42
Coordination support for CSCW-systems in concurrent engineering environments

Dipl.-Inform. Oliver Schumacher[1], *Dr.-Ing. Joachim Warschat*[2]

[1]*Institute for Human Factors and Technology Management, University of Stuttgart, Nobelstr. 12, D-70569 Stuttgart*

[2]*Fraunhofer-Institute for Industrial Engineering, Nobelstr. 12, D-70569 Stuttgart*

Abstract

Modern product development processes require increasingly the multi-disciplinary collaboration of experts but for the integration of specialized applications into a concurrent engineering environment mechanisms for coordinating sessions have to be established.

We describe the architecture of a system of multiple agents supplying the functionality of a sieve for messages sent to a common database system. Several floor policies are provided in a interpretable way in order to achieve user acceptance on the one hand and to maintain coordination of various applications in one session on the other hand.

Keywords

Coordination, collaborative work, concurrency control, concurrent engineering, floor passing, multi-agents, Tcl/Tk.

1 INTRODUCTION

The rapid technological change, the fact of diminishing product life spans and growing customer requirements are outlining conditions for modern product development which become more and more hard to please. Especially the communication and collaboration of experts of different

N. Terashima et al. (eds.), *Advanced IT Tools*

disciplines in the early phases of product development processes are important to guarantee the reduction of iteration cycles and therefore the time to market (Bullinger, 1995).

Whereas concurrent and simultaneous engineering (CSE) systems like CONSENS (Bullinger et al., 1996) provide software platforms for the integration of different tools on the basis of a common product model (Koch et al., 1994) there is no architectural support for collaboration in synchronous sessions in order to exploit all the information resources within an enterprise to improve design decisions in stages where little knowledge about the product is available.

2 PROBLEMS IN HETEROGENEOUS ENVIRONMENTS

Because of the diversity of problems and skills of the engaged engineers user-adapted, specialized tools with graphical user interfaces have to be still applied in order to increase the effectiveness in the development process - but considerations of concurrency control have to be taken into account now. So the meaning of coordination during sessions and therefore the problem of floor control is becoming important and the question for an architectural layer solving this problem in a flexible and scalable way arises.

Principally an approach which prevents simultaneous operations is not utilizing the potential of users working in parallel and therefore it is hampering the effectiveness of collaborative work. Synchronous CSCW-tools that are currently supporting nowadays product development processes like Whiteboards, Joint Editing or Joint CAD (Schumacher, 1995) use built-in coordination mechanisms like floor passing which coordinate the work of the users. These mechanisms consider the way users interact with the applications and are often designed to be as restrictive as necessary and as liberate as possible and are naturally strongly dependent on the interaction model. Although it is reasonable in some cases to limitate activity to a certain set of users this approach is too restraining for many other situations. Hence for the integration of tools with different user interfaces as described before there is the difficulty of combining different coordination mechanisms with different degrees of restriction in one session.

3 COORDINATION APPROACH FOR COOPERATIVE SYSTEMS

Cooperative systems are characterized by the fact that several users are interacting simultaneously with one or more applications which have access on a common data set. In order to work coordinated with these cooperative systems group protocols are applied. In contrast to technical protocols which are supported by hardware or software solutions the category of social protocols where activity is coordinated by the participants themselves seem to be well suited.

But especially when a latency of transmission appears and the number of session participants increases (Lauwers et al., 1990) or the behaviour of the participants is getting to disinhibited because of the lack of a social presence (Procter et al., 1994) the use of pure social protocols becomes inconvenient. On the other hand technical protocols used for the purpose of resolving interpersonal conflicts are criticized and the improvement of communication structures in order to increase the awareness of other participants is recommended by (Hughes, 1993). In addition to this there is the danger of human-computer conflicts which results in the frustration of users (Condon, 1993).

3.1 Design criteria

In order to achieve a compromise on the use of technical protocols and the social acceptance of a coordination mechanism it is reasonable to provide several floor passing policies per session. The diverse mechanisms can be classified by the following criteria (Lauwers et al., 1990) like

- the number of participants holding the floor simultaneously and therefore having access to write operations,
- the rules for managing the floor and
- the scope of the coordination mechanisms.

