Support for Cooperative Design of End-user Tailorable Software

Jeanette Eriksson
Blekinge Institute of Technology, School of Engineering
P.O. Box 520 S-37225 Ronneby
jeanette.eriksson@bth.se

Abstract. Many contemporary business areas are dynamic and characterized by change. End-user tailorable software that allows the users to continue its evolution and adjustment is suitable in such environments. Unfortunately, the changes in the environment make it hard to know what flexibility to build into the software. The study presented here was aimed at providing an instrument that makes it possible to distinguish between different types of end-user tailoring, and to support discussions between users and developers concerning what kind of tailorability to build into the software. The study was performed in cooperation with a telecom company where tailorable software is essential to keep up with the fast changing market. The study resulted in ten attributes characterizing end-user tailorable software and a matrix capturing the values of the attributes. The matrix can be used as a guide and a basis for design decisions when implementing end-user tailorable software.

Keywords: Attributes of end-user tailoring, Design support, User participation.

1 Introduction

A fast changing world requires more and more flexibility in software, to provide support for higher reusability and to prevent the software from expiring too fast. One way to provide this kind of flexibility is via end-user tailoring. A tailorable system is modified while it is being used, as opposed to being changed during the development process. Tailoring a system is "continuing designing in use" [10, p. 223]. It is possible for a user to change a tailorable system through the support of some kind of interface.

Tailorable software is needed when the environment is characterized by fast and continuous change. As Stevens and his colleagues put it "The situatedness of the use and the dynamics of the environment make it necessary to build tailorable systems. However, at the same time these facts make it so difficult to provide the right dimensions of tailorability." [19, p. 273]. The study presented in this paper aims at providing an instrument that can support the work of finding the right dimension of tailoring when designing end-user tailorable software.

When discussing what we here call *tailorability* with people in industry, they seldom think of or talk about this kind of software in terms of tailoring; instead they simply call it flexibility. When observing work with tailorable software, or performing interviews or discussing tailorable software with people in industry, it emerged that there was confusion in the discussions between users and developers when they discussed flexibility. The reason for this is that they view flexibility from different perspectives. Flexible software is one thing when using it and a totally different thing when building the software. Accordingly, we have to look at tailoring from both the system and the user perspective, [20] as the user perspective reflects how users work with tailoring and the system perspective elucidates important issues from the developers' point of view.

There was even misunderstanding amongst the developers themselves. The reason for this was revealed to be the fact that the perspective on the software seamlessly alters between a system and user perspective. The developers in particular make this shift without thinking. The reason for this is of course that they have to take both perspectives into account in order to make good software. The fact that the differences between the two perspectives are considerable and that they are unaware of the shift in perspectives makes discussions about flexibility very complex. Under such circumstances it is hard to reach a consensus about which flexibility to implement whilst at the same time being convinced that the chosen type of flexibility is best for the situation. To make software successful it is important that there is a consensus between users and developers about how the system must work. Users and developers must have a common understanding of the phenomenon to reach a valid agreement [15]. If both developers and users understand tailoring and the differences inherent in it, then it is easier to discuss design issues and to make informed design decisions.

From an industrial perspective we end up with two issues to be dealt with:

- It is hard to know which dimensions of tailoring to implement.
- It is hard to discuss tailoring, since users and developers have different understandings of the phenomenon.

There are several conditions concerning user knowledge, technical issues and business organization that must be fulfilled to make a tailorable system work in the long run, and the tailorable software has to be supported by a collaboration between developers and users [5]. The development of tailorable software is an ongoing process where users are co-designers, [7] as it is users who evolve the software at use time. This kind of ongoing design can be called Meta Design [7]. Meta-Design is a development process where stakeholders are co-designers. Participatory Design (PD) [18] is another paradigm that includes stakeholders in the design process. PD has historically focused on involving users in the design process at the time for design, but the Participatory Design focus can be broaden to even include user involvement in design during use time [8]. Informed Participation [3, 4] is related to PD, as Informed Participation also lets people other than developers collaborate in design efforts. Informed Participation addresses open-ended design issues and tries to obtain an ownership of the problems among participants and to make the participants actively contribute to the design activities. The matrix presented in this paper is intended as support for *informed participation* in a development project. Often users' participation in development projects is mainly concerned with the user interface. We agree with [11] that the users' view of the system is not only the interface. Task related needs are what motivate end users to make changes to the system [14].

