qualitäts-Sicherung" = Knowledge-Based Systems for Quality Assurance).

WibQuS attempts to interrelate successful individual quality management methods occuring in various phases of a product life cycle into an architecture optimized for quality information flow throughout the organization. To get an understanding of the need for intelligent cooperation, Section 2 reviews the major methods supported in WibQuS and points out the richness of their interrelationships which is currently not fully exploited. We identify the need for three facets of intelligent cooperation: concept integration to promote mutual understanding and process coordination; technical integration to enable remote and asynchronous collaboration; and social integration to ensure organizational implementation, adequate human-machine interaction, and adequate support of human-human interaction through machines.

A coherent solution to these cooperation problems requires a common framework. Section 3 shows that distributed repository models can be used for this problem characterization. Sections 4 to 6 elaborate our solution for each of the three facets of cooperation within this framework. Section 7 reports some initial empirical evidence conerning the validity of the approach and sketches some specific extensions and improvements derived from these experiences.

2. The WibQuS Project

In 1992, the German Federal Ministry of Research and Technology (BMFT) embarked on a programme to promote Total Quality Management in German industry. Within the BMFT programme, the goal of the WibQuS project is to support collaboration within and across interdisciplinary quality teams throughout the organization and beyond through integrated information and communication technology, especially knowledge-based systems, CSCW, and repository technology.

The WibQuS consortium consists of six engineering institutes, a social science group responsible for ergonomic and organizational aspects, and our (computer science) group. The goal is to design a structure for collaboration among different quality methods, and to demonstrate it with a prototype on significant real-world examples taken from the household, automobile, and service industries.

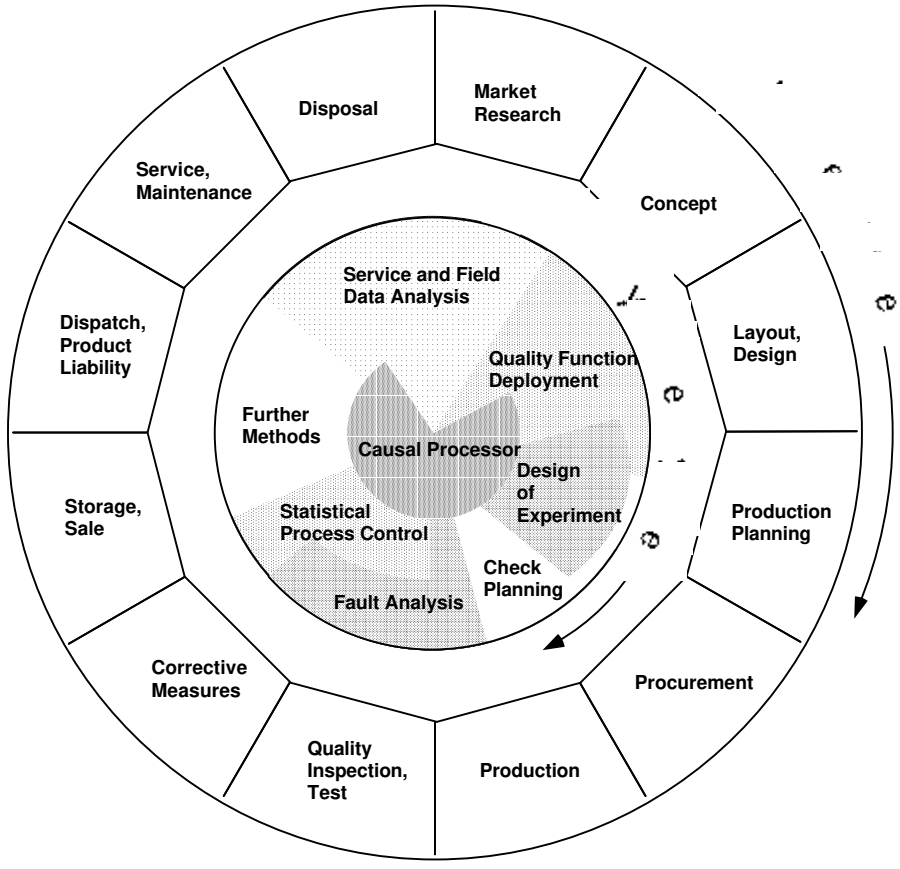

Fig. 2.1: The Quality Cycle in WibQuS

A lot of methods and tools have been proposed to ensure product and process quality along the product life cyle [15]. In European Norm EN 29 004, they are structured according to a so-called Quality Cycle that accompanies the product life cycle (Fig. 2.1). Within WibQuS, each of the engineering groups investigates computer support for an individual promising quality method within the cycle. The social science group advises on, and evaluates adherence to human-factors guidelines and organizational implementation strategies for such systems.

The specific methods investigated in WibQuS and their role in the quality cycle are shown in the center part of Fig. 2.1:

- *Quality Function Deployment* (QFD) is a systematic and structured approach to ensure consideration of customer satisfaction in development, construction, and work preparation. The method assesses the relative importance of customer criteria and

relates them to product or process properties in a so-called House-of-Quality matrix. A set of such matrices are filled by several interdisciplinary and mutually overlapping teams. In supporting the QFD method by computer, one of our goals is to exploit the experiences of already finished product development phases and information from the service field to support team sessions.

- *Computer-Aided Design of Experiment (CADOX)* is used to plan experiments that evaluate a product or process before the actual production phase. Knowledge-based systems advise the user in choosing a good test sequencing strategy and appropriate statistical methods, focusing on important issues, as defined in QFD or collected from process traces of similar products. Again, interdisciplinary teams are needed to bring all the relevant experiences to bear.

- *Extended Statistical Process Control* (XSPC) is a method for associating, with the production process, an accompanying control process intended to limit the variance in product or process parameters. A knowledge-based XSPC system assists the user in deriving the control strategy from the normal process definition, taking into account results from the earlier phases.

- *Computer-Aided Fault Analysis (CAFA)* is essentially a diagnosis and repair system for ongoing production processes. In addition to analysing actual faults and proposing countermeasures, CAFA also collects statistical data for longer-term process improvement and conducts active audits, not only when there are obvious problems. The basis for these investigations is input from XSPC as well as documented product information from manufacturing and assembly.

