Software Agents : Process Models and User Profiles in Distributed Software Development

Norbert Glaser
INRIA Sophia-Antipolis, Projet ACACIA
BP 93, 06902 Sophia-Antipolis, France
Norbert.Glaser@inria.fr

Jean-Claude Derniame
CRIN - Bat. LORIA
BP 239, 54506 Vandœuvre CEDEX
Jean-Claude.Derniame@loria.fr

Abstract

The development of software products has become a highly cooperative and distributed activity involving working groups at geographically distinct places. These groups show an increasing mobility and a very flexible organizational structure. Process methodology and technology have to take such evolutions into account. A possible direction for the emergence of new process technology and methodology is to take benefit from recent advances within multi-agent systems engineering : innovative methodologies for adaptable and autonomous architectures; they exhibit interesting features to support distributed software processes.

Keywords : software agents, process model , profiles.

1 Introduction

Software processes are complex activities involving several actors who need to cooperate on various levels for achieving coherent results. In the last three years several proposals advocated methodological support for cooperative and distributed development. This is an important issue since software processes becomes more an more spatially distributed (see [2] for an example). Within such an environment, process methodologies need to support heterogeneity, flexibility and a high mobility of working groups. Some process methodologies try to solve this issue through the integration of CSCW and process technology [6], others consider decentralized process modeling [16]. Our approach for supporting distributed process development is based on competence profiles and statistical process models. Both provide the necessary information for software agents [1] to cover with the flexibility of working groups and to maintain the quality of the resulting software product.

[1] A software agent can be defined as a small program performing a task in using information gleaned from its environment; it should be able to adapt itself based on changes occurring in its environment.

We advocate the use of competence profiles to facilitate the organization of and communication between flexible and mobile working groups. Competence profiles could be seen as identity cards representing the group members' competences which are required by the development process, and describing the individual roles within the working group. In other words, roles allow us to describe the organization of a working group and to formalize the migration between such groups [12]. Combining competence profiles with software agents will allow us to cover with the high mobility of people involved in the development process : flexible agent architectures would take in charge the exchange and management of information.

A second important question is how to cope with process assessment in a geographically distributed environment. The quality of a software product is already influenced by continuously changing requirements motivated by the speed of innovation and customer satisfaction. Now, the quality of the product faces moreover inconsistencies of decisions and actions and possible derivations from the prespecified norm. Process technology has to provide means to adapt quickly design and quality requirements without facing the redesign of the whole development process. Our proposal is inspired by the method for statistical process control which is a common approach within manufacturing for controlling the quality of a product. We claim that a statistical process model could be built for a development process and used by software agents to maintain the quality of the resulting software product.

Having user profiles and process models, we still need support for identifying and building the software agents which should manage the communication within working groups. The emergence of methodologies for multi-agent systems engineering (MAS) could have stimulative impulses on distributed program development as they include, among others, features like object-oriented design, generic model libraries, conceptual specification languages. We argue for the use of the CoMoMAS methodology [9], a knowledge engineering approach for MAS development. It provides tools for a functional analysis and identifies software agents and the related competence profiles.

2 Modeling Distributed Software Processes

Distributed software development associates the design, development, testing, validation and integration phases to geographically dispersed working groups. The discussion at IWRDPT'97[2] highlighted several important issues within distributed software development : how to limit possible derivations from a prespecified norm, how to increase visibility and transparency of decisions and actions, how to assess a distributed process, how to share information towards process evolution. This section details our methodology and models for distributed software development; a comparison to workflow management systems and traditional software engineering approaches is given in section 5.

2.1 The Software Process and its SPC Model

The quality of a software product is a central issue during software engineering and a critical one within a distributed working environment. Despite of validating and testing the software product at a final stage, we advocate to control quality continuously during the development process. An appropriate technique is statistical process control which was introduced by [13] as part of IBM's *Total Quality Management*. The principles of this technique, originally used for controlling a manufacturing process [11], are indeed very close to software process control.