Human-centred design is necessary when designing tailorable software, since the users are co-designers. The users bring profound knowledge of the business process and organizational issues into the development project, which should be used in the design of the technical solution [9]. Gasson [9] also argues that there is a need for a dialectic process between organizational problems, implementation of changes in the business process and technical solutions to achieve a balance between human-centeredness and the design of technical solutions. The study presented in this paper aims at providing an application of Gasson's ideas in the context of tailorable software. The application, or matrix, is targeted to deal with the issues of deciding what dimension of tailoring to implement, by supporting a common understanding of end-user tailoring among users and developers.

A classification is a useful tool to aid an understanding of a phenomenon such as tailoring. A classification consisting of four categorises of tailoring is presented in [6]. The categorization is designed to take both the user and system perspectives into account, so that it can act as a basis for communication between developers and users when designing tailorable software. The categorization presented in [6] was found to be promising for use in industry. The categorization of end-user tailorable software is intended as a means of communications to involve the users more in the design process and is therefore suitable as a basis for supporting the cooperative design of end-user tailorable software.

The categorization is presented in Section 2. The formulation of the categories is at a rather abstract level and to make the categorization more precise and easier to use in practice, the categories should be assigned tangible attributes or characteristics. The idea is that the attributes of the categories can guide you to the most appropriate type of tailoring for a specific situation after you have pinpointed what type of business environment the software will be a part of, the skill and knowledge of the users and how much the developers are able to contribute to the tailoring process after the software has come in use.

In summary we have two research questions to answer to be able to deal with the industrial problems discussed above:

1. Which attributes characterize end-user tailorable software?
2. How can different dimensions of end-user tailoring be distinguished?

To answer the questions, a study was performed in cooperation with a major telecom company in Sweden. Developers and users were interviewed to elucidate which attributes are relevant to describe tailoring and how they perceive different kinds of end-user tailoring.

The rest of the paper is structured as follows. The next section will present the categorization of tailoring that acts as a base for the study. Section 3 describes the research method applied. The results of the study are presented in Section 4. The section consists of two parts, each answering one of the research questions. The first research question resulted in ten attributes characterizing end-user tailorable software and the second research question resulted in a matrix summarizing the values of each of the attributes for the four different categories of tailoring. The matrix can be used to *support the cooperative design process* when designing tailorable software. Finally, the paper ends with a discussion and conclusions.

2 Categorization of Tailoring

The categorization proposed by Eriksson et al.[6] is intended as a means of communication between developers and users in situations when deciding what kind of tailorability to implement. The categorization takes into account both a user perspective and a system perspective. The user perspective represents which changes can be made or the purpose of the activity, while the system perspective corresponds to how the change is achieved in the system (on a high level). The categorization is shown in Table 1.

Table 1. Categorization of tailorable software

	User Perspective	System Perspective
Customization	The end-user makes small changes, e.g. sets parameter values.	Parameter Values are interpreted and used in existing code.
Composition	The end-user relates different existing components to each other.	The relationships between the components are defined by a composition language. (It does not matter which programming language)
Expansion	The end-user creates a new component.	Components are integrated into the software by the implementation language and the new component does not differ from the pre-existing components. The composed component is used as a starting point for further tailoring. The software may generate code that is added to the pre-existing code, or incorporate the new component into the application in some other way.
Extension	The end-user adds code to the software.	New code (implemented by the end-user) is added to the pre-existing code. The application may also generate code to integrate the end-user's code into the software.

