5

A quality-intensive approach to software development

Tervonen I., Kokkoniemi J.* and Smith G.***
** Department of Information Processing Science, University of Oulu, P.O. Box 333, FIN-90571 Oulu, Finland,*
phone: +358 8 553 1908, fax: +358 8 553 1890, e-mail: tervo@rieska.oulu.fi
*** Department of Computer Science, University of Manchester, P.O. Box 88, Manchester, M60 1QD, UK*

Abstract

This paper addresses the problem of quality-intensive software development, which invites a software engineer to evaluate the software from the quality viewpoint. Our approach is based on the GRCM (goal-rule-checklist-metric) model and on the use of three supporting tools, a graphical editor, the QuestMap tool and the Quality Training tool. We also illustrate our approach and the use of supporting tools in an example of a beer pump system.

Keywords

Quality model, quality goals, software development, organizational memory, CASE tool

1 INTRODUCTION

It is largely accepted that quality is the most important objective in software development. A software product of poor quality does not provide any benefit for its producer, but requires continuous correction, which entails a wasteful use of resources. The problem is that although we have quality models (e.g. the SQM model (McCall et al., 1977)) and standards (e.g. ISO 9126 (ISO, 1990)) which help us to set goals for a software product in terms of quality factors, we do not have an accepted procedure for reaching these goals in practice, i.e. for building high quality software.

This paper presents an approach to quality-intensive software development which invites and "forces" a software engineer to evaluate his artifacts (descriptions at different levels of abstraction and prototypes) from the quality viewpoint. We thus follow the statement of Crosby (1996) "Quality has to be caused not controlled", although our approach also provides an opportunity to inspect the artifacts. The present approach is based on our earlier experience and research regarding the QDA approach (Tervonen, 1994) and software design and inspection

D. Gritzalis (ed.), *Reliability, Quality and Safety of Software-Intensive Systems*

(Tervonen, 1996), where we have used a "GRCM (goal-rule-checklist-metric)" model to provide the structure for organizing quality information.

We first define our approach based on the GRCM model and then introduce the three tools used in it. Finally we illustrate our approach and the use of supporting tools in an example of a beer pump system.

2 THE QUALITY-INTENSIVE APPROACH

The GRCM model presented here provides a limited but appropriate common background for participants in software development and inspection. As depicted in Figure 1, the general quality goals are prioritized for a specific project and ultimate objectives for software development are set. The origin of the GRCM model lies in its hierarchy of quality models (cf. McCall et al., 1977). Although it is based on the SQM synthesis model (due to the numerous changes made to the Software Quality Metrics model (McCall et al., 1977), we call it a SQM synthesis), we also recognize its relation to the ISO 9126 standard (ISO, 1990).

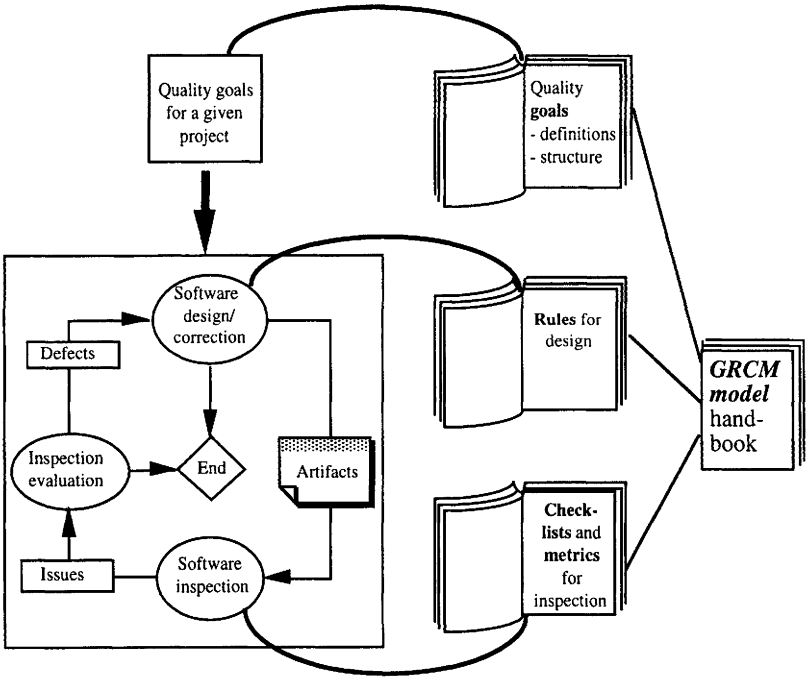

Figure 1 The quality-intensive approach based on the GRCM model.