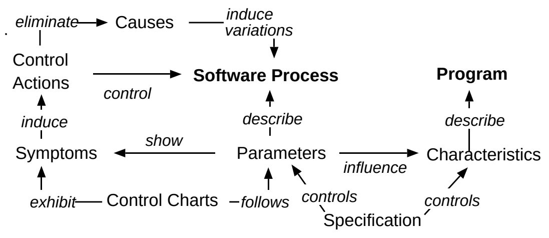
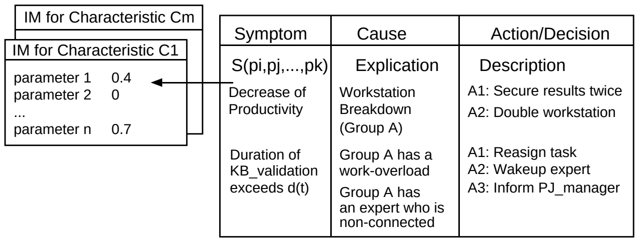


Figure 1. SPC Model of a Software Process

Figure 1 illustrates a simplified statistical process control model for software engineering; a more elaborated form is presented in [11]. The basic idea is to maintain the process within prespecified control limits and to guarantee in this way a certain quality of the software product. Thereby, information about the product is given by a set of characteristics which are associated with specification limits describing a prespecified norm for the product. Controlling the process means now to identify a set of parameters of the process and to maintain them within given control limits. Exceeds a parameter a given control limit or a pattern of values repeats over time, then the process is said to show some abnormal behavior and a symptom is detected. Some action needs to be taken in order to eliminate the cause of variation in the process behavior and to put it back under control. [13] gives categories of symptoms.

[2]International Workshop on Research Directions within Process Technology, Nancy, France



Symptom	Cause	Action/Decision
S(pi,pj,...,pk)	Explication	Description
Decrease of Productivity	Workstation Breakdown (Group A)	A1: Secure results twice A2: Double workstation
Duration of KB_validation exceeds d(t)	Group A has a work-overload Group A has an expert who is non-connected	A1: Reasign task A2: Wakeup expert A3: Inform PJ_manager

Figure 2. Example of an Influence Matrix

Monitoring all parameters is very expensive; only those parameters need to be controlled which have a considerable influence on the quality of the product. Statistical methods contribute to the analysis of the process and the identification of the parameters which need to be controlled. Figure 2 illustrates the *SPC process model* which is used for process control. If we want to guarantee a certain quality of a given characteristic, we use the the influence matrix (IM) to decide on the parameters to control. Each matrix qualifies the influence of parameters on a given characteristic.

What kind of parameters and characteristics are the most suitable for modeling a software process and product ? Parameters need to be measurable while characteristics are given in an engineering specification of a product. A possible set of characteristics can be non-functional design requirements [22], i.e. reliability, adaptability, modularity, maintainability, robustness, to cite only some of them. The identification of parameters depends on our understanding of a software process. We see a software development process as a set of tasks which are obtained through functional analysis of the process and can be refined in using day-to-day activity reports of software developers [20]. A task corresponds to a clearly identified goal within the software process and resumes the human actions for accomplishing it. A task uses input from other tasks, produces some output, requires resources, and has a certain duration and priority. We should be able to measure these tasks : a set of plausible parameters are the availability of resources (machines and humans), the coherence of the results of the working groups' activities, the degree of overhead for communication and coordination within working groups, the efficiency of human actors (training time/development time).

The SPC process model is instantiated for every software process but all software agents use the same instantiated model and maintain it current. The model allows software agents to support working groups who wish to reduce or to limit derivations from a prespecified norm and to visualize the influence of their actions on the quality of the program. Having detected a derivation, a software agent informs the group that it represents, or it asks another one of the same software process for support.

2.2 Modeling Working Groups

The software development team is assumed to be represented by a set of software agents. These agents support the collaboration between the members of a working group and between working groups. We illustrate such an environment in section 3. A software agent needs a profile of the user it wants to represent. A typical profile specifies the competences of the user, his interests, his role within the working group, his working status, and other useful information. We structure the competences into reactive (BHV), cognitive (PSM), cooperative (CPM) and social (SCM) ones [12]. The social competences of a team member are expressed by the role that he plays within the working group. It is of course possible to extend this definition of a user profile if we consider other aspects like the social behavior of working groups and their dynamic reorganization.

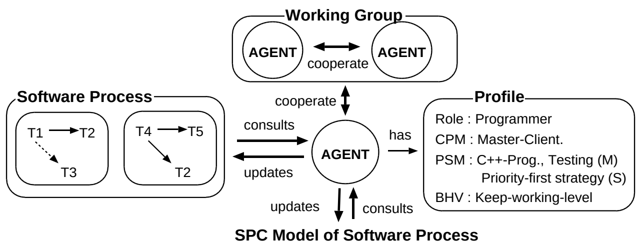


Figure 3. User Profiles and Working Groups

An example of a software agent is illustrated in Figure 3. It supports a user having the role of a programmer and whose competences include knowledge about C++ programming and testing, the ability to solve tasks which it selects priority-based; it receives tasks from other agents through a master-client protocol and is not capable to negotiate any task assignment. Finally, it tries to keep always a minimum level of activity.