Customization is the simplest way of doing tailoring. It means that the user sets some values of one or more parameters and those parameters manage what functionality that is used. *Composition* means that the user has a set of components to choose from and he or she can connect them in specific ways to gain the desired functionality. *Expansion* also means that the user chooses components from a set, but the difference is that the users' combination of components is build into the system to become an integrated part. The new component is treated in the same way as the predefined components and will be accessible in the set to choose from next time the software is tailored. *Expansion* is the category which provides for the highest flexibility. It means that the user writes code that is integrated into the system either by wrapping up the new code into system generated code or, if written in a predefined way, through simply adding it to the code mass of the software. The user can either write the code in a high level language or in a visual programming language.

3 Research Method

Tailoring is especially well suited for applications used in a business environment that is characterized by fast changes, such as that in the telecom business. For example, new services continuously evolve and the supporting business systems therefore have to adapt to the altered requirements. The study was performed in cooperation with a telecom operator in Sweden. The company is dependent on flexible software that allows the user to alter the software when the need arises. Accordingly they have many tailorable systems running. The study aimed to elucidate (1) which attributes can be ascribed to tailorable software and (2) how different types of tailoring can be distinguished from each other. To achieve this, interviews were conducted where the categorization was used as a basis for the interviews.

We interviewed six developers and four users at the company. The developers were programmers, system owners and technical project leaders. The users all worked with several different systems, but their main tasks were within the same system. The users were a system coordinator, a work manager, users responsible for working with new requirements, and users helping out with further development of the system.

The interviews lasted approximately one hour to one and a half hours. Since a pilot study made it clear that it might be necessary to elaborate on some of the questions, we performed semi-structured interviews [17] which means that the respondents were asked the same questions in the same order, but follow-up questions were asked and explanations were given.

To be able to discuss the four categories on equal terms with both developers and users, the categories were translated into four representative examples. The examples were at a rather high level, free from unnecessary details, but concrete enough to allow the respondents to discuss the examples. The examples were not confined to the tasks in the telecom company. A summary of the examples in English can be found at http://www.ipd.bth.se/jer/tailoring/examples.htm.

The interviews were audio taped and transcribed in full to provide for traceability. The individual transcriptions and the analysis of the material were sent to the respondents for verification.

3.1 Design of Interviews

The researcher interviewed one respondent at a time. First the developers were interviewed, and then the users. The interviews were conducted according to a specific sequence. First the respondents read the examples of the different categories and thereafter they were asked if they could spontaneously assign attributes and qualities to the first example representing customization. Thereafter they had to answer some statements about the example and at the end they were asked if they could find any resemblances between the given example, and systems they work with

or know about at the company. The procedure was the same for all four examples representing customization, composition, expansion and extension respectively.
After reading the examples and spontaneously expressing their views on the characteristics of the categories, the respondents had to take a standpoint on eleven proposed attributes that originate from the cooperation with the telecom company. The attributes have emerged through participant observations, discussions and interviews.
The interviews made it clear that changes may be required because of changes in the business environment, because of a need for improved usability or because of internal issues in the system itself. The attributes can be divided into corresponding groups. One group concerned the category's suitability for different types of *business changes.* Another group of attributes related to *usability* and a third group involved *software attributes*. The attributes are listed below.

Business changes

Attribute 1:	Frequency of change – how often the business changes occur, frequently or infrequently.
Attribute 2:	Anticipation of change – to what extent it is possible to anticipate the business changes.
Attribute 3:	Durability of change – how long the business changes last.
Attribute 4:	System support for change – how well the software supports business changes
Attribute 5:	Consequences if handled wrongly – how extensive the consequences would be for the company if the changes are handled wrongly.

Usability issues

Attribute 6:	Simplicity – how easy it is to realize the changes in the software
Attribute 7:	User control – how much control the users have of what happens in the software
Attribute 8:	Accountability – how easy it is for the users to know if the result is correct.
Attribute 9:	Realization speed – how fast it is to realize the changes in the software.