The GRCM model has three links with the SQM synthesis, the goals correspond to factors (e.g. usability) and the rules to criteria (e.g. ease of use), and the metrics support quality measurement in both models. The goals are broken down into rules and further into checklists. The aim of the rules is to guide software engineers in software design, while checklists are generated from specific rules and used as guidelines by inspectors, to help of them check that these rules have been followed.

3 A TRIAD OF TOOLS

The triad of supporting tools consists of a type of graphical editor, QuestMap™ as a problem solving tool and the Quality Training tool as a supporting tool for the organizational memory. The interaction between the tools is shown in Figure 2. The Quality Training tool acts as a central tool and provides an introduction to quality concepts. It also supports the organizational memory by recording design solutions and their justifications for further reuse. As depicted in Figure 2, we record snapshots from the graphical editor and snapshots and discussions from the QuestMap tool. The organizational memory consists of collected records, and thus contains the design principles and practices of a specific company. The definitions, snapshots and QuestMap records are stored in the database and are accessed from the Quality Training tool and used for argumentation and learning purposes, etc.

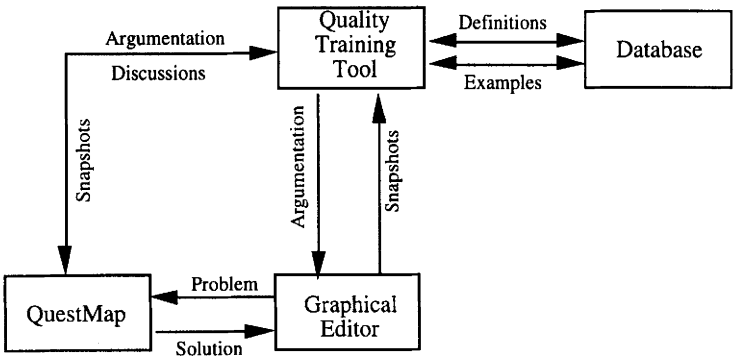

Figure 2 Interaction between the three tools and a database.

The snapshots made using the graphical editor show examples of design methodologies, while the other snapshots show QuestMap discussions on quality concepts. The graphical editor may be used to produce a design diagram, which may be influenced by the quality definitions and design practices recorded via the Quality Training tool. Difficult issues arising from the design may be discussed and solved using QuestMap. Important discussions can be saved and stored in the database for future reference.

We explain next the major characteristics of the Quality Training tool and the QuestMap tool. We do not consider the graphical editor, because its use is somewhat trivial.

™ QuestMap is a trademark of Corporate Memory Systems, Inc

3.1 Quality Training tool

The major ideas of the Quality Training tool can be traced to the "QDA tool" prototype (Tervonen, 1994), aimed at supporting quality-driven assessment (placing emphasis on the designer's justification by means of quality terms). The "QDA tool", implemented in a Smalltalk/V & Macintosh environment, illustrates new characteristics which advanced graphical editors, as parts of CASE tools, should support in the future. The Quality Training tool uses a series of definitions and examples for teaching the concepts inherent in high quality software. It can display the definitions of eleven concepts (goals) which contribute to software quality and the interrelationships between them. Figure 3 shows the main screen of the Quality Training tool, from which many of its features are accessed. The tool is implemented in Visual Basic and consists of a number of windows for displaying the information. Selections are made using menu items, option buttons or command buttons.

