Process Support for Cooperative Work on the World Wide Web

Klaas Sikkel
Faculty of Computer Science, University of Twente,
PO Box 217, 7500 AE Enschede, the Netherlands
sikkel@cs.utwente.nl

Olaf Neumann, Sabine Sachweh
Faculty of Computer Science, University of Paderborn,
33098 Paderborn, Germany
{neumann,sachweh}@uni-paderborn.de

Abstract

The World Wide Web is becoming a dominating factor in Information Technology. Consequently, Computer-Supported Cooperative Work on the Web has recently drawn a lot of attention.

"Process Support for Cooperative Work" (PSCW) is a Web-based system supporting both structured and unstructured forms of cooperation. It is a combination of the "Basic Support for Cooperative Work" (BSCW) shared workspace system and the Merlin Process Support Environment. The current PSCW prototype offers a loose connection, in effect extending BSCW with a gateway to Merlin. With this prototype we have successfully addressed the technical issues involved; further integration of functionality should not pose any real problems.

We focus on the technical side of the PSCW system, which gives a good insight into the issues that have to be addressed generally in the construction of Web-based Groupware.

1. Introduction

The World Wide Web hardly needs an introduction. Originally it was designed as an infrastructure to improve the accessibility of scientific data [4]. One of the factors that may have contributed to it success is that the Web incorporated (rather than replaced) previously existing protocols like *gopher* and *ftp*, so that large amounts of legacy data were included in the Web from the start [5].

The explosive growth of the World Wide Web as a mass medium started with the emergence of browsers that were easy to use and easy to obtain. Meanwhile, the Web has changed the usage, if not the very nature, of the Internet.

With the Web becoming a dominating factor in Information Technology, not only in the Internet but also in the form of intranet solutions, it may have a profound impact on the Groupware market. Traditional groupware vendors (e.g. Lotus Notes) are putting a lot of effort in creating appropriate WWW-interfaces to their products. In addition, novel Web-based systems for cooperation support have started to emerge. See [2] for a collection of recent advances in this area.

In this article we introduce such a Web-based cooperative system, "Process Support for Cooperative Work" (*PSCW*). It is an extension of the *BSCW shared workspace system* [1] with a generic process engine that enables structured forms of cooperation.

Using the Web as a platform for groupware construction comes with a number of limitations, imposed by the architecture of the Web, but also has definite advantages. Cross-platform distribution comes for free and the system is well integrated with the typical work environment of the envisaged user group. In this paper we focus on the advantages and difficulties encountered in designing PSCW that arise from its embedding in the World Wide Web.

After a brief sketch of the Web in Section 2 and an introduction to BSCW in Section 3, Process support is elaborated upon in Section 4. We sketch the architecture of the PSCW system in Section 5. Discussion and conclusions follow in Section 6.

2. The World Wide Web as a platform for collaborative work

The Web, as it was originally designed at CERN, was intended to support a more active form of information sharing than what we find currently. In Addition to downloading documents and filling out forms, the HTTP specification included, from its earliest drafts, features for uploading documents to a Web server. At the time when the Web became popular, however, these features were not

0-8186-8332-5/98 $10.00

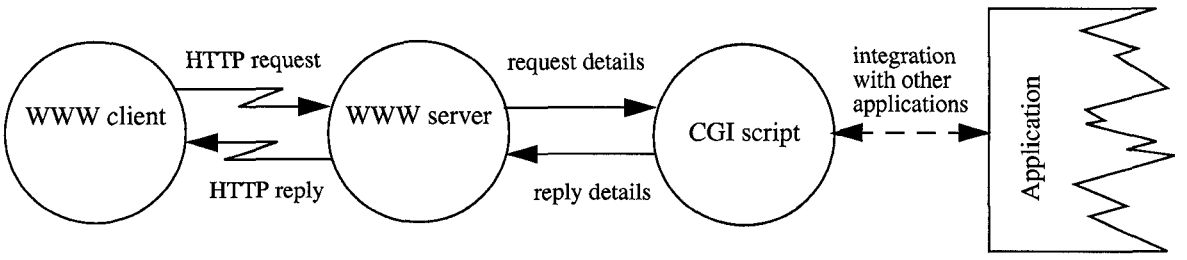

figure 1: Extending a Web server via the Common Gateway Interface

supported by the available servers and browsers and the Web became a more passive world-wide multimedia hypertext system.