Software attributes

Attribute 10:	Fault tolerance– to which degree the software prevents mistakes.
Attribute 11:	Complexity– how complex the software is

3.2 Analysis

The analysis has been performed in a systematic way, according to a specific, pre-defined schema. The material from the interviews consists of spontaneously stated attributes, predefined attributes, comments, and feedback from respondents. The four components have been considered in the analysis.
The analysis of the interviews consists of two parts that respectively correspond to the two research questions.

Analysis 1:	Analysis to determine what attributes characterizes end-user tailorable software.
Analysis 2:	Analysis to determine how the respondents perceive the different types of tailoring and decide a value for each attribute, to be able to distinguish different dimensions of tailoring.

Analysis 1. The first step in Analysis 1 is to compare each attribute to see if they are perceived in the same way for all four categories. If they are the same for all the categories then they do not add any information that could be used to distinguish between the categories. All attributes are compared and if they are not the same for all categories they are added to the pile of remaining attributes. If the attribute is the same for all four categories the respondents' comments are consulted to determine if the attributes really were perceived as the same. Perhaps the respondents had made a statement based on different interpretations of the proposed attributes. If the attributes are found to be the same they are removed, otherwise they are added to the pile of remaining attributes. To facilitate determination of whether the attributes were perceived as the same, all statements were assigned a value. A statement interpreted as positive towards an attribute generated a score of 300 and a negative statement was assigned 100 points. Accordingly a neutral statement generated 200 points. Initially, to see if the attributes were the same for all categories, the value of the attribute was summarized. For example if all the users think that Example 1 has high fault tolerance the sum is 1200 points (4 users x 300 points) and if all the users think that Example 4 has low fault tolerance it generated a total of 400 points (4 users x 100 points). The sums for each category are compared and if they are the same they have to be examined further and each comment must be checked more closely.
The second step in Analysis 1 is an examination of how a respondent's answers relate to the other answers in the group. The coefficient of variance has also been used as a measure of the disagreements between respondents [16]. If the respondents' view of the attributes of the examples varied a lot, then the attributes should be removed, as this does not reveal anything about the category. The remaining attributes from step A were examined. If there is a deviation in opinions within the group the respondents' comments were checked. Based on the comments, the relevance of the attributes was questioned. If the attribute was found relevant it was added to the pile of remaining statements otherwise it was removed.
In step three of Analysis 1, the respondents' spontaneously assigned attributes were listed and compared with the pre-defined attributes. If they were the same the attributes were added to the comments, otherwise they were considered as attributes of the intended category.

Analysis 2. The remaining attributes from Analysis 1 were analysed to explore how the user group relates to the developers group, per attribute. The median value for each attribute was used for guidance. If the users and developers agree upon the attributes the attributes were collected into one pile, but if opinions differ, the respondents' comments are considered and the user specific-and developer-specific statements are accumulated into separate piles.

4 Result

When examining the totals in the first step of Analysis 1 there were some attributes that had the same total, but when the individual scores and the comments were inspected it was revealed that they actually differed. The result of the analysis is that none of the attributes was perceived as the same for all four categories and therefore none of the attributes could be excluded at this stage.
The second step in Analysis 1 resulted in the removal of three attributes (3, 5 and 6), as there were strong disagreement among the respondents. The attributes concerned durability of changes, consequences if handled wrongly and simplicity. The respondents regarded durability of change and simplicity as somewhat unimportant, and their answers were therefore fairly random. The consequences of the change being handled wrongly were too difficult to state as it is highly dependent on the situation.
The users found it difficult to spontaneously come up with attributes describing the four examples. They experienced difficulties in moving from the concrete example to a more abstract level. They found it to be easier to associate the example with a system they work with. It was much easier for developers to come up with attributes for the four examples and the developers came up with a couple of attributes each.
When comparing the developers’ attributes with the pre-defined attitudes it was found that most of the attributes were the same. The attributes that differed from the pre-defined ones related to usability issues and were mentioned by several of the developers. The attributes were of two kinds and concerned:
Frequency of use: how often the end users use the software and thereby the degree of familiarity the users have with the software, and
User competence: how skilled the users of the software are.
Analysis resulted thereby in ten relevant attributes that can be used to describe end-user tailorable software (see Table 2).