Another requirement for a coordination layer is the need for scalability. Users must be allowed to enter or to leave a running session at any time without resulting in an inconsistent state of the session and the coordination layer.

3.2 General architecture

For the integration of different applications into a collaborative environment we planned a „coordination layer" as an architectural support on the basis of floor policies. The coordination layer is designed to work as a sieve which is placed in between the applications and a global database system and is supposed to filter messages sent by applications in order to perform manipulations of common data. These write-messages are passed to the database if and only if the coordination layer recognizes by the reason of the chosen floor policy and its current state that the application is allowed to manipulate data.

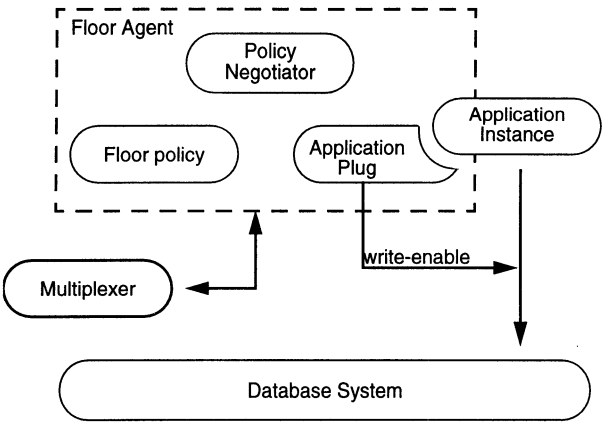

Figure 1 General architecture of a floor agent's components and the multiplexer.

With the presumption of having a global database system available which is propagating changes to all application instances by itself it is on the one hand not necessary to broadcast messages to all participants and thus on the other hand it is needless for an application to be able to interpret these messages in order to actualize its state and data. Therefore the present realiza-

tion of the coordination layer consists of a system of multiple floor agents with a replicating character which are bound to the applications and with each agent responsible exactly for one attached application process. The floor agents provides the interpreters for floor policies in order to make the usage of different mechanisms possible and the user interfaces to enable users to manage the floor.

For the communication between the floor agents a multiplexer that serializes all messages in order to achieve an ordered delivery was implemented. The overall architecture of the applications and the floor agents communicating via the message multiplexer is shown in Figure 1.

Floor policy

Due to the fact that several floor control mechanisms have to be taken into consideration we specify each floor policy within the coordination layer in a formal way in order to allow its interpretation by the floor agents.

The different mechanisms are described in an object-oriented way in order to take advantage of similarities between floor control mechanisms (see Figure 2) and therefore there is a class hierarchy available as the agent's floor policy. The complete functionality of a floor control mechanism and its according user interface is encapsulated into one class. Hence when the floor agent decides to choose a certain policy just an object-instantiation of the relevant class is necessary.

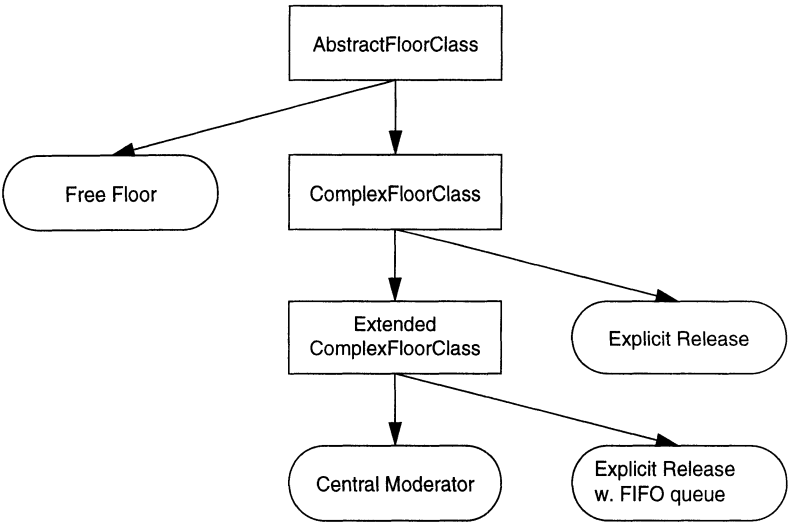

Figure 2 A class hierarchy for floor policies.