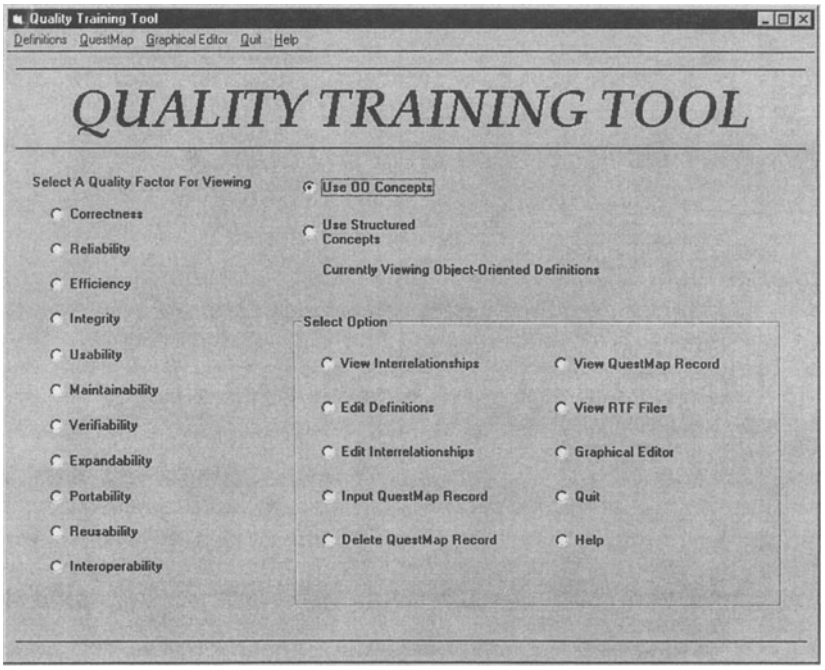

Figure 3 The main screen of the Quality Training tool.

The Quality Training tool provides further information in the form of rules which have to be followed to implement each goal and checklists giving instructions on how to check implementation of each rule. The GRCM model is used to organize the definitions and provide for their consistent use. The eleven concepts (goals) for which definitions, rules and checklists are given are: correctness, efficiency, expandability, integrity, interoperability, maintainability, portability, reliability, reusability, usability and verifiability. The interrelationships show how

concepts can be related, providing a more realistic view of software quality. An interrelationship may be advantageous to software quality, e.g. that between reliability and usability, or disadvantageous or conflicting, e.g. that between interoperability and reliability, in that emphasising interoperability can cause problems with reliability. All of the descriptions may be edited.

A screenshot from the Quality Training tool showing object-oriented definitions (which the tool is predominately concerned with) for the portability goal is shown in Figure 4. The tool's definition of portability is displayed in the top text box, and the rules for successful implementation of the concept are listed below it. A definition for a rule can be obtained by clicking the option button next to the name of the rule, which causes it to be displayed in the middle text box. Similarly, the checklists for the currently selected rule are listed and each one can be selected and displayed by clicking its option button.