The identification of the user profiles requires knowledge about the software process and the competences to solve the tasks into which it is decomposable through functional analysis. The next section presents the CoMoMAS, a multi-agent engineering approach, which offers an analysis method to obtain these profiles. User profiles need to be updated to keep current with the evolution of working groups. [15] presents a mechanism for agents to learn the behavior of users with whom they interact and to update user profiles.

2.3 Methodological Support

CoMoMAS [10] proposes a methodology and a framework for the development of multi-agent systems. CoMoMAS offers interesting elements for software engineering : the set of conceptual models which describe different view points on a software process; the reusable generic schemata representing results from functional, competence, cooperative and organizational analysis. CoMoMAS allows the transformation of conceptual models into executable programs, but this is within this paper of less importance; we are much more interested into the results from conceptual analysis of a software process; these results are used to define the software agents, the user profiles and the related competences.

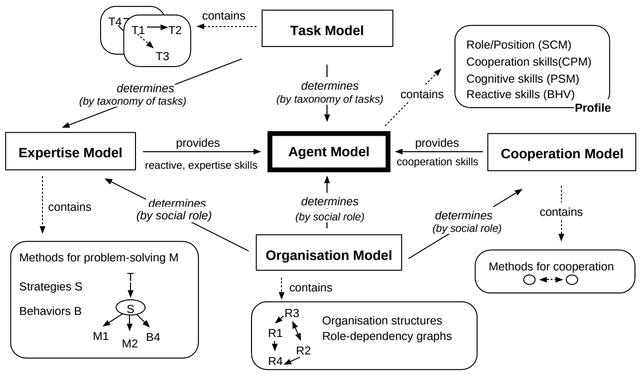


Figure 4. Conceptual Models for DSD

Figure 4 illustrates the five conceptual models for distributed software development (DSD). The models are inspired by the CommonKADS approach [21] for KBS development. The knowledge of the expertise, cooperation and organization models guide our definition of the competence profiles for the human actors solving the tasks of the software process which are represented in the task model. The conceptual agent models resulting from CoMoMAS for a working group are realized through one software agent. The task model represents the tasks and subtasks into which the development process can be decomposed through functional analysis; data and time dependencies between tasks are included. Solving a task requires various competences, problem-solving methods and communication skills, from the people being involved into the development process. Competence and cooperation analysis provide us this information : for example, programming skills, planning skills, coordination and negotiation methods. Not all members of a working group need to have the same competences; a working group is composed of members playing different roles. Roles describe the organization of a working group; the organization model represents this information and relates competences to roles.

3 Intelligent Agent Technology

Intelligent agent technology has found one of its first applications for systems and network management to enhance their performance. Recent applications areas are, among others, information access and management on the Internet, adaptive user interface to accommodate the increase its complexity, and workflow and business process management. Software agents [8] provide various types of support : they may make collaboration more efficient by sharing resources and by enhancing team work; they may enhance synergy and consistence of software processes by providing assistance during development, and they may reduce costs of human agents within process management by ascertaining and automating workflows and business processes.

Within software engineering software agents could be used as support for the automated exchange of information, the maintenance of process models, and in particular for the specification of joint-working plans and for solving conflicting goals during development. The idea is to delegate some organizational tasks to the agents which they are able to handle having acquired the competence profiles of the involved people.

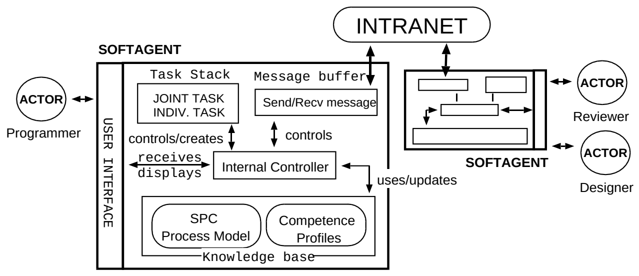


Figure 5. A Software Agent Architecture

Figure 5 illustrates a possible architecture for cooperative process development comprising two software agents. The agents support the collaboration of three users, i.e. the programmer, designer and reviewer, on a common software process. Each agent has a knowledge base composed of a SPC process model and a set of competence files modeling the human actors. This is the original part of our proposition rather than technical details for internal control, communication and cooperation between software agents. At the moment, we consider the exchange of information to be the principle interaction between software agent and human actors. An agent visualizes the composition and workload of the working group, the progress of the project and the list of tasks to be done by the human actor, and it asks for confirmation and the result if a tasks was completed.