- *Test Planning* includes a family of methods applied during several phases of the product life cycle to plan for detailed analyses of some quality aspect of product or process. In essence, quality criteria given by the customer are mapped to measurable, often geometric technical criteria, measurement instruments are selected, sample size and timing of measurements are determined, and special-purpose software for conducting the actual test semi-automatically is developed.

- *Service and Field Data Recording and Analysis* with the help of a dialogue-oriented knowledge-based system offers after-sales feedback on products. Statistical service information as well as informal registered data are analyzed, in order to be integrated in the planning phase of the quality process. Besides data capturing and analysis, this information also improves maintenance and service knowledge.

- The *Causal Processor* is a general service that maintains cause-effect relations between construction, manufacturing and assembly for all the other tools, and is in particular applicable to the analysis of the causes and effects of possible faults in the early planning phases (FMEA).

Each individual method is used by teams which can be highly interdisciplinary. Especially in the case of QFD, CSCW techniques are probably the most important supporting information technology.

Even the brief descriptions above show the additional need for collaborating *across* methods. For example, service data analyses should feedback rapidly not only into the criteria catalogue for Quality Function Deployment of the next product version, but could also suggest different statistical process controls within the same cycle, or even change the definition of that process.

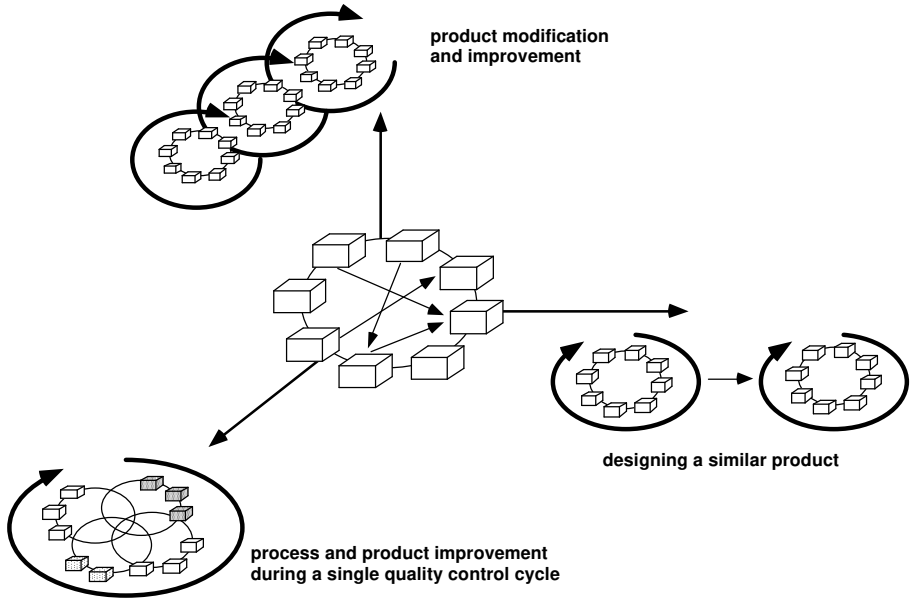

Fig. 2.2: Interrelating quality cycles

Total quality control also means minimizing the effort spent on new products from design to service by reusing experiences from former cycles. We distinguish three contexts in which the quality control cycle can be seen (Fig. 2.2).

- Within a single quality cycle, each step of the process has to provide feedback to earlier steps as a basis for improving process and product as soon as possible, in order to reduce time and costs. An investigation of causal dependencies between tasks and methods is very important because we need the real data flow within the cycle to determine responsables able to effect improvements. Conversely, each participant in the cycle should be able to obtain information from subsequent processes to verify his own work through analyzing data as feedback information.
- The answer to new customer requirements is the development of product varieties. Product improvements, additional and special functionalities realized in product

versions use cycles very similar to their predecessors. Both positive and negative experiences should be forwarded to later cycles such that products and processes are improved considering the inadequacies during the already finished cycles.

- If there exists a functional correlation of products (e.g., a washing machine may be similar to a dishwasher) it may be useful to reuse information from similar product cycles. In this case it is more important to exchange experiences made in pre-existent situations.

Integration does not only mean support for the technical data flow between processes and persons but also the exchange of experiences. Two main problems include the *spatial* and the *conceptual distance*. These distances have to be bridged to ensure a straight, goal oriented development of a quality information system. The spatial distribution of involved groups and systems as well as the large amount of data require a distributed information system architecture.

The same reason is responsible for the need of a convenient concept for distributed system development. The main problems during the modeling process, the design of the information system and the realization of the basic architecture from the technical and organizational point of view will be discussed in the subsequent sections.

3. Intelligent Cooperation in the IRDS Framework

To solve the different problems of intelligent cooperation among teams, a coherent conceptual, technical, and social framework is required. The framework we are going to advocate here is based on repository notions. More precisely, we shall start from the Information Resource Dictionary Standard (IRDS) as recently defined by ISO [7] and shall demonstrate how it can be used and specialized to integrate the three aspects of intelligent cooperation. Figure 3.1 sketches this framework.

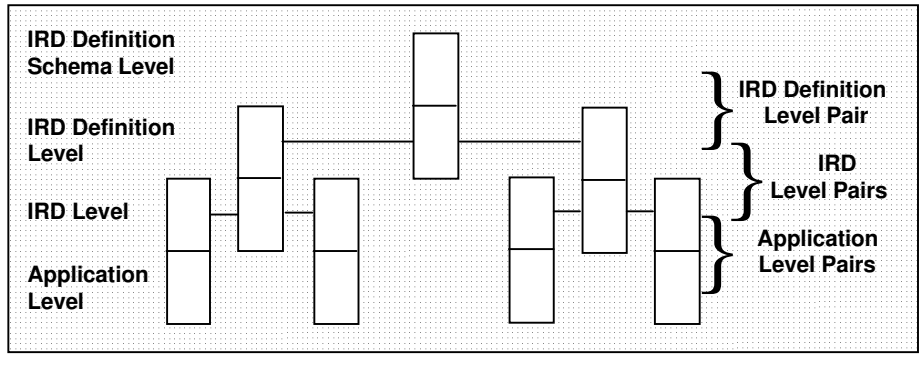

Fig. 3.1: ISO-IRDS architecture

From earlier work on meta information systems, ISO IRDS has adopted the basic organization along the classification abstraction of semantic data models. In a layered type universe of four levels, level *n+1* (called the *defining* level) constitutes a type system for level *n* (the *defined* level). In other words, it defines the language in which level *n* can be specified. The four levels are the *Application Level*, the *IRD Level*, the *IRD Definition Level*, and the *IRD Definition Schema Level*.