The Web is based on a simple client-server architecture that allows clients to request information from servers. A server may reply to a request by sending a document stored in its file system or by passing on the request to a program that computes a reply. Web servers provide a standard API called the Common Gateway Interface (*CGI*). It specifies the request information that is passed on to a *CGI script;* the script is executed in real-time and expected to provide a reply that can be returned to the client. A CGI script, in turn, can call other applications. This general architecture is sketched in Figure 1.

Thus arbitrary applications can be interfaced to the World Wide Web, but the functionality of such applications should be able to cope with some specific constraints imposed by the Web's architecture.

- Web servers are passive, stateless servers and it was anticipated that communication only takes place when requested by Web clients ("browsers"). Limited possibilities for server-initiated communication (through Java applets, program fragments contained in HTML pages) are available with standard browsers.
- The Internet does not guarantee any Quality of Service, which is a serious problem when real-time constraints are involved. For example audio communication over the Internet is problematic and does match the quality of an ordinary telephone call.
- Not a fundamental problem, but a practical obstacle in using the Web as a front'end to an application is the lack of support for user interface design. HTML allows for mark-up tags and simple form-filling widgets, but does not support features common in desktop interfaces, such as drag-and-drop, multiple selection and semantic feedback.

The advantages of having a Web interface to an application are obvious. Users throughout the Internet can share the application. Moreover, the Web offers a truly *cross-platform* environment. This is particularly important for distributed cooperative work. Groupware, in order to be used, must be embedded in the regular working environment of its users. Distributed groups, even within the same organization, are usually faced with a heterogeneous environment in which different hardware and software is used within the group. Most groupware systems and prototypes are based on one particular platform and thus unsuitable for application in heterogeneous environments.

There are various degrees of Web integration of applications. Some applications require customized Web servers or browsers. When the interface to an application is realized exclusively through the CGI API, the system can be installed as an extension of an arbitrary Web server and, more importantly, can be employed by end users with arbitrary browsers.

Web-based applications that use regular browsers may still need additional software at the client side: so-called "helpers". When a browser receives a reply in a datatype it cannot interpret, it starts an appropriate helper to handle the data (if it knows where to find it). Novel helper applications can be added by defining additional datatypes and configuring the users' browsers accordingly.

It has considerable advantages when a Web-based application does not need application-specific helpers: The end users can interact with the system immediately, *without any software installation or configuration at the user side.* This eliminates much of the practical difficulties involved in getting a group of users started with a new application—in particular for groupware, where it is essential that everybody in the group has access to the system.

In large organizations a potentially very important advantage of not needing special software at the client side is that there are no software maintenance costs—which can be substantial for large numbers of workstations. This is one of the prime assets of Intranets.

3. Basic Support for Cooperative Work

"Basic Support for Cooperative Work" (BSCW) [1] is a Web-based groupware system, based on the notion of a "shared workspace." This is a repository for information,

accessible (only) to the group of members of the workspace. Workspaces may contain different kinds of objects: documents, threaded discussions, folders, etc. A basic form of version management is supported.

Users identify themselves with name and password (using the Web's "basic authentication" standard). Listings of workspace folders are presented as Web pages. Hyperlinks in form of text and buttons provide functionality for navigating through the workspace and for carrying out operations. These may involve forms for further interaction with the user.

Cooperation is unstructured (from the system's point of view), it is up to the members to organize a workspace and to coordinate the work.

Some *awareness facilities* are provided. Operations on objects trigger events, which are registered by the server and can be inspected by the users. The presence (or absence) of awareness icons in the user interface gives the user an overview, at a glance, of what has recently happened in a workspace.

"Soft locking" can be used to prevent simultaneous editing by different users. A soft lock warns the user that an object is claimed by another person, but does not physically prevent the user from accessing the object when she persists.

In May 1997 Version 3.0 of the system was released, offering search functionality and server-side document conversion as main new features. For Version 3.1, due November 1997, a more flexible access control mechanism is envisaged [21]. Support for synchronous cooperation is also being added [22]. Existing video conferencing and document conferencing tools are to be integrated into the BSCW system.

Likewise, in order to support more structured forms of cooperation within BSCW, it makes sense to integrate an existing tool rather than re-implement the functionality within the BSCW server. To that end the *Merlin* Process Support Environment [13]—to be discussed in Section 4—was selected.