The competence profiles specify an agent's possible roles, methods, strategies and cooperation protocols for solving tasks in joint action with other agents. An example is given in Figure 3 for a programmer. The profiles are at the basis of collaborative software development and can be used to specify, first, the required competences for a specific project, and second, the list of available human actors for a given software process. Profile databases could be an important contribution to the mobility which is found within distributed software development.

The presented architecture is being implemented and analyzed in the COMOMAS environment [9] which contains an extended version of the multi-agent simulation tool MICE [18]. The environment allows us to study profile-based software agents and the reorganization of working groups. The results will have an impact on the software agent architecture which we should support the construction of a knowledge server. Main characteristics of this project are : participation of geographically distributed working groups, flexible organization of working groups and a permanent evolution of the knowledge bases to be realized.

4 Application

The realization of a knowledge server on the intranet with corporate knowledge [5] for the analysis and reconstitution of traffic accidents involves working groups from three geographical locations. Working group A comprises a project manager, human factor specialists and software engineers who are specialized either in knowledge-base (KB) development or in software architectures. Working group C is composed of investigators and experts. Investigators are those people who collect the information about the traffic accident; experts are considered to have background knowledge about traffic accidents. We distinguish between infrastructure, vehicle and driver experts. Finally, group B is a mixed one of KB specialists and an infrastructure expert. The groups contribute partially to every phase of the software process as illustrated in Figure 6. The software product includes several knowledge bases, a user interface, and modules for updating, maintaining and exploiting the knowledge bases.

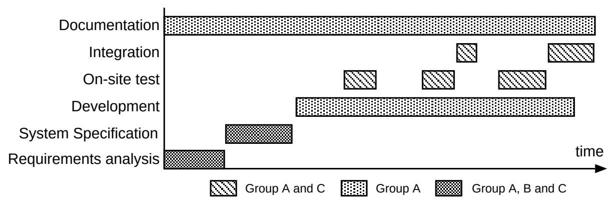


Figure 6. An Example for DSD

A process phase is decomposed into tasks which are distributed over the working groups. The tasks are obtained through stepwise functional analysis of the system to be built and its environment. Let us focus in the following on the development, on-site test and integration phases which involve two working groups. Figure 7 illustrates the control dependencies between some of the tasks obtained through functional analysis.

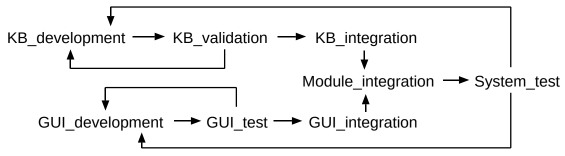


Figure 7. Task Model: Control Flow

Competence analysis of the tasks delivers us the categories of competences which are necessary to solve them. KB_development requires, for example, experiences about languages for knowledge representation (KR) and interview techniques (PSM), about progress monitoring (BHV) and about project management (CPM) in order to coordinate the development of the several knowledge bases if they are realized in cooperation. The assignment of these tasks to the members of working groups depends on their competences. A task may be assigned to several members depending on the competence profiles but attributed finally only to one of them because of, for example, the a temporary work overload of the others.

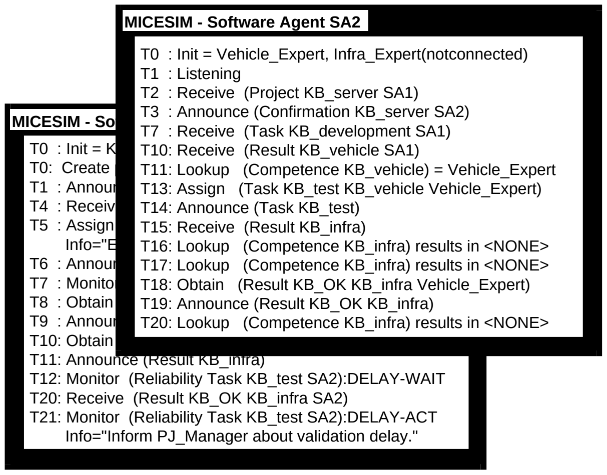
Analyzing the organization of the two working groups delivers us a typology of roles associated with different types of competences. Typical competence profiles for the members of the two working groups are given in the table below.