ISO IRDS additionally promotes a so-called "level pair" architecture in which the relationships among the four levels are defined by a hierarchy of three SQL database specifications on top of each other. As a consequence of the level pair concept, both application systems and dictionary systems can be distributed and heterogeneous.

Application Level Pairs correspond to the actual applications to be integrated. Dictionaries or *IRD Level Pairs* contain as their data the schemata of the application level pairs, and definitions of the underlying data models as their schemata. A single *IRD Definition Level Pair* contains descriptions of the dictionary schemata as its instances and a pre-defined way of defining such schemata as its own schema, formulated in the standard as a set of SQL tables.

The IRDS standard demands some services by which schemata of the defined level pair can be automatically generated from the data of the defining level pair, and for the handling of working sets and versions. In the following, we interpret the problems of quality management in this framework and categorize the different kinds of support for intelligent cooperation required. The result is shown in Fig. 3.2.

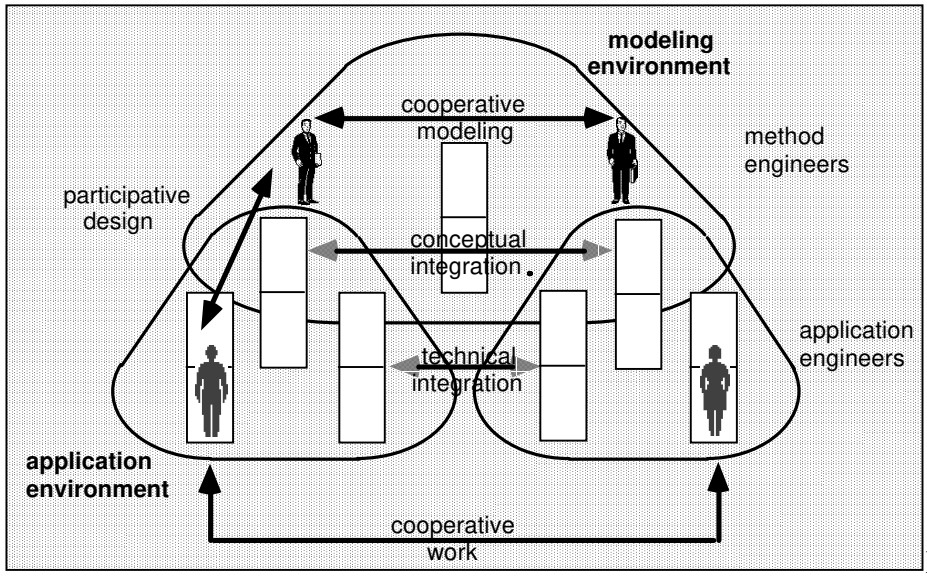

Fig. 3.2: Aspects of intelligent cooperation in the IRDS framework

The first observation to be made is that Fig. 3.1 can be grouped into hierarchically organized pairs of level pairs, indicated by three overlapping triangles in Fig. 3.2. We call the upper one the *modeling environment*, the lower ones the *application environments*. People involved in the modeling environment are called *method engineers*, people in the application environments are called *application engineers*. In WibQuS, we use this framework in the sense that each individual quality method is considered a semi-autonomous application environment.

Now we are in a position to look at the required facets of cooperation in such a framework. We can distinguish more technical aspects (shown as gray double arrows in Fig. 3.2) and social aspects (shown as black double arrows).

On the technical side, we first note the problems of bridging different terminologies and the need for process coordination across methods. This c*oncept integration* takes place by relating the individual dictionary schemata in the IRD Definition Level Pair database. Moreover, the technical components of distributed application systems must be networked and enabled to interoperate (call each other, exchange data, etc.), even if they are associated with different dictionaries. We call this *technical integration* .

The black double arrows in Fig. 3.2 show that *social integration* involves at least three different problems. First, method engineers have to collaborate on the conceptual integration task in a *cooperative modeling* effort. In WibQuS, we have gone through this exercise for relating the six methods mentioned earlier.

Second, it must be noted that the application engineers form part of the system to be designed. Their conceptual "schema" is represented in the dictionary, their actions at the application level and thus outside the scope of the system. People do not like to be planned at, and the philosophy of Total Quality Management strongly emphasizes bottom-up information flow. One implication is that application engineers should actively participate in the design and re-design of method models, another that method models should not unduly restrict the freedoms of the application engineer.

Third, our approach is creating a computer-supported cooperative work environment among the application engineers. The usual human factors criteria of CSCW systems apply, but additionally new organizational communication paths are opened up which must be carefully considered.

Summarizing the discussion so far, we have found the IRDS framework suitable to represent the three kinds of intelligent cooperation required in the quality cycle. For modeling purposes, it is necessary to express these three aspects directly as meta level concepts at the IRD Definition Schema Level. In WibQuS, we chose a meta model which represents each of the three aspects by a single concept: the conceptual aspect by *method*, the technical aspect by *object*, and the social aspect by *agent* (Fig. 3.3).

The three aspects are related by a fourth underlying concept, the *task*. Tasks take objects as input and produce new objects. For instance, the quality definition task takes customer wishes as inputs and produces functional product properties. Tasks are supported by methods such as QFD. Through their input and output objects, tasks indirectly formalize the possible information flows between methods. All three categories are related to human *agents* (e.g., a team for QFD). An agent *knows* methods for *solv*ing tasks on objects they *own*. The *includes* attribute indicates that all concepts can be hierarchically decomposed (e.g., a task may have sub-tasks).