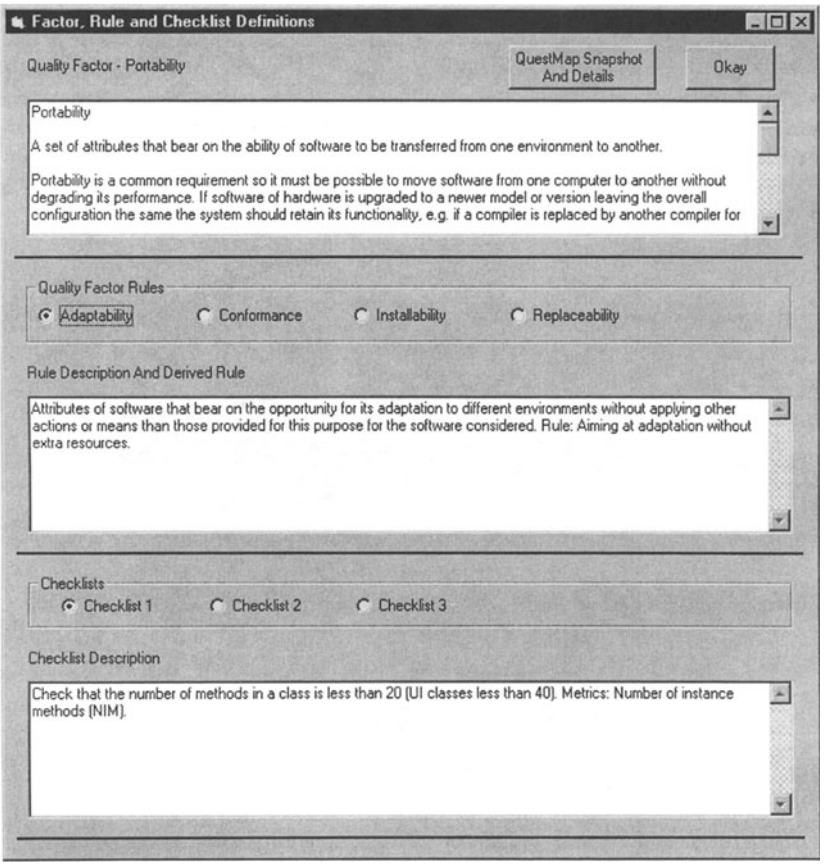

Figure 4 The definitions, rules and checklists for the portability goal.

The creation of an organisational memory involves capturing and organising information so that it can be easily located and retrieved in an understandable form for reference or reuse. This process extends and amplifies knowledge assets, as information will be retrievable and time will not be needlessly wasted discussing or researching a problem that has already been solved (Conklin, 1996). The ability of the tool to store, delete and display QuestMap discussions meets these requirements. The Quality Training tool provides a groupware aspect by allowing users to select the discussions they wish to store, delete or display.

3.2 QuestMap™

The technology recommended by Conklin (1996) for capturing information for an organisational memory includes the use of hypertext, groupware and a rhetorical method. QuestMap is a hypertext group discussion tool based on the IBIS (Issue Based Information Systems) approach, which is a well known branch of the design rationale tradition and implements the rhetorical method of Conklin. In QuestMap we have a particular issue which may have various positions justified by certain arguments. Issues, positions and arguments are saved in nodes in a network. QuestMap permits interaction between users, who simultaneously conduct discussions by placing icons, each of which represents a particular conversational component in a common window. An example of the discussion on reusability, in which some of the icons and links are used, is shown in Figure 5.

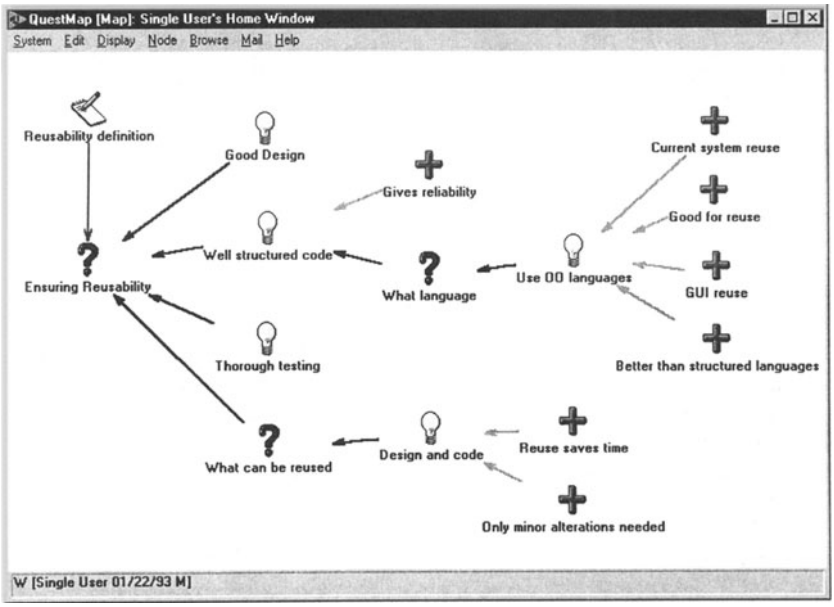

Figure 5 A discussion on reusability.

™ QuestMap is a trademark of Corporate Memory Systems, Inc

The 10 icons representing the conversational components used in QuestMap are: Question, Idea, Pro, Con, Argument, Decision, Reference, Note, List View and Map View. The links between the icons are: Supports, Objects To, Responds To, Related To, Specialises, About, Resolves and Challenges. Each icon possesses a content window which allows a name and additional information to be entered for the icon. Searches can be made for icons, e.g. all those containing a certain keyword, and hyperlinks created by copying an icon from one view to another. Although the snapshot can only show the title of each icon, the reason for its existence is made clear. QuestMap allows a textual record of a discussion to be made, and each snapshot is accompanied by such a description, shown in a separate window, which lists all of the icons, their descriptions, types and relationships to other icons in an ordered manner.