The BSCW server is available for a variety of UNIX systems and for Windows NT and can be downloaded from the project's Web pages.[1] Communication between the BSCW server and its hosting Web server is based exclusively on the CGI API (from Version 2 onwards; The first version [3] employed a modified web server), hence a BSCW server can be added as an additional service to an arbitrary Web server. BSCW does not need special helpers (but there is one for more advanced usage, viz. uploading multiple documents in a single operation).

The fact that the end user only needs the standard software on his workstation in order to work with the system may account for much of the popularity of BSCW: GMD-FIT runs a public server which has currently more than 5000 users. Moreover, 1200 copies of the current version (3.01) have been downloaded in 4 months time, while many organizations are still using previous versions.

1. http://bscw.gmd.de/

4. Process support

Production processes involving multiple concurrently working users have to be coordinated in order to assure an acceptable degree of quality. A range of systems supports such coordination in a given, predefined process, e.g. workflow systems, cf. [10]. In software development the coordination of multiple participants is particularly critical and much of the advanced work in process support can be found in process centered software development environments [8], [9].

Process support systems need to be flexible, since many processes tend to change regularly, at least from one project to another. To allow an (ex-)change of processes so-called process modelling languages (*PMLs*) have been developed. A process support system is parametrized by a concrete process model described in a PML and executes this model to coordinate the process participants.

Most process support systems have been developed to be used in local environments (e.g. a LAN) to allow a centralized administration of the developed products, typically some sort of document, and of process information. Included within these documents is global process information like version histories of documents or the state of the executed process. In geographically distributed projects some form of distribution is needed.

The classical idea to solve this problem is to use distributed databases to store and access documents and, more importantly, the global process information. This approach has two major problems. The first is the availability of distributed databases and the second is that in consequence of database usage all information has to be stored in the used database, which massively restricts the tools usable in such an environment. A better alternative is to keep the data centralized and use the Web as a distribution medium.

Merlin [13] is the process executing part of a process centred environment arranged around a so-called process engine. Merlin executes processes described in a PML called ESCAPE [15]. ESCAPE is a language based on concepts and syntax of OMT [20] adapted to the specific needs of the process modelling domain. Stemming from the domain of software development, ESCAPE and Merlin are geared to support processes that comprise the production of electronic documents.

Consequently, the major items to specify a process with ESCAPE are the document types to produce, aggregation types of documents (called system types) and possible relations between instances of the object (document or system) types. For each object type a number of activities is

specified which can be applied to the object instances. State charts are used to describe the life cycle of an object type. Transitions are either events or activities. Events use inter-object relations to coordinate different object instances. Activities refer to real world operations modifying the object state. Furthermore, roles define responsibilities for a set of object types in certain states.

Of course these items are not complete and also not unique to ESCAPE but are commonly agreed to be basic elements of process descriptions in the software development domain, cf. [16], [7], [6], [23]. Compared to other approaches Merlin/ESCAPE provides a high level of abstraction for process specification and exploits the concept of inheritance in order to make the definition and adjustment of project or company specific processes as safe and as easy as possible, In addition, it has some specific features to specify processes with integrated SCM-support [17].

Merlin is a state based system. At each point in time Merlin is in a well-defined state which is used, for example, to restrict the work space of a user to a subset of all known documents, and for each document to a subset of all activities possible on the document, based on the state of the document and the role the developer is in. The current subset is presented to the user in a so-called agenda. In contrast to BSCW, where visibility and operations allowed on objects are static, the agenda in Merlin is context sensitive (to-do-list). Only those documents currently needing attention and activities currently possible are shown. Additionally, Merlin uses states for coordination purposes. For example, an implementation can be related to a design document by an *is_implemented_in* relation. If a designer creates a new design component or changes an existing one, a programmer working in a different workspace is informed if he is affected by the change.

5. Realization of the PSCW system

"Process Support for Cooperative Work" (*PSCW*) is an integration of Merlin and BSCW. The current version is a prototype that provides a rather loose integration. A second version, integrating parts of the functionality of both

