Application-Oriented Software Development for Supporting Cooperative Work

Heinz Züllighoven, Guido Gryczan, Anita Krabbel, Ingrid Wetzel
Software Engineering Group, Computer Science Dept., University of Hamburg
[zuelighoven, gryczan, krabbel, wetzel]@informatik.uni-hamburg.de

1 What is Application-Orientation?

Customer relations have become one of the central concerns of companies both in the manufacturing and the service sector. Software engineering thus should rethink its aims and methodologies. IT usage and software development have to be seen as means to an end which means providing professional users with useful and usable software so that the can offer adequate services to their customers. This way of looking at IT and software development is what we call application-orientation.

Application-orientation means that analysis, design and construction of software is firmly based on the tasks and the way of dealing with them in everyday work situations. Understanding the tasks at hand and the concepts behind them is the main challenge for software developers. Application software, therefore, should reflect and represent the core concepts and the familiar objects and means of work of the application area.

2 The Tools & Materials Approach

Building application-oriented software is a task that has to combine usage quality with work organisation. The ultimate goal of our so-called Tools & Materials approach, is to adequately support users in their tasks. But THE user as such is a fiction. Also, tasks are of very different nature. Complexity increases as users have to fulfil their tasks in different organisational settings.

This means that software developers have to cope with a design task in its original sense, creating artifacts with a suitable functionality and an adequate way of handling. Our guideline for this design task is what we call a leitmotif.

A *leitmotif* makes the way of looking at software explicit. It helps developers and users to understand and design a software system on a general level.

A leitmotif can be made tangible with the help of a set of design metaphors which solidify the general guidelines in a pictorial way.

A *design metaphor* describes a component of a software system by means of an artifact from the users’ 'every-day' world. With the help of these metaphors it is possible for software developers and users to relate design and implementation components to familiar implements and terms, so that all parties share a common background and have a basis for communication.

We have chosen Tools and Materials as our predominant design metaphors, since we can frequently describe a work situation by saying how tools and materials are used to achieve a goal. Materials are those objects of the application domain that are worked upon and then become part of the work results. Tools reflect the organisation of work. They objectify our experience of working with materials in a similar way in different situations. Two additional design metaphors are automaton and desktop. An automaton realizes a task completely so that user-interaction is reduced to input data. The user has little means to change the implemented flow of control. The desktop captures the notion of space. Design metaphors mentioned are almost always used in the context of a desktop. Design metaphors have an application-oriented interpretation ("I use the cashier-tool to deposit an amount of money in my account") and a technical interpretation ("The material class *account* must have a method *deposit* that takes an *amount of money* as parameter"). They play an important role when partitioning frameworks (Bäumer et al. 1997).

3 Cooperative Work and Workplace Types

Originally, we have build support for the individual workplace mainly in the domains of software engineering environments and in the banking sector. But we soon had to face the task of integrating components that could support cooperative work. Over the last two years we have extended our approach into this area. The original approach can be characterised by task-orientation, involvement of users, and a reification of relevant issues as components of the software system. With the shift of focus towards cooperative work, the new issues are: (a) intertwining of functionality and cooperation within tasks (Grudin 1994) (b) a distinction between explicit and implicit ways of cooperation and their support and, (c) a reification of means and media of cooperation and of the cooperation process itself.

We do not start by asking, how people cooperate but we focus on the tasks and the characteristics of the relevant work situation. We try to understand what type of work we are faced with, who does what with whom and, most importantly, why.

A basis for supporting various types of cooperation is to identify different workplace types. Taking, for example, normal office-type of work, we have identified (1) the well-equipped workplace for expert users suited for flexible, situated tasks, the (2) function workplace for expert users with repetitious tasks, and the (3) back-office workplace for a small set of routine tasks for low qualified users.

Each type of workplace is equipped with the adequate items (i.e. tools, materials, automata). This categorisation of types of workplaces is by no means exhaustive. We emphasize the underlying principle: Workplace types are related to the work organisation in an application domain.

4 Cooperation Types

Based on workplace types and the cooperative work within an application domain, we have set up an explicit model of cooperation which consists of actors, objects and means of work and explicit means and media of cooperation. The central idea is to extend the different workplace types by components which can support cooperation but which fit within the overall leitmotif and the metaphors for applications in this domain. So, the design issue is fitting different workplace types to the cooperation model of the application domain.

Evaluating different projects, we have identified similarities among various ways of cooperation and introduce cooperation types. A cooperation type abstracts from concrete ways of cooperation. It characterizes work division, used cooperation and coordination mechanisms and tasks. For cooperation types we distinguish *cooperation media* from *cooperation means*. Cooperation means are objects used for structuring and coordinating the cooperation while cooperation media are objects supporting the exchange of information or material. Our interpretation of coordination is based on (Malone and Crowston 1994).

In the following we sketch three examples of cooperation types: implicit cooperation through a common archive, explicit cooperation by PO boxes and explicit coordination with dockets.