Textual records of QuestMap discussions may be recorded in the Quality Training tool database for future reference, which creates an organizational memory capability for the tool. An organisational memory requires the capture and organization of information in a manner in which it can be easily located and retrieved in an understandable form. The tool achieves this by allowing important discussions concerning quality issues to be recorded, displayed and deleted if required. Since the information is easily retrievable, time is not needlessly wasted discussing or researching a problem that has already been solved.

4 AN EXAMPLE

Our example illustrates the use of quality concepts to choose between alternative design solutions by presenting some situations arising in the design of a user interface for a beer pump system. Electronically operated beer pumps used in restaurants consist of a pressure tank (operating with carbon dioxide), a beer tank, a cooler, an electrically operated valve, a stream indicator and a tap. A button is used to open the valve, which causes the carbon dioxide to force the beer through the cooler towards the tap. There are usually two buttons for dispensing either a half pint or full pint of beer from the tank, which usually has a capacity of 30 liter (60 pints). When the tank is almost empty a warning is given that it will shortly have to be replaced. There is a possibility that surplus carbon dioxide in the pipeline could cause large amounts of foam, and a means of forcing the foam out of the system is necessary.

The software calculates the number of pints dispensed from the tank. When the level falls below a pre-defined limit a warning is given, and once 60 pints have been dispensed the system automatically locks until the tank is replaced.

Our solution considers the idea of using a proximity card to control the dispensing process. A similar system involving a proximity card and card reader is being used by a local bus company. A card gives credit for a specific number of bus journeys, and when it is passed in front of the reader the number of journeys remaining on the card is reduced by one. The reader can read a card without any contact up to a distance of 20 centimetres. The storage capacity of a card can be up to 30 bytes, which includes a serial number. The reader contains a display which can show 2 lines of text of up to 20 characters, three indicator lamps and a sound signal. When all of the journeys on a card have been used up the card can be recharged. This can be done using the card reader.

In this example we focus on the user interface. Its requirements are shown in Table 1.

Table 1 Requirements of the user interface

All activities are controlled using proximity cards and instructions are presented via a display screen. All inputs must be made via an input button. Use of the control computer's display and keyboard is not permitted.
The beer pump must be easy to use and understand since users may be under the influence of alcohol.
The reliability of the system must be high. Card units must not be deducted erroneously without beer being dispensed, nor can the system allow free pints. Each time a pint is given a deduction must be made from the proximity card.
The following types of card should be available: A Normal card which can be recharged to give credit for pints. A No Limit card which can be used indefinitely without recharging. A ServiceCard for dispensing pints, recharging other cards and controlling service operations.
The system must automatically monitor the quantity of beer which remains in the tank and prevent further use when the tank is empty. Use of the pump must be barred until the tank is replaced.

Initial tests of the user interface found that the instructions provided were insufficient and the messages too short. The designers had also overestimated people's skill in using the beer pump. For example, beer was wasted as people did not manage to put their glass under the pump before the flow of beer commenced. This caused some corrections to be made to the control software. We now describe the support given by the QuestMap tool for defining the quality goals and for choosing between two alternative versions.

4.1 Defining the quality goals

We first prioritize the most important of the 11 alternative goals (presented in the GRCM model) for the user interface of the beer pump. We may choose to place our focus on usability aspects, e.g. striving for understandable instructions, or we can emphasize reliability, e.g. ensuring that the beer flows into the glass, for example. In addition we choose the portability goal, because this kind of system could have potential clients worldwide. The QuestMap presentation in Figure 6 summarizes the reasons for our prioritization, and the textual description in Table 2 explains them in more detail.