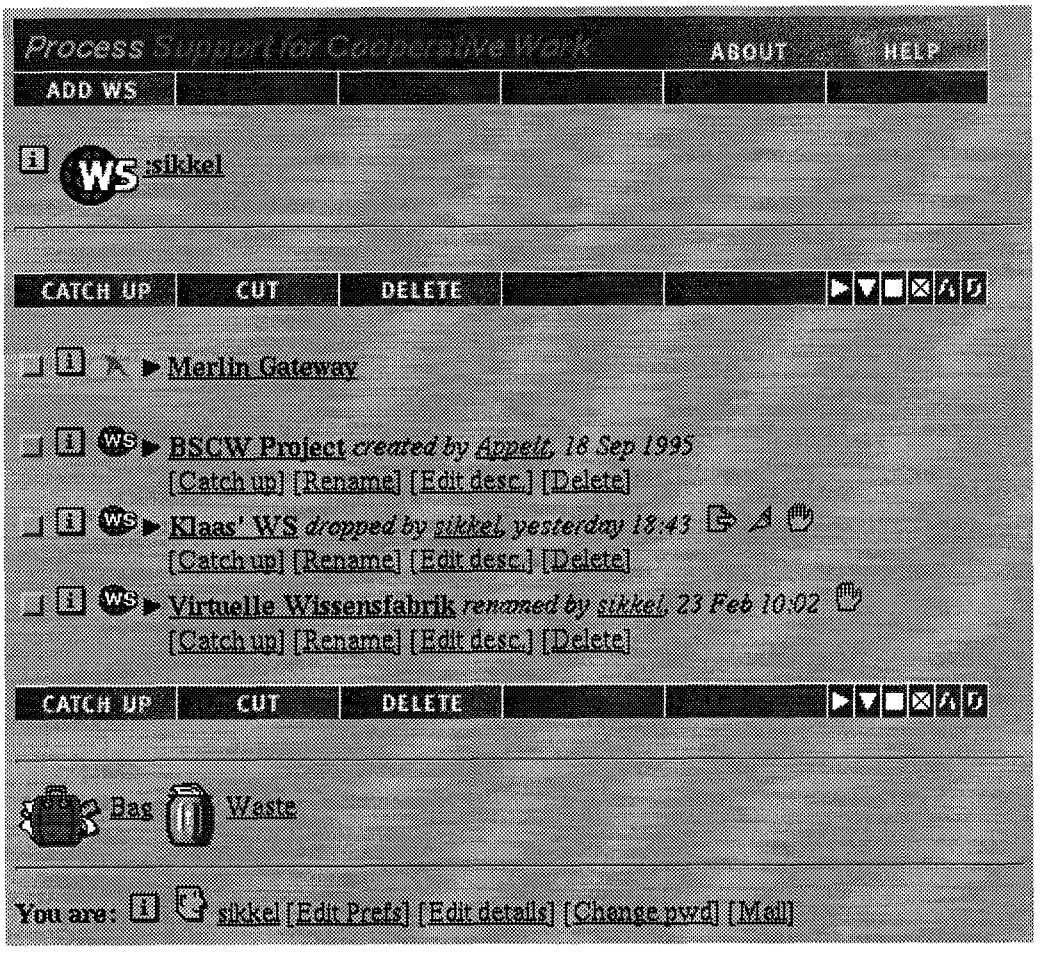

Figure 2: A user's index page in the PSCW prototype

systems, is due Spring 1998.

In the current version, the PSCW system is a BSCW server extended with a gateway to—and providing a user interface for—a Merlin process engine.

A user's index page shows the workspaces of which she is a member as well as a Merlin gateway. See figure 2. Clicking this link will connect her to the Merlin environment. The original Merlin user interface components have been replaced; the objects shown to the user are collected in an HTML page assembled by the BSCW server. A Merlin user may have several user interface components open simultaneously, for example: the *Working Context* showing the objects currently relevant for the user; a *History Browser* offering access to previous versions of an object. In PSCW, the contents of all user interface components is collated onto a single HTML page.

Figure 3 gives an example of a working context. Relations between objects are represented as annotations to these objects. A graphical representation of a relation (as in Merlin's usual user interface) is envisaged but has not yet been implemented. The emphasis in the current version was not in optimizing the user interface but in solving the underlying technical issues.

