Position paper at the
Fifth International Workshop on Requirements Engineering: Foundation for Software Quality
(REFSQ'99), Heidelberg, Juni 1999

# Towards a Method for Evolutionary Implementation of Groupware

Klaas Sikkel[1], Huub Ruël[2], and Roel Wieringa[1]

[1] Faculty of Computer Science, University of Twente,
P.O. Box 217, 7500 AE Enschede, the Netherlands
{sikkel,roelw}@cs.utwente.nl
[2] Faculty of Technology and Management, University of Twente,
P.O. Box 217, 7500 AE Enschede, the Netherlands
h.j.m.ruel@sms.utwente.nl

**Abstract.** Groupware is a typical example of an application domain in which requirements are hard to elicit and keep changing before, during, and after the introduction of the system. This calls for an evolutionary implementation approach. Several socio-technical models give an explanation of the interaction between the technical and the social system, i.e., the software and its organisational environment, but these models have not yet led to a clear method for evolutionary implementation. In order to arrive at such a method, we need a theoretical framework for how this adaptation process takes place. Adaptive Structuration Theory (AST) is a good candidate for such a framework. The BITE research project at the University of Twente aims to operationalise the concepts of AST through application in several industrial pilot projects. Based on these pilots we will develop a method for evolutionary implementation of groupware.

## 1. Groupware requirements are hard to capture

Groupware is a generic name for computer systems and applications that support collaborative work. There is a large variety of types of systems that can be categorized as groupware, including video conferencing systems, collaborative authoring systems, and group decision support systems, to name a few that have little in common. In contrast with workflow, which aims to make work more efficient by implementing work processes, groupware does not prescribe how the work is to be done.

The general problems with requirements, such as elicitation of requirements that are hard to verbalize, requirements that shift during a project, etc., are well known. For groupware, however, it is even more difficult to capture the requirements, for two reasons.

Firstly, one has to note the difference between work processes and work practices. Work processes are described in some form, but office work hardly ever consists of slavishly following a prescribed procedure. Office procedures are normative accounts of ideal cases, templates used for rational reconstructions of work actually done, to be used for accountability purposes. Work practices include informal, locally determined

practices to get the work done. These practices are not documented and hard to elicit. Moreover, work is 'situated' [16]: reality is inexhaustible and one has to deal with contingencies of all kinds. Groupware systems aim to support work practices. Therefore, testing the usability of these systems can only be done *in situ*, using it for the real work. Hence the requirements cannot be validated until the system has been built and installed and used.

Secondly, the introduction of a successful groupware system affects the work practices. When a groupware system is successfully employed, the frame of reference of the users is expanded [13]. They will think of new ways of using the system and become aware of missing features that were not relevant in the situation before deployment of the new system. Hence the requirements continue to shift after the system has been installed.

## 2. It does not suffice to capture requirements in advance

Techniques and methods to handle system design in those areas, in which requirements analysis is hard, include participatory design [6] and ethnographical methods [9]. However, no matter how well the requirements analysis is done and the users' interests are represented in the system design, this does not guarantee a system that is right the first time. As motivated above, evolutionary implementation, in which system (re)design is interlaced with deployment of the system for practical work, is needed, so as to adapt the system to the work practices that evolve with system deployment. This is known as the socio-technical design circle [18, 12].

## 3. Evolutionary implementation

In order to organize an evolutionary implementation process, there are three kinds of concerns that have to be dealt with: (1) the organisational environment, (2) software design issues, and (3) the interplay between these.

### 3.1. Organisational issues

We have noted that the organisational environment is influenced by introduction of a system. If we want to obtain – and maintain – a better match between the software design and the organisational environment, we need to understand how this adaptation process takes place. We believe that Adaptive Structuration Theory (AST) [14, 3] can be helpful to obtain such an understanding. It provides a theoretical framework of how the adaptation takes place, and gives insight in the variables to be observed in order to monitor the adaptation process. Moreover, it emphasizes that adaptation is subject to group dynamics: different groups adapt to the same technology in different ways.

**Adaptive Structuration Theory.** AST is based on Anthony Giddens' structuration theory [4, 5]. This theory is an attempt to synthesize the extreme positions in the agency-structure debate in the social sciences: the deterministic and the voluntaristic point of view towards human action. Giddens takes the point of view that human action is constrained by institutions or structures, but that these structures, in turn, are influenced and altered by human action through the process of ongoing use.

DeSanctis and Poole adapted Giddens' theory to study the interaction of groups with information technology, and called it A*daptive Structuration Theory.* AST criticizes the technocentric view of technology use and emphasizes the social aspects. Groups "mediate technological effects, adapting systems to their needs, resisting them, or not using them at all" [14, p.177]. Groups using information technology for their work dynamically create perceptions about the role and utility of the technology, and how it can be applied to their activities. These perceptions can vary widely across groups. These perceptions influence the way how technology is used and hence mediate its impact on group outcomes [14, 3].

