A Synchronous Cooperative Architecture for the PROSOFT Software Engineering Environment[1]

Rodrigo Quites Reis[*, #]
Carla Alessandra Lima Reis[*, #]
[*]*Universidade Federal do Pará - UFPA*
Departamento de Informática
Belém - PA – Brazil

Daltro José Nunes [#]

[#]*Universidade Federal do Rio Grande do Sul*
Instituto de Informática
Programa de Pós-Graduação em Computação
Porto Alegre - RS - Brazil

E-mail: *{quites, clima, daltro}@inf.ufrgs.br*

Summary

This paper shows the evolution of a software engineering environment (SEE) called PROSOFT to support the formal development of groupware applications. This environment, which is centered in the data-driven approach for software development, evolved to support cooperation in the software development process. Its transition is founded in a client/server communication model called Distributed PROSOFT that provides software mechanisms to permit concurrent use of the environment resources. Thus, this paper presents a formal model that provides an object middleware with synchronous handling and version support for the objects created with the software tools integrated to the environment. Cooperative PROSOFT is presented as an architecture for the formal development of groupware applications that permits the formal validation of cooperative applications specified under its paradigm. A consequence of this work is the integration of the advantages found in formal specification techniques to the groupware development that provides the development of higher quality groupware applications than those obtained with the use of traditional techniques.

Keywords: CSCW, Software Engineering, Cooperative Software Engineering Environment, Formal Methods, Algebraic Specification, Version management.

Resumo

Este artigo mostra a evolução de um ambiente de engenharia de software chamado PROSOFT para apoiar o desenvolvimento formal de aplicações *groupware*. Este ambiente, norteado pela abordagem *data-driven* de desenvolvimento de software, evoluiu para suportar o trabalho cooperativo no desenvolvimento de software. Esta transição é baseada em um modelo de comunicação cliente/servidor chamado PROSOFT Distribuído que provê mecanismos para o uso concorrente de recursos do ambiente. Assim sendo, este artigo apresenta um modelo formal que estabelece um meta-gerenciador de objetos PROSOFT com recursos de cooperação síncrona e versionamento de objetos produzidos pelas ferramentas integradas ao ambiente. O PROSOFT Cooperativo é apresentado como uma arquitetura de suporte ao desenvolvimento de *groupware*baseado em especificações formais que permite a validação formal de propriedades das aplicações cooperativas especificadas sob o seu paradigma. Uma consequência deste trabalho é a integração das vantagens das técnicas de especificação formal ao desenvolvimento de *groupware*, proporcionando o desenvolvimento de aplicações *groupware* de qualidade superior que as obtidas com o uso de técnicas tradicionais.

Palavras-chave: Trabalho Cooperativo Auxiliado por Computador, Engenharia de Software, Ambiente Cooperativo de Engenharia de Software, Métodos Formais, Especificação algébrica, Gerência de versões.

[1] This work is partly supported by CAPES, CNPq and FAPERGS.

1. INTRODUCTION

First, we will discuss the characteristics of the Cooperative Software Engineering and implications of the use of Computer Supported Cooperative Work (CSCW) techniques in the construction of SEEs. Next, we will present the PROSOFT SEE, developed at the Institute of Computer Science, Universidade Federal do Rio Grande do Sul (UFRGS) at Porto Alegre - Brazil, under the direction of Prof. Dr. Daltro José Nunes. Finally, the cooperative extension to the PROSOFT SEE is presented, showing its main architecture and implications for the development process of groupware applications.

2. CSCW AND SOFTWARE ENGINEERING

Software Engineering is a social or a group activity, where the work of many people with different experience, expectation and expertise is needed ([TOP 95] [VES 95]). Software teams need to work together to reach a common main goal, that is, the development of a software product. Study of DeMarco and Lister pointed at [VES 95] says that the developers of large systems stay 70% of the time working together. Similarly, Jones reported that group activities (communication, coordination and negotiation) take 85% of the cost of large systems of software.

Communication is more frequently the most difficult aspect of groupware and is especially important in Software Engineering. To resolve conflicts that appear from the cooperative activity, the techniques of CSCW and Software Engineering are being combined, which result in the development of cooperative SEEs. The main idea behind groupware is the view of a shared environment compound by cooperative objects. Some aspects that encourage the integration of CSCW and Software Engineering ideas are:

- The development of very large systems, where many professionals are involved;
- The need to increase the speed of software development;
- The introduction of new members to a running project, traditionally traumatic [PRE 93].