4.1 Cooperation with Archives

- *Situation/Example*: A typical situation for using archives is a small group of specialized advisors in a bank (e.g. for company credits) who are in charge of a group of customer companies. They share the same customer records.
- *Cooperation Model*: An advisor team provides continuous services to a selected group of customers. The team member share folders and files holding the customer records. A common archive storing these folders can be accessed from each work place. Usually, a team member will retrieve only part of the customer records (e.g. a credit folder) for an individual task. While working with the archive each workplace receives awareness information of who works on the same material in parallel or who has worked on it in the past. There are conventions on how to solve conflicting accesses. These conventions might be enforced by the system or they are organizationally respected by the users supported by appropriate awareness information (Mark et al. 1997).
- *Cooperation Means*: Common archive with folders and files. Awareness information.
- *Cooperation Media*: Browser for accessing the archive.

4.2 Cooperation with PO Boxes

- *Situation/Example*: PO Boxes are used in situations where, e.g., credit officers or customer advisors send credit application forms to their back-office. These forms have to be distributed among the people working in that back-office in order to mail routine letters and perform standard checks.

- *Cooperation Model*: The management of a unit has to distribute simple tasks among a group of persons. In principle, every task can be handled by each person of the team and it can be completed by a single person. So we need an easy way of forwarding the tasks together with dossiers and documents. In addition, it must be easy to find a substitution, if a person is not on duty. PO boxes for each team member are connected to a general group mailbox. The group mailbox is linked to an email service. Other groups and units send their documents and forms to this general mailbox. One person is in charge for distributing the incoming mail to the different PO boxes. The PO boxes with their name tags indicate which work places are available and additionally (by showing their content) who is actually there or whose work assignment is unbalanced. There is no implemented access mechanism. Each person involved knows by convention who has the right to put something into a box and who is allowed to empty a box.
- *Cooperation Means*: General group mailbox, individual PO boxes, distributed documents like dossiers and forms.
- *Cooperation Media*: Email system, browser tool for accessing mail boxes.

4.3 Cooperation with Dockets

- *Situation/Example*: Dockets are used when a complex task like granting credits is accomplished in several steps. These sub-tasks are handled according to well-known rules and with a clear work division e.g. between back office, department head, controller and account manager.
- *Cooperation Model*: A small fixed set of people works on complex tasks exchanging information and documents. The task is subdivided into well-known steps and is being performed by known rules including given sequences and responsibilities. Every actual work step has to be documented. But the concrete work situation frequently calls for flexibility in the selection of individual work steps and the overall working sequence. Dockets (i.e. to-do lists) can be fixed to folders or dossiers which are passed between the work places involved. These dockets are used as process pattern signalizing the usual work sequence and the persons or workplaces involved. Additionally, they indicate the status of the process. Each receiver of a document or folder with an attached docket has control over the next steps by possibly changing the process pattern. The docket works together with a mail system which delivers the attached material according to the status of the docket. While a complex task is processed each person involved can request its current state and location from a task monitor.
- *Cooperation Means*: Dockets attached to folders or documents, task monitor with status information.

- *Cooperation Media*: Mail system.

5 A Flexible Cooperation Support

Once we have identified a set of workplace types and cooperation types, we have a basis for coping with different cooperative work situations in an application domain. Because if we start to combine workplace types with cooperation types, we realize that in general almost all combinations are feasible. As figure 1 shows, these combinations only differ to their degree of usefulness.

	Cooperation with Archives	Cooperation with PO boxes	Cooperation with Dockets
Expert workplace	**	**	**
Function workplace	*	-	**
Back-office workplace	*	**	**

Figure 1: Suitability of cooperation types for workplace types
[(**) very useful (*) useful (-) just working]

What does figure 1 mean? We found out that cooperation types and workplace types can almost freely be intertwined. This means in a concrete application domain, that we can provide substantial support for cooperative work for a wide range of working situations by recombining a few workplace types with a small set of cooperation means and media. The combination we actually choose, depends on what work analysis and job design (or business process reengineering) tells us.

6 References

Bäumer et al. (1997) D. Bäumer, G. Gryczan, R. Knoll, C. Lilienthal, D. Riehle, H. Züllighoven: Framework Development for Large Systems. CACM, October 1997, Vol. 40, No 10, pp. 52 - 59.

Grudin (1994). J. Grudin: Groupware and Social Dynamics: Eight Challenges for Developers. CACM, Vol 37, No 1, 1994, pp. 92-105.

Malone and Crowston (1994) T.W. Malone, K. Crowston: The interdisciplinary study of coordination. ACM Computing Surveys, Vol.26, No.1, pp.87-119

Mark et al. (1997). G. Mark, L. Fuchs, M. Sohlenkamp: Supporting Groupware conventions through Contextual Awareness, In W. Prinz, T. Rodden, J. Hughes, and K. Schmidt (eds.), Proceedings of ECSCW'97, Sept. 7-11, Lancaster, England, Kluwer Academic Publishers, Dordrecht, 1997, pp. 253-268.

Schmidt and Bannon (1992) K. Schmidt**,** L. Bannon: Taking CSCW Seriously: Supporting Articulation Work. In: Computer Supported Cooperative Work: An International Journal, 1 (1992) 1, pp. 1–33.