The results of Analysis 2 showed that users and developers had the same perception of Example 1 (customization).
For Example 2 (composition) the users and developers had slightly a different perception of user control, accountability, fault tolerance and complexity. When it comes to user control and accountability the users judge the accountability and control to be medium high, while the developers think it is somewhat higher. In other words, the developers thought that Example 2 contains slightly more accountability and user control that the users did. For fault tolerance and complexity there were also some small differences. The users considered the fault tolerance and complexity to be medium high for Example 2, whilst the developers thought that fault tolerance is somewhere between medium high and low and the complexity between medium high and low. (See Table 2)
Also for Example 3 (expansion) there were some differences in views. One thing is that the developers had a unanimous view that Example 3 is well suited when there is a need for high support for changes, but the users are not that sure. They believe that such software provides quite a lot of flexibility, but they are not certain that Example 3 really supports change so well that it should be regarded as giving “high support of change”. A small variation also exists in the judgment of the amount of user control

and accountability provided by Example 3. The developers consider Example 3 to provide medium high user control and accountability while the users believe it to be somewhere between medium high and high. The differences of opinion in this case were however very small. A more significant difference was found when it came to anticipation of change. Here the users and developers had diametrically opposite opinions. The users thought that Example 3 was suitable for situations characterized by a high degree of anticipated changes. The developers thought to a higher degree that Example 3 was also well suited for unanticipated changes. (See Table 2)
The issue of user control and accountability for Example 4 (extension) resulted in some discussions of which knowledge is build into the system and what should be controlled by the user. Both users and developers agreed that it is possible to view Example 4 as supporting either high control and accountability or low control and accountability. There is very little user control and accountability built into Example 4, but on the other hand the user handling the software should be skilled and know what he or she is doing. Thereby you could say that the software gives control to the users. User control and accountability should therefore be regarded as high. The uncertainty is represented by question marks in Table 2.

Table 2. Matrix of the attribute values of the four categories of end-user tailoring. (L=Low, M=Medium, H=High, ?= Uncertainty of how to use the attribute)

Characteristics		Customization	Composition	Expansion	Extension
Business Changes	Frequency of change	M	M	H	H
	Anticipation of change	H	M	L-H[1]	L
	System support of change	L	M	M-H	H
Usability Issues	User control	H	M-H	M-H	?
	Accountability	H	M-H	M-H	?
	Realization speed	H	H	M	M
	Frequency of use	L	H	-[2]	-
	User competence	-[3]	-	M-H	H
Software Attributes	Fault tolerance	H	M-H	M	L
	Complexity	L	L- M	M	H

Note that there are two pairs of attributes that show a dependency (Table 2). User control and accountability have corresponding values for all categories. When user

[1] Users thought the example was highly suitable for anticipated changes, developers thought the example was not so suitable for such situations.
[2] The spontaneously given attributes were not stated for Example 3 and 4.
[3] The spontaneously given attributes were not stated for Example 1 and 2.

control is perceived as high, accountability also has a high value. Fault tolerance and complexity also seem to be related. If fault tolerance is high then complexity is low and vice versa.

When it came to the spontaneously stated attributes, example 1 was considered suitable when there are many end users that use the software only occasionally and Example 2 was regarded as fitting when there are few end users who use the software frequently. Examples 3 and 4 were believed to be appropriate when the end users are skilled and used to computer work, but Example 4 was judged to be appropriate only for a few users that are extremely skilled super users.