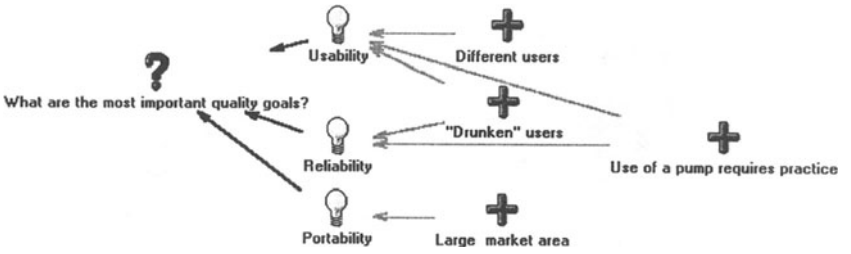

Figure 6 Reasons for the choice of quality goals.

Table 2 Textual description of the QuestMap discussion on quality goals

Quality goals

1 *Question* : **What are the most important quality goals?**
We have chosen certain quality goals and will discuss these. The definitions of the goals are based on the ISO 9126 standard and the ones chosen in this case are usability, reliability and portability.
(QuestMap Administrator)

- 1.1 *Idea* (Responds To Question 1): **Usability**
  A set of attributes that bear on the effort needed for using the software and on the individual assessment of such use by a stated or implied set of users, e.g. the value of a user interface to a set of users.
  (QuestMap Administrator)
  - 1.1.1 *Pro* (Supports Idea 1.1): **Different users**
    There may be regular users who need no guidance and casual users who need much more guidance. There are also many different ways in which to use the system. User activity may be simply dispensing a pint of beer or a more tricky job such as changing the beer tank. These activities require different instructions.
    (QuestMap Administrator)
  - 1.1.2 *Pro* (Supports Idea 1.1): **"Drunken" users**
    Because users may be under influence of the alcohol, the beer pump mechanism must be easy to use and must operate reliably and safely.
    (QuestMap Administrator)
  - 1.1.3 *Pro* (Supports Idea 1.1): **Use of a pump requires practice**
    The instructions should be as informative as possible, because there is only few lines available for them. Relevant instructions increase usability and reliability.
    (QuestMap Administrator)
- 1.2 *Idea* (Responds To Question 1): **Reliability**
  A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
  (QuestMap Administrator)
  - 1.2.1 *Pro* (Supports Idea 1.2): **"Drunken" users**
    Same as 1.1.2.
    (QuestMap Administrator)
  - 1.2.2 *Pro* (Supports Idea 1.2): **Use of a pump requires practice**
    Same as 1.1.3..
    (QuestMap Administrator)
- 1.3 *Idea* (Responds To Question 1): **Portability**
  A set of attributes that bear on the ability of software to be transferred from one environment to another.
  (QuestMap Administrator)
  - 1.3.1 *Pro* (Supports Idea 1.3): **Large market area**
    The potential market area for the system is large. It should therefore be easy to adapt to different environments, and easy to install in sports arenas, concerts halls, etc.
    (QuestMap Administrator)

4.2 Choosing between alternative design solutions

We can also use the QuestMap tool to choose between design alternatives. In view of the prioritized quality goals, we can justify our solutions in terms of usability, reliability and portability factors. The design decision situation considers improvement of the beer pump service with additional messages, and with an indicator which confirms that a glass has been placed below the tap.

We have two alternative versions for operating the beer pump, i.e. dispensing the beer. The first simply indicates that the button has been pressed, while the second is more complicated and ensures that there is a glass under the tap, to prevent waste, and that the glass is not

removed until it is full. These alternative versions are described with state-transition diagrams, but due to space limits they are not presented in this paper. Instead, we present the QuestMap supported discussion concerning the justification for alternative versions. A summarized description is given in Figure 7 and more detailed reasons in Table 3.

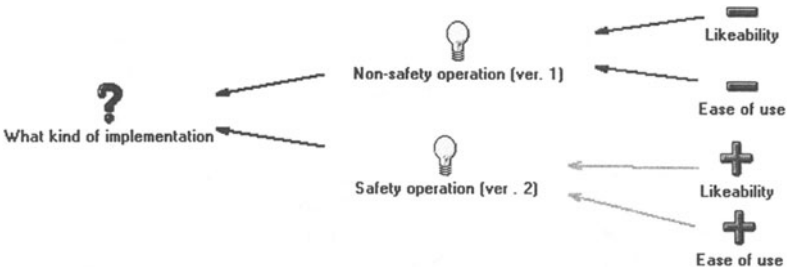

Figure 7 Choosing between alternative modes of operation.