Application plug

For attaching an application to a floor agent it is necessary to build a wrapper that communicates with the application process. The floor agent supplies a write-enabled flag that has to be passed to the application.

Policy negotiator
When instantiating a session of different tools the agents start a negotiation about the mechanisms that will be selected for coordination. Each agent is told about the subset of mechanisms its application can handle and the role of its current user. If there is a intersection found as a result the session can be started with the least restrictive mechanism used. A similar negotiation will take place if a new session member is invited but is not able to handle the current coordination mechanism. If in order to keep a present session running an invitation is revised if the new negotiation is not successful.

3.3 Realization

A floor agent prototype is currently implemented with the object-oriented extension (Increment Tcl) to the Tool Command Language (Tcl/Tk) (Ousterhout, 1994) and the Tcl-DP supplement for communication. An interpretable language was chosen because of the idea of describing the policies in a formal way in order to be able to switch floor passing mechanisms during runtime arbitrarily. Different examples like e.g. whiteboards or application sharing tools and the thought of moderating video-conferences led to the realization of the following classes:

- In a free floor class there is no coordination at all and each user is allowed to write. It is the similar mechanism that is used in whiteboard or brainstorming applications.
- The pre-emptive floor class is a specialization of the free floor class. It restricts the floor to just one user but everyone can grab it.
- When using the explicit release class other user can't grab the floor until the user who is currently holding the floor releases it explicitly.
- The explicit release w. FIFO-queue class extends the explicit release class with a waiting queue.
- In the central moderator class there is one moderator who assigns the floor to one or more users. The moderator is privileged to revoke the floor as well.

4 RELATED WORK

Numbers of researchers proposed distributed problem solving technology for concurrent design. A Distributed and Integrated Environment for Computer-Aided Engineering (DICE) on basis of a blackboard architecture was presented in (Sriram et al., 1992) and an approach for a Distributed Intelligent Design Environments (DIDE) by applying multi-agent system technology was proposed in (Shen and Barthès, 1995).

A realization of coordination in distributed environments built upon a Shared Interface-Object Layer (SOL) focusing on simple graphical interaction techniques is described in (Smith et al., 1994) and techniques of Cooperative User Display Agents supporting different views are explained in (Bentley et al., 1992).
Real-time groupware concepts like shared context, group window or telepointer are shown in GROVE (Ellis et al., 1991). The idea of a common ground as a basis for groupware was explained in (Clark and Brennan, 1990) and most influence for the coordination layer is coming from the area of collaborative writing (Galegher and Kraut, 1994), (Sharples et al., 1993).

A knowledge-based model for Human-Computer-Human-Communication regarding the example of cooperative authoring of hypermedia-documents that considers a Coordination Com-

ponent based on a replicating CLOS implementation was researched in (Gunzenhäuser et al., 1994). Replicated architectures for groupware themselves are presented in (Borghoff, 1995).

5 FUTURE WORK AND CONCLUSION

The coordination layer is a flexible way to ensure floor control in groupware applications and different applications can be integrated by implementing corresponding application plugs. Currently we use the floor agents for the moderation of audio-/video-conferences as well.

A main advantage is the potential adaptability to different organizational structures and roles and the support of different views but as a matter of fact we don't provide role models at this time. Because of the ongoing work concerning the project management and role models in the product development process itself descriptions of floor control mechanisms are not irrevocable and have to be redesigned.

For the implementation of multiple sessions we will examine a cooperative transaction model for groupware-support on the database level and the multiplexer for communication purposes will be replaced by a group protocol like the reliable multicast protocol (RMP) (Montgomery, 1994) to guarantee order delivery and atomicity of messages.

6 REFERENCES

Bentley, R., Rodden, T. Sawyer, P., Sommerville, I. (1992) An architecture for tailoring cooperative multi-user displays, in *Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work.*

Borghoff, U. and Schlichter, J. (1995) Rechnergestützte Gruppenarbeit, Springer.

Bullinger, H.-J. (1995) Rapid Product Development: An Integrated Approach of Human, Technical and Organizational Resources, in *Proceedings of the 28th ISATA Conference*, Automotive Automation Limited.