The process of applying the structures by which human action is 'guided' and created is called *structuration.* This is one of the two central concepts to AST. Structures are provided by the tasks of the group, organizational culture, group norms, the knowledge represented by participants and the technology used. The structures available to group members from these factors are the *structural potential* and the specific structures used are *structures in use* [14].

The second central concept in AST is *appropriation.* Users of technology makes choices from the structural potential of the technology. Users select specific structures and adapt them to their needs, while they reject others. It is this selection and adaptation of structures that DeSanctis and Poole call *appropriation.* Appropriation can differ between groups, even when the same structures are available, and can be characterized by at least three aspects: faithfulness of appropriation (which is the extent to which structures provided to a group are used in a manner consistent with the *spirit* of the technology), consensus (which is the extent of agreement among group members on how a certain technology should be used), and attitude (which are the views about using the technology) [14, 2].

In sum, AST gives us a theoretical model of how groups of users adapt to changes in technology, and provides the variables to be observed in order to monitor this process. This makes its more tangible than general ethnological and socio-technical approaches.

**Applying Adaptive Structuration Theory.** An empirical study on the implementation of information and communication technology as a continuous process was conducted by Korunka et al. [10]. They used a model which distinguished three levels in the continuous implementation process: (1) characteristics of the technology implementation, (2) structure and style of the technology implementation, (3) effects of the implementation. Taking this model as a basis, and filling it up with variables from AST, we obtain a research model to monitor the adaptation process in a user group. This research model, shown in Figure 1, is the starting point for our empirical research. To do empirical research, the model has to be operationalised further. A key question is how to translate theoretical measures for appropriation [3] into practical

metrics. In the BITE project, we will investigate these issues by applying the research model to industrial pilot projects.

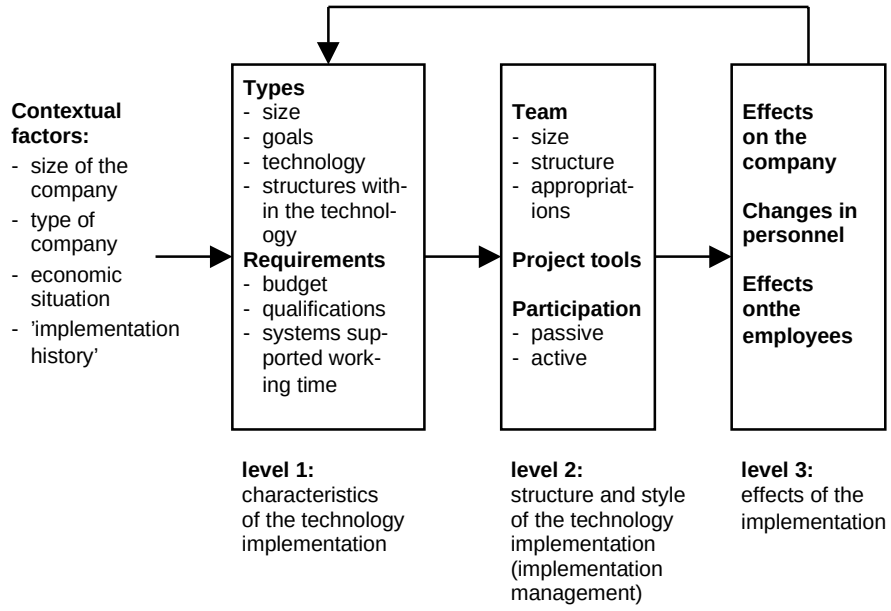


**Fig. 1.** Research model of organisational change

### 3.2. Software design issues

We now turn to the second issue in evolutionary implementation, namely software design. The use of a proper architecture is a key factor to make a system decomposable in modules that can be exchanged and adapted. There is, however, no such thing as *the* architecture of a system. We distinguish between the essential architecture, giving a high level system decomposition in terms of the functional design, and an implementation architecture, which can be quite different [8]. See also Kruchten [11], who distinguishes additional architectural views. Maintaining an essential architecture and tracing the relation between the essential and the implementation architecture improves the flexibility to adapt a system to changing requirements.

In the BITE project, we will build upon our work on architectural design [7, 8] to define architectures that are robust under changes in requirements. These changes are monitored by the variables in our research model for organisational change discussed above.

### 3.3. Matching the software design with the organisational environment

We have an understanding of the issues that play a role in the interaction between the social and the technical system, but as yet we have no way of controlling these issues. There are various socio-technical models, but none of these has given rise to a clear method [17]. How to actively match the software design with the organisational environment in an evolutionary process is a challenge for the next few years.