Role(s)	**Competences**
Project Manager	Project monitoring, Task assignment External reporting, Performance evaluation
Architecture engineer	Java programming, OMT methodology Intranet skills, DB management skills
KB engineer	Interview techniques, KR languages DB organization, KB validation
Human factor specialist	GUI development, HCI design, GUI test procedures
Investigator	GUI on-site testing
Expert	KB content validation (KB_infra, KB_vehicle, KB_driver)

Annotations : DB=Database, HCI=Human-Computer Interface.

The functional analysis of the software process and the analysis of the competences and the organization of the working groups allow us to decide on the kind of software agents that we need to implement for supporting distributed development. Figure 8 shows a possible information exchange between two software agents.

Agent SA1 represents the working group A which comprises a KB_engineer and a project manager; agent SA2 represents the working group C, that is, a Vehicle_Expert



```
MICESIM - Software Agent SA2
T0  : Init = Vehicle_Expert, Infra_Expert(notconnected)
T1  : Listening
T2  : Receive  (Project KB_server SA1)
T3  : Announce (Confirmation KB_server SA2)
T7  : Receive  (Task KB_development SA1)
T10: Receive  (Result KB_vehicle SA1)
T11: Lookup   (Competence KB_vehicle) = Vehicle_Expert
T13: Assign   (Task KB_test KB_vehicle Vehicle_Expert)
T14: Announce (Task KB_test)
T15: Receive  (Result KB_infra)
T16: Lookup   (Competence KB_infra) results in <NONE>
T17: Lookup   (Competence KB_infra) results in <NONE>
T18: Obtain   (Result KB_OK KB_infra Vehicle_Expert)
T19: Announce (Result KB_OK KB_infra)
T20: Lookup   (Competence KB_infra) results in <NONE>
```

```
MICESIM - So
T0  : Init = K
T0:  Create
T1  : Annou
T4  : Receiv
T5  : Assign
      Info="E
T6  : Annou
T7  : Monito
T8  : Obtain
T9  : Annou
T10: Obtain
T11: Announce (Result KB_infra)
T12: Monitor  (Reliability Task KB_test SA2):DELAY-WAIT
T20: Receive  (Result KB_OK KB_infra SA2)
T21: Monitor  (Reliability Task KB_test SA2):DELAY-ACT
      Info="Inform PJ_Manager about validation delay."
```

Figure 8. Monitoring of Software Agents

and an infrastructure expert. The infrastructure expert is on stand-by mode. The agents are collaborating on the development and validation of a knowledge base. The announcement of the first results of the KB_development task at time t=T9 from agent SA1 motivates agent SA2 to look for group members who are able to validate these results. The matching mechanism between required and available profiles is based on keywords. A more elaborated version is discussed in [12]. Agent SA1 starts monitoring the reliability of the evaluation procedure and recognizes at t=T12 that a delay has occurred which it tries to resolve at t=T21. The knowledge how to monitor a task and how to resolve the variation in the validation procedure is provided by the SPC process model (see Figure 1).

5 Discussion

Multi-agent technology provides useful means to support distributed software development. In fact, the use of agents allows us to cover with the many possible ways for achieving objectives within a single application and to reach a balance between the long-term and short-time objectives. The desire for an efficient use of the multi-agent technology, i.e. software agents [17], has also made research on methodologies very popular. The functional analysis within a methodology, like CoMoMAS, demonstrate the closeness of multi-agent and software engineering [14]. Functional models are an interesting starting point for development since they allow us to identify clearly control and data dependencies; this is valuable information which can be used to decide the distribution of tasks between agents and human developers. The advocated functional analysis of a software process can be compared to workflow management systems [1] : process analysis delivers in a bottom-up way a set of activities which are linked by control and data connectors. Nevertheless, reusability and generic libraries are not much addressed within workflow management.

An integration could be realized between our proposal and workflow-like approaches. [4] proposes a component approach to support distributed team-based software development. It relies on a process model called LCPS (Life Cycle Process Support) [3] and aims at supporting the definition of the process that is to be employed for a particular development step, the subsequent enactment of that process (instantiation of its components), its performance (execution by the different agents) [7] and the distribution of process components on a network. A process model component is a set of entities (agents, roles, activities, products, directions), linked together inside a schema around a central activity. In this model, activities are represented by traditional tools and the knowledge dimension is not present; the autonomy of agents is not explicit. Reusability is addressed in LCPS by meta-model entities which are comparable to abstract classes in the object-oriented sense.