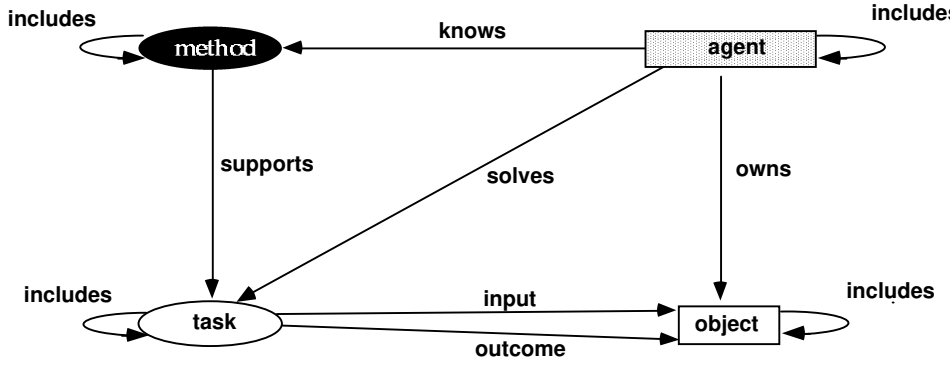

Fig. 3.3: Meta model for facets of integration (IRD Definition Schema Level)

4. Concept Integration and Cooperative Modeling

Different background makes experts rarely speak the same language even if they talk about the same domain. Simple conflicts are homonyms (experts use the same term for different objects) and synonyms (method engineers use different terms for the same object). A more severe problem is to focus the attention of the experts to relationships between the different tasks of the quality cycle, and to reason on the consistency and completeness of the conceptual model. Currently, there only exist natural language descriptions of the quality cycle which hardly suffice for our objective. An additional task is the coordination of the distributed modeling process.

We use two main ideas to solve the above problems. First, a meta language framework is set up which guides the way how method engineers define their partial conceptual models. Second, a computerized modeling tool capable of integrating distributed information sources addresses the coordination problems.

The language framework starts at the IRD Definition Schema Level which contains the "grammar" for defining the tasks, objects, methods, and agents participating in the

quality cycle, as shown in figure 3.3. The nodes represent concepts (classes) connected by links (attributes, properties, relationships).

Distributed conceptual modeling would lead to chaos if the method engineers encode their part models without coordination and supervision. One can well apply the thesis of TQM: errors made in this early phase are the most costly ones. Thus, early detection of errors and conflicts is crucial for the overall goal of intelligent cooperation in the quality cycle. A computerized tool which supports the modeling phase may help. The tool has to fulfill the following requirements:

- Method engineers use different (graphical) representations for conceptual models like SA, SADT, ER, etc. A tool for distributed modeling should allow to "view" the concepts in any of these representations.

- It is too costly to develop the conceptual model in joint sessions of all method engineers. The tool should help integrate local developments into the global conceptual model in an ongoing, distributed fashion.

A final goal of our work is, of course, to generate much of the technical integration automatically from the conceptual integration; at least, both kinds of integration must be maintained consistent with each other. This can, unfortunately, not be expressed within the SQL framework of the IRDS standard. Therefore, WibQuS adopted a conceptual modeling language, *Telos* [11], in which it is possible to express relationships across more than one level of the DBMS. *ConceptBase* [8], an implementation of Telos, allows the simulation of a complete IRDS environment in a single system, as indicated in Fig. 4.1.

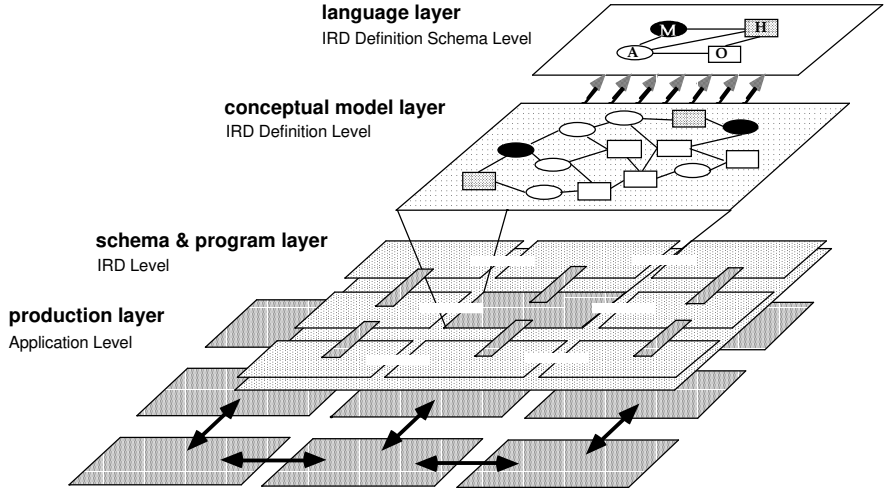

Fig. 4.1: The four IRDS levels simulated in the ConceptBase environment

ConceptBase allows wide-area distributed access to the knowledge base containing the conceptual model. Furthermore, Telos covers all four levels of the IRDS hierarchy by its meta-class facility. Deductive rules and integrity constraints can be user-defined for purposes of consistency checking, message passing between method engineers, and view definition. Graphical languages used by the engineers (e.g., SA/ER or IDEF1X) can be defined within Telos and related to the common conceptual model.

Figure 4.2 shows an excerpt of the conceptual model developed in the WibQuS group. It describes an information flow between the QFD and XSPC methods. The task of planning the process control requires as input the definition of the production process. This definition includes information about process parameters generated within QFD. The classification as mainstream task indicates that this information flow is crucial for fulfilling the task.

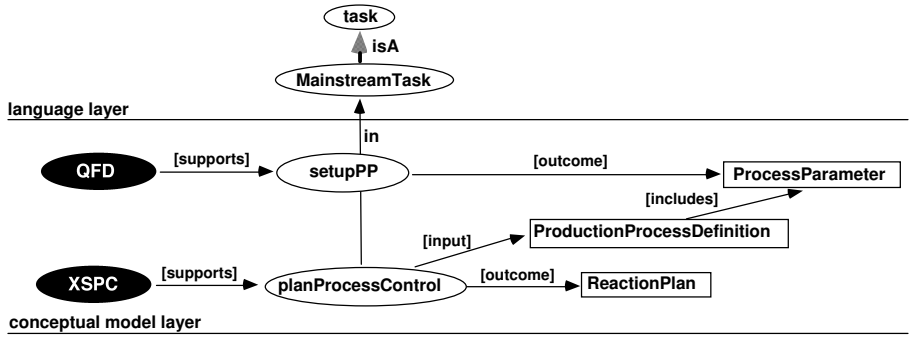

Fig. 4.2: Excerpt from the conceptual model

5. Technical Integration Through Interoperating Database Services

In WibQuS, methods of the quality cycle are supported by computerized tools. The information flow has to be supported by *data and control flow* between the tools. While the conceptual model of the quality cycle defines the possible information flow between methods, it does not say anything how the information is represented in data structures, nor which protocols for data exchange are used. These questions have to be solved in a *distributed* and *evolving* setting. For example, it should be possible to attach the database of the CAD department without re-implementing the interfaces of all tools.