Table 3 Textual description of the QuestMap discussion on modes of operation

Operation
1 ***Question*** : **What kind of implementation?** What mode of operation should we choose if we emphasize usability? The beer pump must be easy to use. It is possible that users may be under the influence of alcohol and hence the instructions must be easy to understand. We can explain usability in terms of likeability and ease of use. *(QuestMap Administrator)*
1.1 ***Idea*** (Responds To Question **1**): **Non-safety operation (ver. 1)** There are only a few instructions in the version 1 and no control mechanism that makes sure that there is a glass under the tap. *(QuestMap Administrator)*
1.1.1 ***Con*** (Objects To Idea **1.1**): **Likeability** Unexpected operation may cause unwillingness to use the system, e.g. if there is no glass under the tap and beer flows onto the floor, or if you do not understand how the system works. *(QuestMap Administrator)*
1.1.2 ***Con*** (Objects To Idea **1.1**): **Ease of use** Missing messages may cause confused situations, e.g. if you do not know how to get one glass of beer. *(QuestMap Administrator)*
1.2 ***Idea*** (Responds To Question **1**): **Safety operation (ver. 2)** There are additional instructions in version 2. The added control mechanism makes sure that there is a glass under the tap, otherwise the system stops. *(QuestMap Administrator)*
1.2.1 ***Pro*** (Supports Idea **1.2**): **Likeability** The added control mechanism makes sure that there is a glass under the tap and prevents unlikely events such as beer flowing onto the floor. *(QuestMap Administrator)*
1.2.2 ***Pro*** (Supports Idea **1.2**): **Ease of use** Additional instructions give relevant information to users. *(QuestMap Administrator)*

5 CONCLUSIONS

This paper addresses the problem of quality-intensive software development, which invites a software engineer to evaluate the software from the quality viewpoint. Our approach is based on the GRCM (goal-rule-checklist-metric) model and on the use of a triad of supporting tools, i.e. a graphical editor, the QuestMap tool for problem solving and the Quality Training tool for supporting the organizational memory. The Quality Training tool acts as a central tool and provides an introduction to the quality concepts. It also supports the organizational memory by recording design solutions and their justifications for further reuse. These solutions are described in the form of snapshots from the graphical editor and the QuestMap tool, which also provides discussions for further argumentation.

In our illustrative example we present some situations that arising in the design of a user interface for a beer pump application. We prioritized usability, reliability and portability as the most important of the 11 alternative goals and justified our design solutions in terms of rules derived from these factors. The example represents one of our first attempts to apply the GRCM model in practice, and our future work will include more experiments of this kind.

6 REFERENCES

Conklin, J. (1996) Designing Organizational Memory: Preserving Intellectual Assets in a Knowledge Economy, a chapter in a book (in preparation) Conklin J., *The Information Paradox: When Information Defeats Understanding*, http://www.cmsi.com/business/info/pubs/desom

Crosby, P.B (1996) *Philip Crosby's Reflections on Quality*, McGraw-Hill, New York, NY

ISO (1990) *Information technology - software product evaluation - quality characteristics and guidelines for their use,* Draft International Standard ISO/IEC DIS 9126, International Organization for Standardization, Geneve

McCall, J.A, Richards, P.K, and Walters, G.F (1977) *Factors in Software Quality*, Volumes I, II, and III, RADC reports

Tervonen, I. (1994)*Quality-driven Assessment: a pre-review method for object-oriented software development,* Dissertation thesis, University of Oulu, Department of Information Processing Science, Reserch papers, Series A19

Tervonen, I. (1996) Support for Quality-Based Design and Inspection, *IEEE Software*, **13**, 1, 44-54

7 BIOGRAPHY

Ilkka Tervonen is a professor (acting) of software engineering at the University of Oulu. He recieved PhD in software engineering from University of Oulu. His current research interests include software quality, software inspection and organizational memory.

Jouni Kokkoniemi is an assistant at the University of Oulu. He recieved MS in software engineering from University of Oulu. His current research interests include software inspection and object-oriented databases.

Gareth Smith is a student at the University of Manchester. His current research interests include software quality and object-oriented approach.