The SPC process model of this paper gives a more flexible approach to support team cooperation relying on competences. Enriched by a support to negotiation between partners, such an architecture would be able to cover not only the functional aspects of the distributed software development but also the implementation.

6 Conclusion

The paper presented a methodology for distributed software development and multi-agent technology to support it. Our main contribution are a statistical process control (SPC) model and competence profiles. A SPC process model, originally used for the control of a manufacturing process, allow us to specify monitoring knowledge about a software process and to use it within a distributed environment as a common source of information for maintaining the quality of the software product. Competence profiles characterize the expertise and organization of working groups who participate in the software process. An architecture of software agents is presented which exploits SPC models and competence profiles for supporting the coordination and information exchange within a distributed working environment.

Considering our aim to model the organization of working groups and their interaction, we may find an interesting extension of our work in [19]. The cited paper proposes a language for the interaction between software teams. It includes roles to model the organization of these teams.

7 Acknowledgments

The authors would like to thank the members of the IWRDPT'97 workshop for the enthusiastic discussions about multi-agent systems within software engineering.

References

[1] G. Alonso and C. Mohan. *Advanced Transaction Models and Architectures*, chapter Workflow Management: the Next Generation of Distributed Processing Tools. Kluwer Academic Publishers, 1997.

[2] V. Delotti and S. Bly. Walking away from the desktop computer: Distributed collaboration and mobility in a product design team. In *ACM CSCW*, pages 209–218, 1996.

[3] J.-C. Derniame and al. Life cycle process support in pcis. In *PCTE*, 1994.

[4] J.-C. Derniame, A. Kaba, and P. Tiako. A process model with distributed and composable components. In *ICSP5 (submitted)*, 1998.

[5] R. Dieng, A. Giboin, O. Corby, and al. Building of a corporate memory for traffic accidents. *AI Magazine*, 1998.

[6] E. DiNitto and A. Fuggetta. Integrating process technology and cscw. In *EWSPT*, pages 154–161. LNCS 913, 1995.

[7] M. Dowson and C. Fernstrom. Towards requirements for enactment mechanisms in software process technology. In *EWSPT*, LNCS 772, pages 90–106, 1994.

[8] M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 7(37):48–53, 1994.

[9] N. Glaser. The CoMoMAS methodology for multi-agent system development. In D. Lukose and C. Zhang, editors, *Second Australian Workshop on DAI*. LNAI Series, 1996.

[10] N. Glaser. *Contribution to Knowledge Acquisition and Modelling in a Multi-Agent Framework - The CoMoMAS Approach*. PhD thesis, U. Henri Poincaré, Nancy I, 1996.

[11] N. Glaser and M.-C. Haton. Experiences in modelling statistical process control knowledge. In *Proc. 12 th ECAI*, 1996.

[12] N. Glaser and P. Morignot. The reorganization of societies of autonomous agents. In *MAAMAW*. LNAI 1237, 1997.

[13] E. Kane. IBM's quality focus on the business process. *Quality Progress*, pages 24–33, 1980.

[14] D. Kinny and M. Georgeff. Modelling and design of multi-agent system. In *ECAI-96 workshop on ATAL*, pages 1–20. LNAI 1193, 1996.

[15] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *Proc. AAAI-94*, Seattle, WA, 1994.

[16] U. Leonhardt, J. Kramer, B. Nuseibeh, and A. Finkelstein. Decentralised process modelling in a multi-perspective development environment. In *ICSE*, 1995.

[17] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37, July 1994.

[18] T. Montgomery and E. Durfee. Using MICE to study intelligent dynamic coordination. In *Proc. 2 nd IEEE Conf. on Tools for AI*, pages 438–444, 1990.

[19] D. E. Perry. Using process modeling for process understanding. In *Software Process Improvement*, Barcelona, 1997.

[20] D. E. Perry, N. Staudenmayer, and L. G. Votta. People, organizations, and process improvement. *IEEE Transactions on Software Engineering*, July 1994.

[21] G. Schreiber, B. Wielinga, R. de Hoog, H. Akkermans, and W. V. de Velde. Commonkads: A comprehensive methodology for KBS development. *IEEE Expert*, 9(6):28–37, 1994.

[22] J. Vanwelkenhuysen. Embedding non-functional requirements analyses in conceptual knowledge system design. In *Proc. of 9 th KAW*, Banff (CAN), 1995.