We use a *trader architecture* for integrating the distributed tools (Fig. 5.1). The trader maintains a copy of the first two IRDS layers in its knowledge base. Thereby, one can use the conceptual model of the quality cycle as schema for describing the data flow in the same we as we used the meta model for defining the conceptual model. The task of the trader is to maintain the declarations for the data flow between the tools. The *objects* of the conceptual model are instantiated to data definitions, and the tasks are instantiated to queries, views, and triggers controlled by the trader.

The role of the trader in the environment is to set up a "market" for information. The tools enter the market by connecting to the trader´s database, and asserting their identity, the services/goods they provide or request. The trader then proposes how to deliver the offered services/goods to the requesting tools. The scenario is open to evolution: if a tool leaves the market, the trader can determine the consequences to other tools, e.g. some requests are no longer satisfiable. On the other hand, services provided by new tools can lead to a recomputation of the deliveries. Ideally, the trader makes the tools independent of the actual configuration of the environment.

In the WibQuS environment, technical integration is based on the industry standard SQL. Actually, a commercial product for distributed SQL access is incorporated. Technically, the architecture conforms to a loosely-coupled federated database system [1]. SQL statements for the data flow (table declarations, queries, views, triggers) are generated from abstract specifications in the conceptual design.

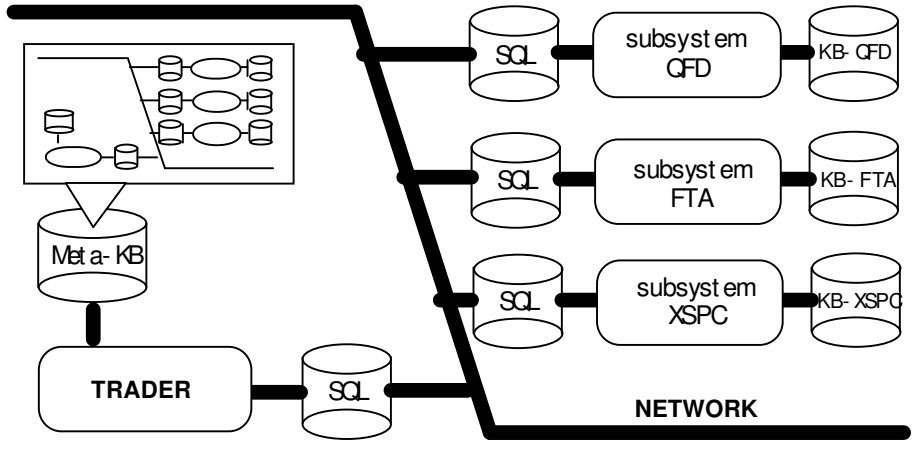

Fig. 5.1: Trader architecture for technical integration

If, for example, the method engineer for XSPC defines that the task of setting up a XSPC chart must use information about the importance of process parameters, the trader can generate a trigger that passes changes to the relation of important process parameters to the XSPC database system. Here, the power of a language like Telos becomes operational: Telos allows to define meta-rules spanning more than one abstraction level, esp. talking about items in the tools databases (production layer) in the conceptual model (conceptual model layer). From such logical rules, one can automatically derive specialized rules for the application schemata, and then reformulate them as SQL queries, views, or triggers [9].

In the WibQuS environment, each tool gets a standard SQL database system as its interface to the outside world. This gives services like recovery, atomic transactions, network transparency etc. Here, we concentrate on the role of the trader and how this

tool facilitates the data flow. One has to distinguish the conceptual model from the data and program layer. The former defines the information flow in the quality cycle per se; the latter maps the conceptual model to concrete tool specifications for a given application environment, e.g. quality management for a car manufacturer. At this level, the part database schemata and the programs operating on them have to be described (compare Fig. 5.2 and 4.2).

The database schemata for process parameters, part hierachies of products, etc. are currently being standardized in the STEP effort [17]. This avoids a re-invention of database schemata for large portions of the information; only some extensions had to be made to represent information like failure-cause relationships which are specific to quality management.

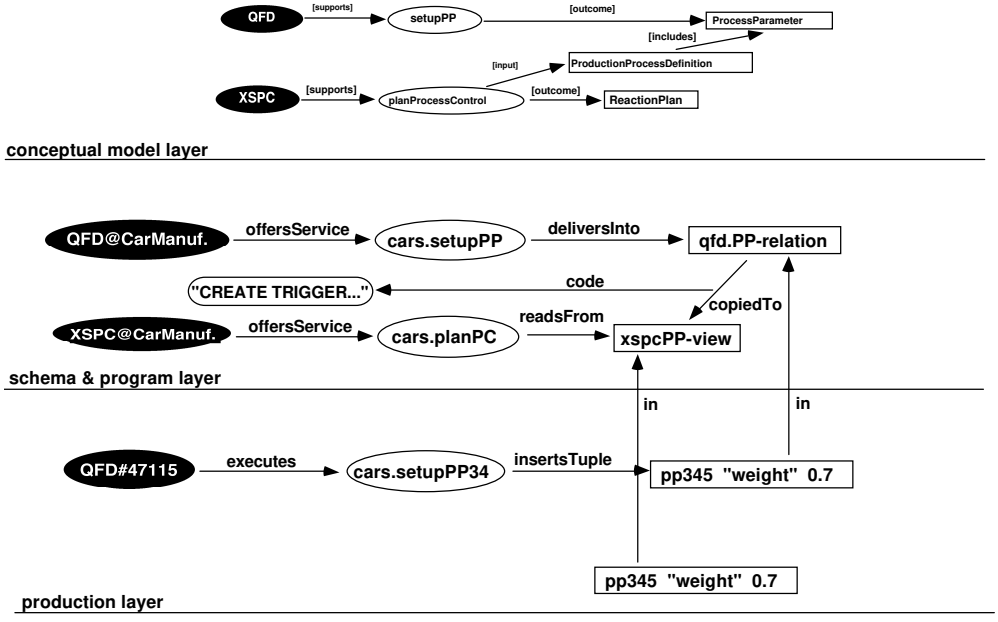

Fig. 5.2: The trader´s view on a data flow

The mapping of the information flow onto a data flow by formulas across more than one abstraction level is demonstrated in the following example:

```
forall m//method t//MainstreamTask
   (exists r//object
   (t [input] r) and (m [supports] t))
   ==> CreateDataFlow(m,t,r)
```

This general policy rule is written in terms of the language layer (meta model) but makes a statement about the schema and program layer across the conceptual model layer. Let´s

assume that m is an implementation of some method, and t be a task implemented in this method that is categorized as a mainstream task. Then a data flow is created that passes each input object of the task to the program m. The clause "x//C" stands for "(x in c) and (c in C)". Here, the variable c is an anonymous placeholder for any instance of the category C. The specific advantage of policy rules is their independence from the conceptual model. They are true for any conceptual model. Method engineers are liberated from the task of handcoding specialized rules for all such objects c.