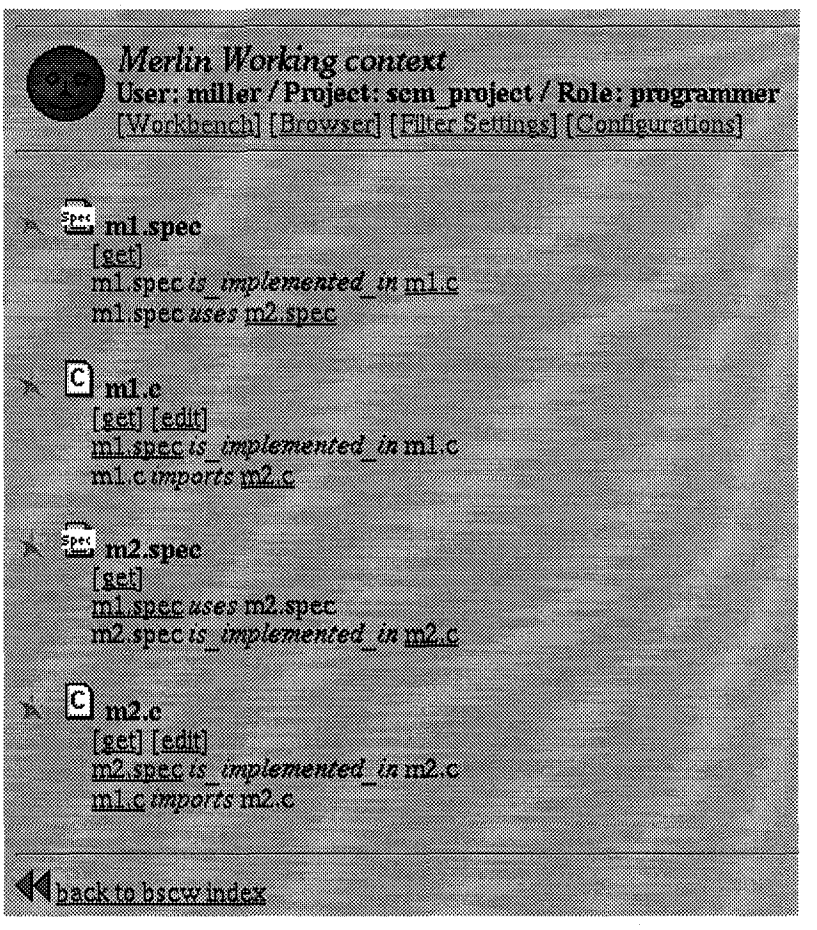

Figure 3: A working context

The BSCW server is implemented as a CGI script communicating with the BSCW persistent object store (hence its architecture is that of the generic Web-based application shown displayed in Figure 1). In PSCW, in similar fashion, The CGI script also communicates with the Merlin system. For technical reasons it proved easier, however, to add another interface component, the Mediator (cf. Figure 4), to streamline the interaction between the bscw SCI script and the Merlin components and eliminate some problems caused by the heterogeneity of these two systems.

A number of issues had to be addressed in order to connect Merlin and BSCW. Most of these are not particular for these two systems, but may generally apply in interfacing cooperative systems to the Web.

- *Lack of support for server-initiated communication.* This was addressed already in Section 2. Fortunately (and one of the reasons for selecting Merlin as the PSCW process engine), Merlin has the policy never to make changes to the user interface unless specifically requested by the user. The only server-initiated update is changing a single icon (the "smiley" in Figure 3), notifying the user that a user interface component no longer represents the current state. This is easily implemented by an applet.
- *Synchronous vs. asynchronous communication.* Merlin builds up a user interface by means of a sequence of asynchronous messages. For PSCW a "refresh" message has been added to the protocol, signalling that a user interface component will remain unchanged (except for the smiley) until the next user request. this is one of the functions of the Mediator at the BSCW side: It collects information from the different user interface components. When all user interface components relevant to a given user request have been refreshed, the Mediator assembles the reply in form of an HTML page.
- *Interfaces and protocols.* In line with our policy to build upon open standards and public domain software, we employed Xerox PARC's *ILU* system [14] to

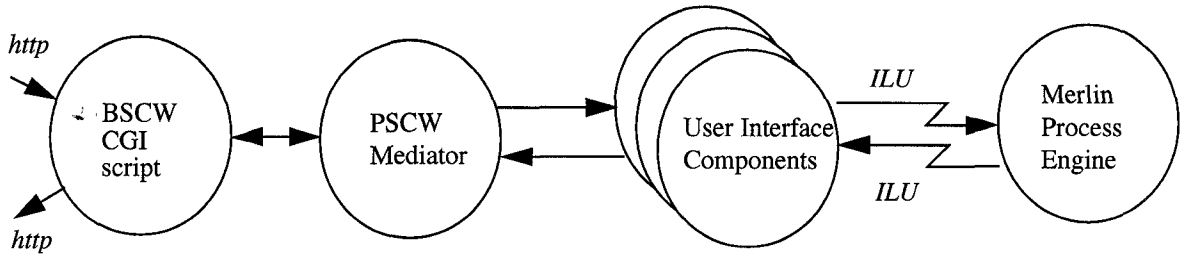

Figure 4: PSCW Architecture

implement the interface between Merlin and BSCW. It is based on OMG's CORBA Architecture and supports the languages (C++ and Python) in which both systems are implemented. The ILU system (version 2.0) proved satisfactory for cross-language remote procedure calls between the two systems.