In the BITE research project we will take up this challenge and investigate the feasibility of a methodological approach. We believe that AST provides us with a theoretical foundation and with the key variables needed to develop a method for maintaining a match between organisational environment and software design. Such a method has to contain the following elements:

1. *A method for modelling groupware system requirements.* The solution should not be sought in improved formal modelling, as work practices, by virtue of their very nature elude formal modelling. A scenario-based approach [1] is likely to be more effective. Scenarios do not guarantee a complete set of requirements, but this is not a problem in an evolutionary approach, addressing a domain in which requirements are inherently incomplete and unstable.
2. *A method for tracing requirements in an evolving implementation.* Tools that support this are available, e.g. DOORS [15]. The issue of how to partition requirements into traceable chunks still remains, however.
3. *A method for monitoring and describing changes in requirements*. This is the innovative part of our project. As elaborated in Section 4.1, we conjecture that AST provides us with a conceptual framework that helps us develop such a method.

The state of the art in Requirements Engineering allows us to determine the fact *that* requirements have changed. We have some understanding of *why* they change. If we can get a grip on *how* requirements change, this may help to improve the process matching the software design with the organisational environment.

## 4. Discussion

We conjecture that Adaptive Structuration Theory can be helpful in determining how the organisational environment is adapted to the introduction of a system and, consequently, how the requirements to the system evolve. Such an understanding is a necessary in order to develop a methodological approach to make the software match a changing environment.

The research project described here is a cooperation between the faculties of Computer Science and Technology & Management of the University of Twente. We hope to report results in a few years time.

## References


1. J.M. Caroll (Ed.) Scenario-based Design: Envisioning Work and Technology in System Development. Wiley, NY, 1995.
2. W. Chin, A. Gopal, W. Salisbury. Advancing the theory of Adaptive Structuration: the development of a scale to measure faithfulness of Appropriation. *Information Systems Research* **8** (1997) 342-367.
3. G. DeSanctis, M.S. Poole. Capturing the complexity in advanced technology use: Adaptive Structuration Theory, *Organization Science* **5** (1994) 121-147.
4. A. Giddens. *The constitution of society: outline of the theory of structuration*. University of California Press, Berkeley, CA, 1984.
5. A. Giddens. *New rules of sociological method: a positive critique of interpretive sociologies*, 2nd ed., Polity Press, Cambridge, UK, 1993.
6. J. Greenbaum and M. Kyng (Eds). *Design at Work: Cooperative Design of Computer Systems.* Lawrence Erlbaum Associates, Hillsdale, N.J., 1991.
7. P. Grefen, R. Wieringa. Subsystem Design Guidelines for Extensible General-Purpose Software. *3rd International Software Architecture Workshop* (*ISAW3*); Orlando, Florida, 1998, 49–52.
8. P. Grefen, K. Sikkel, R. Wieringa. Two Case Studies of Subsystem Design for Extensible General-Purpose Software. Report 98-14, Center for Telematics and Information Technology, Enschede, Twente.
9. J.A. Hughes, D. Randall, D. Shapiro. Faltering from Ethnography to Design. *Proc. ACM Conf. on Computer-Supported Cooperative Work*, 1992, 115–122.
10. C. Korunka, A. Weiss, & S. Zauchner. An interview study of 'continuous' implementations of information technology, *Behaviour & information technology* **16** (1997) 3–16.
11. P.B. Kruchten. The 4+1 View Model of Architecture. *IEEE Software*, Nov. 1995, 42–50.
12. V.L. O'Day, D.G. Bobrow, M. Shirley. The Social-Technical Design Circle. *ACM Conf. on Computer Supported Cooperative Work (CSCW'96)*, Cambridge, Mass., 1996, 160–169.
13. W.J. Orlikowski. Improvising Organizational Transformation Over Time: A Situated Change Perspective. *Information Systems Research* **7** (1996) 63–92.
14. M.S. Poole, G. DeSanctis. Understanding the Use of Group Decision Support Systems: the Theory of Adaptive Structuration. In: J. Fulk, C. Steinfield. (Dds.) *Organizations and Communication technology*, Newbury Park, CA, 173–193.
15. Dynamic Object Oriented Requirements System (DOORS) Reference Manual, Version 2.1 Quality Systems and Software ltd., Oxford, UK.
16. L.A. Suchman. *Plans and Situated Actions*. Cambridge University Press, Cambridge, UK, 1987.
17. A. Sutcliffe and S. Minocha. Linking Business Modelling to Socio-Technical System Design. CREWS-Report 98-43, Centre for HCI Design, City University, London.
18. L. Tornatzky and M. Fleischer. *The process of Technological Innovation*. Lexington Books, Lexington, Mass., 1992.