Bullinger, H.-J. and Warschat, J. (Eds.) (1996) CONSENS: Concurrent and Simultaneous Engineering Systems, Springer.

Clark, H.H. and Brennan, S.E. (1990) Grounding in communication, in *Perspectives on socially shared cognition* (eds. Resnick, L.B., Leving, R.M., Teasley, S.D.); American Psychological Association.

Condon, C. (1993) The Computer Won't Let Me: Cooperation, Conflict and the Ownership of Information; *CSCW: Cooperation or Conflict?* (ed. Easterbrook, S.), Springer, London.

Ellis, C.A., Gibbs, S.J., Rein, G.L. (1991) Groupware: Some Issues and Experiences, *Communications of the ACM*, **34**, January 1991.

Galegher, J. and Kraut, R.E. (1994) Computer-mediated communication for intellectual teamwork: An experiment in group writing, *Information Systems Research*, **5**.

Gunzenhäuser, R., Dilly, W., Ressel, M. (1994) Auf dem Weg zur wissensbasierten Mensch-Computer-Mensch-Kommunikation, in *Innovationen bei Rechen- und Kommunikationssystemen*, (ed. Wolfinger, B.), Springer.

Hughes, P.T. (1993) Going Off the Rails: Understanding Conflict in Practice; *CSCW: Cooperation or Conflict?* (ed. Easterbrook, S.), Springer, London.

Koch, D., Diepold, T., Schumacher, O. (1994) Architectural Specification of the Product Information Archive, in *ESPRIT III 6898 CONSENS, Deliverable D3.2.*

Lauwers, J.C. and Lantz, K.A. (1990) Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems; in *Proceedings of the Human Factors in Computing Systems Conference, CHI-90* (eds. J.C. Chew and J. Whiteside), ACM, New York.

Montgomery, T. (1994) Design, Implementation, and Verification of the Reliable Multicast Protocol; *Thesis West Virginia University*, 1994.

Ousterhout, J. (1994) Tcl and Tk, Addison-Wesley, Reading.

Procter, R., McKinlay, A., Woodborn, R., Masting, O. (1994) Coordination Issues in Tools for CSCW; in Design Issues in CSCW (eds. Rosenberg, D. and Hutchison, C.), Springer, London.

Schumacher, O. (1995) Innovative Information Technologies Offer Improvements for the Product Development Process, in *Proceedings of the 28th ISATA Conference*, Automotive Automation Limited.

Sharples, M., Goodlet, J.S., Beck, E.E., Wood, C.C, Easterbrook, S.M., Plowman, L. (1993) Research issues in the study of computer supported collaborative writing, in *Computer Supported Collaborative Writing*, (ed. Sharples, M.).

Shen, W. and Barthès, J.-P. (1995) DIDE: A Multi-Agent Environment for Engineering Design, in Proceedings of the *First International Conference on Multi-Agent Systems*, (ed. V. Lesser), AAAI Press/ MIT Press, Cambridge.

Smith, G. and Rodden, T. (1994) An Access Model for Shared Interfaces, *Technical Report;* Computing Department, Lancaster University, UK.

Sriram, D., Logcher, R., Groleau, N., Cherneff, J. (1992) DICE: An Object-Oriented Programming Environment for Cooperative Engineering Design, in *AI in Engineering Design* (eds. Tong, C., Sriram, D.), **3**, Academic Press.

7 BIOGRAPHY

Dipl.-Inform. Oliver Schumacher studied Computer Science at the University of Stuttgart and currently holds a researcher position in the Special Research Centre on Rapid Prototyping (SFB374) at the University of Stuttgart. He is the technical manager of the project C3 (Team-oriented communication platform for collaborative work) within SFB374. His fields of work include computer-supported cooperative work, human-computer interactions, concurrent engineering and distributed databases.

Dr.-Ing. Joachim Warschat studied Mechanical Engineering at the University of Stuttgart and received his doctoral degree in Engineering in 1980. He is head of the department R&D-Management at the Fraunhofer-Institute for Industrial Engineering. His fields of work include organizational concepts, project management, simultaneous engineering, and rapid prototyping. Dr. Warschat is the author or editor of several publications in these fields.