The matrix can be used when the tailoring capabilities are insufficient and a new feature is needed, and a development team is put together with both users and developers. In such situations, the matrix can act as a gateway to the categorisation of end-user tailoring and point to a type of tailoring that may be appropriate for the specific feature. The matrix is intended as a basis for discussion between users and developers and the matrix has to be accompanied by complementary tools that relate the different categories of tailoring to implementation techniques to be able to make a decision of how to implement the feature.

The matrix should be seen as a guiding tool. The matrix should not be seen as providing the absolute truth. When designing a tailorable system the matrix could be used as a basis for discussions of the needs and requirements of the specific situation. What the matrix can do is to help the participants focus on a subset of tailoring possibilities and make it easier to choose the right type of tailoring for a specific situation. What can be expected from different types of tailorable software is listed in the matrix, but it is the participants in the project that have to make the tradeoffs between the attributes.

6 Discussion

The matrix is intended for a design environment where the users are informed participants, and where users and developers claim a common ownership of the software product developed. The purpose of the matrix is to act as a basis for design discussions where the users and developers discuss the requirements of the tailorable software to understand better the domain and design problems. The matrix can help the design team to pinpoint issues to discuss and to reach a consensus to enable decisions concerning the dimensions of tailoring needed in the given context. By consulting the matrix and comparing the values of the attributes with what is needed in a specific context, it is possible to get an indication of the kind of tailoring to implement and to be able to make informed design decisions.

There is a similarity between assigning quality attributes to software and assigning attributes to tailoring categories. Both aim to describe a phenomenon by assigning characteristics to it. There are several software quality models, for example [1, 12, 13], and their common effort is to manage quality issues in software development. There is a resemblance between these quality models and the software attributes extracted from our study. Some of the attributes in the matrix can also be found in some quality models. The intention of the matrix is not however to give a general

overview of different quality attributes. The matrix aims to distinguish between different types of tailoring and to support design decisions when designing tailorable software. However, there are some similarities. McCall's model, for example, is an effort to bridge the gap between the users' view and the developers' view [13]. The matrix also aims to bridge the gap between users and developers by providing a means of communication, and although we do not claim it to be as complete as McCall's model, the study gives us a good indication of which characteristics can be assigned to the different types of tailoring.
Bosch [2] advocates assessment of the quality attributes during architectural design. The attributes are used for evaluating the architecture to determine if the architecture has to be transformed or not. The attributes in the matrix are not used for evaluation. The intended use of the matrix could be said to be a bottom up approach in comparison with Bosch's method. The four categories could be seen as a kind of "design pattern light" for tailorable software. Instead of imposing a design pattern after the architecture has failed to provide for the required quality attributes, the matrix starts out from the categories that have assigned attributes and trade-offs are made. The architecture is then built based on the selected category. Another difference between Bosch's approach and ours is that Bosch presumes that it is possible to assign an exact, measurable value to the quality attribute, but we only assume that the participants can grade the attributes from low to high.

7 Conclusion

The study made visible ten attributes of end-user tailoring. In interviews with users and developers at a telecom company the respondents were asked to give their opinions of what characterizes four categories of end-user tailoring. Their perceptions of the categories were analysed and it was possible to process their views into a matrix representing four types of tailoring in the form of attribute values. The attributes represent organizational, business and technical issues to consider and can be used in a dialectic process to balance the human-centeredness and the technical solution, as Gasson requires [9].
The matrix can be used as guidance and a basis for design decisions when implementing end-user tailorable software. The attributes are at a level that can be understood by both users and developers and as shown, even though differences exist, the opinions of users and developers are quite similar. The matrix makes it possible to distinguish between different dimensions or types of tailoring, by providing values for the attributes that characterize end-user tailorable software.
The categories and attributes of the categories, together with the matrix and examples, facilitate the understanding of different types of tailoring and should make it easier for developers and users to discuss tailorability and the requirements associated with tailorable systems.