In Fig. 5.2, XSPC@CarManuf is an example of such an implementation of a method. It realizes the mainstream task planProcessControl by its service cars.PlanPC. Since ProcessParameter is an indirect input of planProcessControl and qfd.PP-relation is one of the data structures implementing this information object, a solution CreateDataFlow(XSPC@CarManuf,cars.planPC, qfd.PP-relation) is derived. This solution is interpreted by the trader as a call to a code generator which produces code for a SQL trigger writing into the relation xspc.PP-view. The bottom layer of Fig. 5.2 records individual operations. The tool instance QFD#47115 (a software process of the kind QFD@CarManuf) executes the task cars.setupPP34 which inserted a tuple into qfd.PP-relation. This induces the execution of an SQL trigger which stores a copy of the tuple in the database of the XPC tool.

With respect to evolution, the above example shows that data flows can be implemented just by stating that a particular task is in the "mainstream" of the quality cycle. Since the trader is part of the distributed environment, changes to the conceptual model (e.g., insertion of a new method with its information flow to and from its neighbor methods) lead to an automatic update of the data flow procedures (SQL triggers, views etc.) located at the individual tools. Moreover, even the policy rules can be evolved! In the above example, the deletion of the rule for CreateDataFlow means that all the code generated for data flows due to mainstream tasks has to be retracted.

Similar to the above approach, the deductive database LDL has been proposed to ease the evolution problem [18]. However, evolution is described within the framework of a centralized deductive database. The trader in WibQuS manages all four IRDS levels where the two bottom levels are distributed over a network. The trader may also been seen as a means to encapsulate the WibQuS tools as active objects in the sense of [3]: the tasks in the schema and program layer play the role of an interface to the tools.

6. Social Integration Through Computer-Supported Teamwork

Total Quality Control has come to have a major impact on management and engineering practices as a major new business strategy [5]. The crucial question is how the personnel and organizational aspects relate to the technical aspects of a quality information system. Any technical artefact would almost certainly fail if it neglects this question. Traditional views of the quality improvement process have to change according to the TQM idea

that organizational, personnel and technical requirements equally influence the organization-wide activities.

There can be contradictions in basic requirements between technological and organizational innovations [2]. The evolution from hierarchical to lean structures in middle management causes changes in work structure, information technology and communication structures because of their knowledge-intensive work.

In addition to the establishment of customer-oriented quality disciplines in the marketing and engineering functions as well as in production, TQM means that each employee of an organization, from top management to the production-line worker, will be personally involved in quality control. The goal of social integration is concerned with organizational structures, individual responsibilities and authorities in every development phase of the TQM system. Thus, we have to emphasize the humans as those who bring the quality processes to life.

System users want to have different views on data and on the processes. A distinction between users, experts and engineers is necessary because each of their views will influence the basic system architecture. We have to meet the quality requirements of each organizational group - including their necessities, abilities, qualifications and desires. The acceptance of the quality control system will decisively depend on the satisfaction of their demands. An implication from these considerations is that it is impossible to separate the design of the man-machine interaction from the design of working and organizational conditions.

The analysis of existing information flows between agents in the quality cycle can lead to an appropriate reorganization. As an example, we established the importance of feedback in two different views. Feedback in the form of customer complaints and failure data from the field leads to analyzing the causes, so that proper corrective action can be taken. Feedback within the quality processes helps identify social requirements considering personnel qualifications or work flow structures. The explicit modeling of feedback can be seen as a control circuit between personal and organizational structures.

The integration of the agent as a central component of the WibQuS conceptual model (see Fig. 3.1) leads to a comprehensive view of the integration process. At the IRD definition level we represent the relationships of agents in the whole process as well as the relations and roles between agents. Teams can be regarded as a union of human and technical agents where we can assign their roles and tasks on an abstract level. In our framework we distinguish between three classes of teams, as shown in Fig. 3.2.

- The framework for an integrated system is developed by a group of method engineers. Though the modeling process requires regular meetings, it is necessary to support distributed work in an appropriate way. The evolution of a common model and thus a

conceptual schema of cooperation and team structures should be made in a single database, considering the local distribution and asynchronous work of the engineers. Any part of the model will be integrated with the already stored parts. It makes sense to use the database environment itself to solve the problems which occur if there are conceptual contradictions, misunderstandings and discrepancies. How WibQuS handled arrangements, comments, negotiations and agreements is described in section 7.

- The second team consists of method and application engineers. The efficient vertical collaboration within such a team demands that the members know each others´ work. This means that the delegation of a task is coupled with the knowledge which qualifications and which effort are necessary. On the other hand, the persons in charge should know about the effects of faults. They should have the possibility to influence the process or the conditions leading to the faults or at least the reactions to fault cases. The participative approach means that there is no strict division of labor. In this way competences and responsibilities can be seen in a new sense. Each team member is not only responsible for her own work, but also for the quality of the team work. Empirical studies show that an extension of work scopes will have positive effects on process and product quality [19]. In the WibQuS model, we can represent team structures in an appropriate manner. An investigation of these team structures according to the task flow within the product life cycle can be used to determine possible participative influences. It can also.detect new information flows which could improve products and processes. Information flow from the execution level of the process to the definition level can cause changes in the design of task and method modeling.