The Merlin Process Engine and User Interface components in Figure 4 are implemented as ILU servers. The Mediator only serves components on a single host, all written in Python, and could be implemented more easily and efficiently as a Unix stream server.

- *Obsolete user Requests.* by having multiple browsers open or paging back through a browser's history, the user may request operations which are no longer applicable. The Merlin system does not check this—relying on the no longer valid assumption that the user can only access the system through the actions currently offered in the user interface. This could be solved with a check by the Mediator. If a user requests a no longer applicable operation, the request is not passed on to the user interface components but an appropriate message is returned to the Mediator immediately.

The resulting architecture is shown in Figure 4. Although the complete system involves a variety of protocols and processes, it does in fact provide a rather straightforward way to integrate two truly heterogeneous systems. Thus a central store for distributed process support, as discussed in Section 4, has been realized.

Note that the BSCW front end and the Merlin process engine need not be located on the same host or in the same LAN; an ILU binding service can be used to connect components at different Internet location. This has in fact been very helpful for the realization of the prototype in a distributed team.

6. Discussion and conclusions

The PSCW system realizes a distributed process support environment with a central data store. This is not an end product but a prototype, realizing a technical infrastructure that can be employed for further integration of the functionality of the two parts of the system. Cooperation, in most practical situations, involves a *combination* of structured and unstructured work [19]. In order to support this, a more seamless integration of BSCW and Merlin is desirable. User authentication, awareness services, and the exchange of objects between both parts of the system are natural candidates for a tighter integration.

This should be realized within the *Virtuelle Wissensfabrik* ("Virtual Knowledge Factory") research framework of the state of North Rhine-Westphalia. The PSCW system is planned to be the process support component of a support environment for distributed research. To that end, a study towards the feasibility of system support for research processes is being carried out by the Organizational Psychology group of the University of Bochum. Relevant subprocesses will be specified in (an extended version of) the ESCAPE process modelling language that was discussed in Section 4.

PSCW is not the only project aiming to bring Process Support to the Web. An interesting, but rather different approach is Rank Xerox' Webflow project [11]. In contrast to our centralized process engine it is based on a distributed architecture, employing coordination middleware.

ProcessWeb [12], [18] exploits an approach comparable to ours. In both cases an existing process engine (ProcessWise and Merlin, respectively) is connected to a distributed environment using the Web as a platform. Differences can be found on two levels. Firstly, and not specifically related to the Web, there are differences in the functionality offered by the process engine; see [8], Chapter 13 for a comparison. Secondly, and in this context more relevant, ProcessWeb replaces the ProcessWise front end with a WWW-based front end. PCSW, in contrast, aims to combine two systems from different areas in cooperative work—viz., BSCW to support uncoordinated group work and Merlin to support process centered multi-user work—into a new environment that is more than the sum of its parts.

The PSCW prototype discussed in this paper is but a first step towards such a more integrated Web-based groupware system. With this prototype, however, we have solved the underlying technical issues and shown the feasibility of the Web as a distribution medium for the integrated system.

Acknowledgements

This work was partially supported by the *Virtuelle Wissensfabrik* research framework of the Ministry of Science and Research of North Rhine-Westphalia.

The first author worked on this project while being a post-doc research fellow at GMD-FIT, Sankt Augustin, Germany.

References

[1] R. Bentley, W. Appelt, U. Busbach, E. Hinrichs, D. Kerr, K. Sikkel, J. Trevor, and G. Woetzel. Basic Support for Cooperative Work on the World Wide Web. *International Journal of Human Computer Studies* 46, 1997, pp. 827-846.

[2] R. Bentley, U. Busbach, D. Kerr, and K. Sikkel (Eds.). *Computer Supported Cooperative Work: the Journal of Collaborative Computing*, 6 (2-3), special issue on CSCW and the World Wide Web.

[3] R. Bentley, T. Horstman, K. Sikkel and J. Trevor. Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System. *The World Wide Web Journal.* Proceedings of the 4th International WWW Conference, Issue 1, December 1995, pp. 63-74.