Acknowledgment. This work was partly funded by The Knowledge Foundation in Sweden under a research grant for the project "Blekinge - Engineering Software Qualities (BESQ)" (http://www.bth.se/besq).

References

1 Boehm, B. W., Brown, J. R., Lipov, M.: Quantitative Evaluation of Software Qualities, North Holland, Proceedings of the 2nd International Conference on Software Engineering, ICSE 1976, California, USA, (1976).
2. Bosch, J.: Design and use of Software Architectures: Adopting and evolving a product line approach. Pearson Education, Addison-Wesley and ACM Press, Reading, MA, (2000).
3. Brown, J. S. and Duguid, P.: The Social Life of Information. Harward Business School Press, Boston, MA, (2000).
4. Brown, J. S., Duguid, P., and Haviland, S.: Toward Informed Participation: Six Scenarios in Search of Democracy in the Information Age, The Aspen Institute Quarterly, 6, 4 (1994), 49-73.
5. Eriksson, J. and Dittrich, Y.: Combining Tailoring and Evolutionary Software Development for Rapidly Changing Business Systems, Journal of Organizational and End User Computing (JOEUC), 19, 2 (2007).
6. Eriksson, J., Lindeberg, O., and Dittrich, Y.: Four Categories of Tailoring as a Means of Communication, submitted to the Journal of Software and Systems (2007).
7. Fischer, G.: Meta-Design: Beyond User-Centered and Participatory Design. Proceedings of HCI International 2003, Crete, Greece, June 2003. Lawrence Erlbaum Associates, Mahwah, NJ, (2003), 88-92.
8. Fisher, G. and Ostwald, J.: Seeding, Evolutionary Growth, and Reseeding: enriching Participatory Design with Informed Participation. Proceedings of the Participatory Design Conference (PDC'02), Malmö University, Sweden, (2002). 135-143.
9. Gasson, S.: Human-centered vs. user-centered approaches to information system design, JITTA: Journal of Information Technology Theory and Application, 5, 2 (2003), 29-46.
10. Henderson, A. and Kyng, M.: There's No Place Like Home: Continuing Design in Use, Design at Work, J. GreenBaum and M. Kyng, (eds.), first ed. Hillsdale, NJ: Lawrence Erlbaum, (1991), pp. 219-240.
11. Ilvari, J. and Iivari, N.: Varieties of User-Centeredness. Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS '06, Hawaii, IEEE, (2006).
12. ISO: ISO/IEC 9126 Information Technology - Software Quality, International Standard Organization.
13. McCall, J. A., Richards, P. K., and Walters, G. F.: Factors in Software Quality, Nat'l Tech Information Service, 1, 2 and 3. (1977).
14. Nardi, B. A.: A Small Matter of Programming - Perspectives on End User Computing. MIT Press, Cambridge, 1993.
15. Preece, J., Sharp, H., and Rogers, Y.: Interaction Design - beyond human-computer interaction. John Wiley & Sons, Inc., New York, (2002).
16. Regnell, B., Höst, M., Natt och Dag, J., Beremark, P., and Hjelm, T.: Visualization of Agreement and Satisfaction in Distributed Prioritization of Market Requirements, in 6th International Workshop on Requirements Engineering: Foundation for Software Quality. Stockholm, Sweden, (2000).
17. Robson, C.: Real World Research, Second ed. Blackwell Publishers Ltd, Oxford, (2002).
18. Schuler, D. and Namioka, A.: Participatory Design: Principles and Practices. Lawrence Erlbaum Associates, Hillsdale, NJ, (1993).
19. Stevens, G., Quaisser, G., and Klann, M.: Breaking It Up: An Industrial Case Study of Component-Based Tailorable Software Design, in End-User Development, vol. 9, H. Lieberman, F. Paternò, and V. Wulf, (eds.) Dordrecht, Netherlands: Springer, (2006), 492.
20. Stiemerling, O.: Component-Based Tailorability, Dissertation. Bonn, Germany: Bonn University, (2000).