- Application engineers establish the third class of teams. This horizontal integration can be achieved through job enlargement. People with monotonous activities should change their work systematically. This means they have to care about a comprehensive view of processes and activities. The application engineer should know which are the preceding and which are the following working steps, defining the team she is working in. She should also know about the final goal or the end product and what is her contribution. Therefore one of the goals in WibQuS is to give adequate system help for an optimal coordination of successive working steps. Application engineers want to have their own views on data and processes as well as method engineers, experts and managers. They want to look at information belonging to their own work as well as at information belonging to the whole team. This requires appropriate data filters and special dialogue forms of the integrated quality system which will be developed in WibQuS.

The modeling of agents in the quality information system has some obvious advantages in answering questions such as:

- process status: who is involved in the quality process ?

- localization and classification of knowledge: who can solve a task ?

- accountability: who is responsible for the correct and efficient execution of a task?

- qualification: what additional training and experiences are needed?

But there are serious privacy and labor-relationship questions involved with such data which need substantially more investigation. Moreover, the added value of more detailed agent models is rapidly decreasing as such models get overly detailed.

7. Experiences and Extensions

The proposals made here for the three facets of intelligent cooperation in the quality cycle are being evaluated with the specific methods and demonstration applications of the WibQuS project, both at the method engineering and at the application engineering level. Below, we describe the experiences gained with our approach at the method engineering level and propose some extensions to the conceptual integration framework that are expected to remedy some of the shortcomings observed.

7.1 The WibQuS Method Engineering Case Study.

The WibQuS team of method engineers had the goal to define an integrated conceptual model of the quality cycle, and thus gave a good example of the cooperative modeling task in Fig. 3.2. The process of distributed modeling comprised the following phases:

Learning Phase: First, the meta model shown in Fig. 3.3 was proposed to the team and discussed at length. The result was a common agreement on the linguistic meaning of the four categories in the meta model. This initial learning phase included the joint development of key concepts in the quality cycle.

Installation Phase: The meta model and the key concepts were coded in Telos and made available by graphical ConceptBase clients on the workstation of each method engineer. Instead of pure Telos syntax, method engineers could use customized graphical representations of the conceptual model. The translation to Telos objects is described by deductive rules. User-defined integrity constraints are included into the meta-model for prohibiting certain design errors.

Coding phase: All method engineers modeled their method individually at their site, but in terms of the common meta model. The sites were distributed over Germany and connected by Internet. Local changes were passed to the central ConceptBase server where some immediate consistency checks with other submodels were possible. To avoid impeding progress in model development, such checks were formulated as parameterized views which could be re-evaluated on demand, rather

than as strict integrity constraints that would reject problematic updates. Informal email was used on the side to discuss questions of bilateral interest.

Review phase: Joint meetings took place every four weeks to discuss the current state of the conceptual model. They also served for delegating certain modeling tasks to subgroups, e.g. the modeling of the concept "property" as a product or process parameter. Stricter consistency checks were applied at this time, inconsistencies and missing connections corrected.

Voting phase: The meta model was extended by attributes for assigning "ok/not-ok" status to each concept, accompanied by text comments. The method engineers gave their votes to all objects of the conceptual model. The purpose of voting was the detection of semantic errors and incompletenesses not expressible in Telos. The result of the voting was discussed in a joint meeting, errors were corrected and the conceptual model officially accepted.

Coding and review phases were mixed while the voting phase took place in the last four weeks of the conceptual modeling. The resulting method models totalled about 400 concepts, with about five times that number of intermediate results and attached comments produced during the discussion.

7.2 Experiences with Cooperative Modeling in Telos

During the modeling process, a separate researcher partially observed individual and group activities and conducted regular interviews with all method engineers.

One of the major benefits of computerized support for distributed conceptual modeling was the acceleration of communication. Technically, an update from a local model to the global conceptual model (maintained by ConceptBase) took a few seconds even though partners were working at sites distributed across the country. One may argue that "paper" models can be distributed by fax or email in seconds, too. However, such documents are not easy to integrate because of unrestricted language and task complexity. Moreover, guidance by the formal language of objects, tasks, methods, and agents forced the method engineers to relate their part model to the part models of others. This triggered a lot of informal communication by phone, email, or ad-hoc subgroup meetings.

We also experienced some shortcomings of the approach. Time for learning the modeling framework should not be under-estimated, and some frequently needed modeling operations, such as renaming of concepts, are not supported by most DBMS´s, including ConceptBase. Finally, a step-by-step method for conceptual modeling, that tells where to start and how to proceed from a given intermediate state, is still lacking.

7.3 Extensions to the Meta Model

One of the main criticisms of the cooperative modeling procedure was that the initial meta model did not provide enough guidance for the modeling process. One way of solving this problem is to incorporate more knowledge about the domain at hand.

In the case of WibQuS, this is the domain of quality management. Therefore, it makes sense to extend the generic model of figure 3.3 by an explicit notion of quality *goal*. Goals can be seen as a specialization of the type *object* in figure 3.3, and *methods* within the quality cycle can now be categorized by how they contribute to achieving and redefining these goals, as shown in figure 7.1 [16].

Product methods drive activities in the normal product life cycle. *Control methods* measure achievement of goals by the product methods, through questions either about the product or about the process. *Improvement methods* close the quality cycle by helping users to exploit experiences or new goals to re-define the process itself.

The model in figure 7.1 has the interesting property that it subsumes three recent proposals made in the software process literature. It can therefore not only be used to give more semantics to our meta model of the quality cycle, but can also be used as a paradigm of gradually improving this model itself.