[4] T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyck Nielsen, and A. Secret. The World Wide Web. *Communications of the ACM*, 37(8), August 1994, pp. 76-82.

[5] A. Dix. Challenges and Perspectives for Cooperative Work on the Web. *Computer Supported Cooperative Work: the Journal of Collaborative Computing*, 6 (2-3), 1997 pp. 135–156.

[6] M. Dowson. Software Process Themes and Issues. In *Proc. 2nd Int. Conf. on the Software Process*, IEEE Computer Society Press, 1993, pp. 54–62.

[7] P. H. Feiler and W. S. Humphrey. Software Process Development and Enactment: Concepts and Definitions. In *Proc. 2nd Int. Conf. on the Software Process*, IEEE Computer Society Press, 1993, pp. 28–40.

[8] A. Finkelstein, J. Kramer, and B. Nuseibeh (Eds). *Software Process Modelling and Technology.* Advanced Software Development Series. Research Studies Press Ltd., Taunton, Somerset, England, 1994.

[9] P. Garg and M. Jazayeri. Process-Centered Software Engineering Environments: A Grand Tour. In A. Fuggetta and A. Wolf (Eds.), *Software Process*, Trends in Software, Wiley, 1996, pp. 25–52.

[10] V. Gruhn. Business Process Modeling and Workflow Management. *International Journal of Cooperative Information Systems*, 4(2-3), 1995, pp. 145–164.

[11] A. Grasso, J. Meunier, D. Pagani, and R. Pareschi. Distributed coordination and workflow on the World Wide Web. *Computer Supported Cooperative Work: the Journal of Collaborative Computing*, 6 (2-3), 1997, pp. 175–200.

[12] R. M. Greenwood and B. Warboys. ProcessWeb — Process Support for the World Wide Web. *Proc. 5th European Workshop on Software Process Technology (EWSPT'96)*, Springer Verlag, 1996, pp. 82–85.

[13] G. Junkermann, B. Peuschel, W. Schäfer, and S. Wolf. MERLIN: Supporting Cooperation in Software Development Through a Knowledge–Based Environment. In [FKN94], Chapter 5, pp. 103–129.

[14] B. Janssen, M. Spreitzer. ILU 2.0 Reference manual. Xerox PARC, Palo Alto, Ca., cf. ftp://ftp.parc. xerox.com/pub/ilu/.

[15] G. Junkermann. A Dedicated Process Design Language based on EER-models, Statecharts and Tables. In *Proc. 7th Int. Conf. on Software Engineering and Knowledge Engineering*, University of Pittsburgh, 1995, Knowledge Systems Institute, pp. 487–496.

[16] J. Lonchamp. A Structured Conceptual and Terminological Framework for Software Process Engineering. In *Proc. of the 2nd Int. Conf. on the Software Process*, IEEE Computer Society, 1993, pp. 41–53.

[17] O. Neumann, S. Sachweh, and W. Schäfer. A High-Level Object-Oriented Specification Language for Configuration Management and Tool Integration. *Proc. 5th European Workshop on Software Process Technology (EWSPT'96)*, Springer Verlag, 1996, pp. 137–143.

[18] I. Robertson, B.C. Warboys, B.S. Yeomans. An evolvable software development environment. Unpublished draft, Dept. of Computer Science, University of Manchester, 1997.

[19] W. Prinz and A. Syri. Two complementary tools for cooperation in a Ministerial Environment. *Proc. First Int. Conf. on Practical Aspects of Knowledge Management*, Basel, Switzerland, 1996.

[20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object–Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, N. J. 07632, 1991.

[21] K. Sikkel. A Group-based Authorization Model for Cooperative Systems. *Proc. European Conf. on Computer Supported Cooperative Work (ECSCW'97)*, Lancaster, Sept. 1997, pp. 345–360.

[22] J. Trevor, T. Koch, and G. Woetzel. MetaWeb: Bringing synchronous groupware to the World Wide Web. *Proc. European Conf. on Computer Supported Cooperative Work (ECSCW'97)*, Lancaster, Sept. 1997, pp. 65–80.

[23] Workflow Management Coalition. Terminology & Glossary. WFMC Document WFMC-TC-1011, Workflow Management Coalition, Avenue Marcel Thiry 204, 1200 Brussels, Belgium, 1996. Issue 2.0.