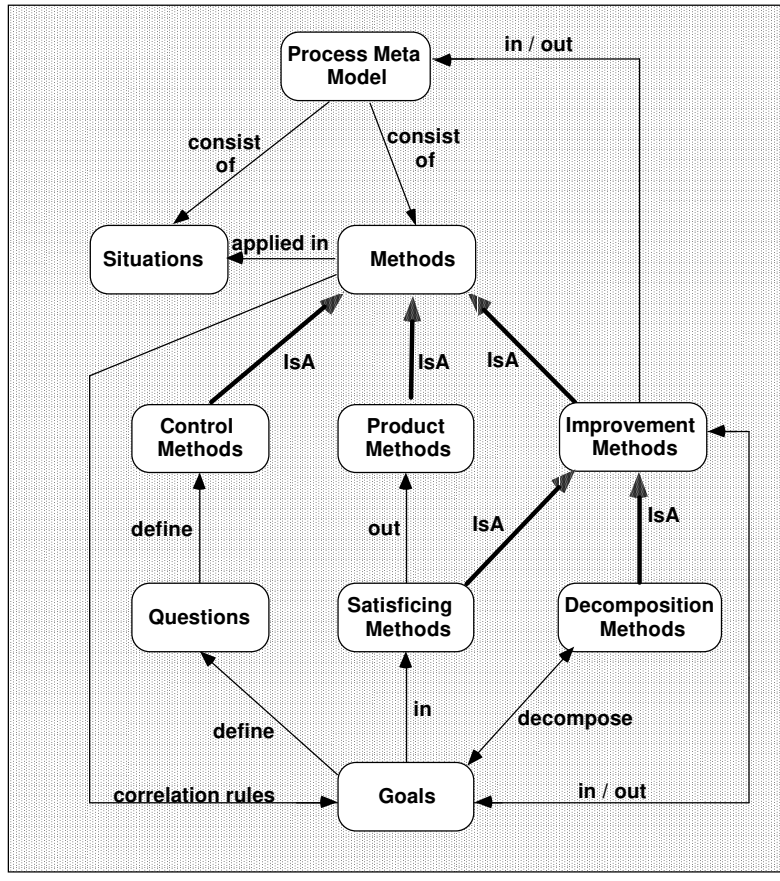

Fig. 7.1: Goal-oriented extension of the quality process meta model [16]

Mylopoulos et al. [12] combine goal *decomposition methods* and *satisficing methods* (both special cases of improvement methods) to access a database of product methods through AND/OR graphs. They also express, through *correlation rules*, that striving for one goal may positively or negatively affect other goals. Their model contains no feedback whether these goals are actually achieved.

Relating goals to operational control methods (metrics) through the intermediate construct of *questions* to be answered was inspired by the TAME model for improvement-oriented quality measurement [14]. The actual improvement cycle, including the modeling and re-design of methods, is not included in TAME.

Conversely, the idea of improvement methods that change the definition of the process itself (both the method definitions and statements about the situations in which they are applicable) appears in the PRISM model of change management in software environments [10]. PRISM says little about the quality relationships of the methods, nor does it contain control methods.

In figure 7.1, the three different kinds of methods have only been characterized by the way how they relate to goals. In [16], they are given additional semantics by differentiating the objects they work on, and the situations in which they are applicable. For example, we now have the framework for asking quality-oriented questions across methods, such as: what measures have been taken to support a certain goal/ to prevent a certain problem ?

This gives us a powerful framework for structuring, documenting, and supporting quality-oriented processes which could, for example, also be used in the case of legal problems associated with quality. Future experiments will have to show to what degree these theoretical advantages translate into practically more useful models of intelligent cooperation in the quality cycle.

Acknowledgments. This work was supported by the German Federal Ministry of Research and Technology (BMFT) as part of the WibQuS project (Fkz. PFT-92017-02QF30041). WibQuS partners include the groups of professors T. Pfeifer, W. Eversheim, and M. Jarke of RWTH Aachen, H. Weise of TU Dresden, G. Warnecke and K.J. Zink of Univiersität Kaiserslautern, and H.-J. Warnecke of Universität Stuttgart. Thanks are due to our cooperation partners, most importantly to the project coordinators Tilo Pfeifer and Robert Grob, and to our colleagues Rainer Gallersdörfer, Peter Peters, and Klaus Pohl who contributed to the model and its evaluation.

References

1. D. Bell, J. Grimson (1992) . *Distributed Database Systems*. Addison-Wesley.
2. H. Brinkmann (1991) . Technological and organizational innovation. *Proc. IFIP Conference on Information Systems, Work and Organization Design* (Berlin, July 1989), Elsevier Science Pub.
3. M.L. Brodie (1993) . The promise of distributed computing and the challenge of legacy information systems. In D.K. Hsiao et al. (eds.): *Interoperable Database Systems (DS-5)*, North-Holland, pp. 1-31.
4. W.E. Deming (1986). *Out of the Crisis.* Cambridge, Mass.
5. A.V. Feigenbaum (1991). *Total Quality Control*. McGraw-Hill.
6. J.R. Hauser, D. Clausing (1988). The house of quality. *Harvard Business Review,* May-June.
7. ISO/IEC 10027 (1990). *Information Technology -- Information Resource Dictionary System (IRDS) - Framework*. ISO/IEC International Standard.
8. M. Jarke, S. Eherer, R. Gallersdörfer, M. Jeusfeld, M. Staudt (1993). ConceptBase - a deductive object base manager. Aachener Informatik Berichte 93-14, RWTH Aachen Germany.
9. M. Jeusfeld, M. Jarke (1991). From relational to object-oriented integrity simplification. *Proc. 2nd Intl. Conf. Deductive and Object-Oriented Databases,* LNCS 566, Springer-Verlag.
10. N. Madhavji (1992). Environment evolution: the PRISM model of change. *IEEE Trans. Software Eng.* 18(5), pp. 380-392.

11. J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis (1990). Telos -- a language for representing knowledge about information systems. *ACM Trans. Information Systems* 10(4), pp. 325-362.

12. J. Mylopoulos, L. Chung, B. Nixon (1992). Representing and using non-functional requirements: a process-oriented approach. *IEEE Trans. Software Eng.* 18(6), pp. 483-497.

13. J.S. Oakland (1989). Total quality management. *Proc. 2nd Intl. Conf. Total Quality Management,* Oxford, UK: Cotswold Press, pp. 3-17.

14. M. Oivo, V. Basili (1992). The TAME goal-oriented approach. *IEEE Trans. Software Eng.* 18(10), pp. 886-898.

15. T. Pfeifer (1993). *Qualitätsmanagement*. Carl-Hanser-Verlag.

16. K. Pohl, M. Jarke (1992). Quality information systems: repository support for evolving process models. Aachener Informatik Berichte 92-17, RWTH Aachen, Germany.

17. N. Shaw (1991). *STEP Part 1 - overview and fundamental principles.* ISO TC184/SC4/Editing Document No. N-11, May 1991.

18. C. Zaniolo (1992). Intelligent databases: old challenges and new opportunities. *Journal of Intelligent Information Systems* 1(3), pp. 271-292.

19. K.J. Zink, A. Ritter (1990). Quality Circles - Instrument zur Einführung, Nutzung und partizipativen Gestaltung neuer Informationstechnologien? *Computer und Recht* 6 (2), pp. 147-152.