3. THE PROSOFT SOFTWARE ENGINEERING ENVIRONMENT

PROSOFT is a software development environment built to support the formal development of software, providing the data, control and presentation integration between the tools developed under the environment. Thus, PROSOFT is a framework for software engineering tools, providing basic and standard services. The specification of this environment was guided by: the data-driven approach for software development [NUN 95], the model notion [BJØ 78], abstract data types and the object-oriented paradigm.

The PROSOFT environment consists of a set of homogeneous object-oriented software tools called **Ambiente de Tratamento de Objetos**[2] (ATOs). Every ATO is described by its class and a set of operations to create, manipulate and visualize objects that are belonging to that class. The construction of one or more ATOs represents the solution of a problem. Communication between the ATOs is made by an interface routine called **Interface de Comunicação do Sistema**[3] (**ICS**).

[2] Object Handling Environment

[3] System Communication Interface

3.1. ATO

ATOs are the basic components of the tools developed. Each ATO corresponds to a class (abstract data type) encapsulation and a set of operations that manipulate these class instances. Each ATO is divided in three parts:

- A **class**, defined by a graphical notation that shows the composition of data types; the class instances are called objects;
- The **interface** and the **operations** (semantics**)** of the defined operations, similar to the signature and axiom notions presented in [WAT 91].

The general ATO structure is shown in figure 1.

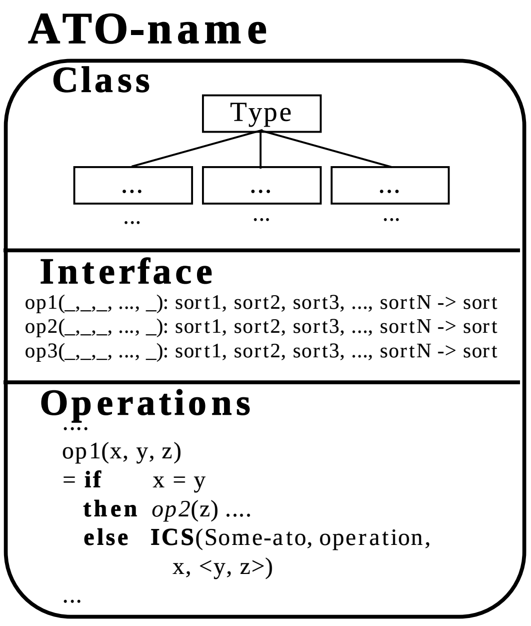

ATO-name
Class
Type
...
...
...
...
...
...
Interface
op1(_,_,_, ..., _): sort1, sort2, sort3, ..., sortN -> sort
op2(_,_,_, ..., _): sort1, sort2, sort3, ..., sortN -> sort
op3(_,_,_, ..., _): sort1, sort2, sort3, ..., sortN -> sort
Operations
....
op1(x, y, z)
= if x = y
then op2(z) ....
else ICS(Some-ato, operation,
x, <y, z>)
...

Figure 1 ATO Structure

Several software-engineering tools were specified and integrated to the environment. Often a software system is composed by some ATOs. Therefore, a mechanism is necessary to ensure the correct use of an ATO determined function without incurring in ambiguity [NUN 89]. If an ATO operation needs a service provided by another ATO operation, one message should be dispatched to the requested ATO with the operation and arguments. This service is provided, in the PROSOFT environment, by the ICS. The current implementation of the ICS mechanism is based on the *Remote Procedure Call* paradigm, as described in the next section.

3.2. DISTRIBUTED PROSOFT

The actual trend to provide software mechanisms to permit the concurrent use, letting users share information brought a new characteristic to PROSOFT: allow its use in a shared and distributed way. So, the Distributed PROSOFT was developed and is presented in ([SCH 95] [SCH 97a]).

The characteristics that separate Distributed PROSOFT from the traditional PROSOFT are:

- The environment is designed in the accordance with a **client/server** model in conformance of OSF/DCE (Distributed Computing Environment) [ROS 92];

- Specific ATOs can be started on demand as a reflection of the users' needs. The server that processes these requests is called **ATO-Server**. An ATO-Server is associated with a single tool or with a set of intrinsically related tools which might be grouped to form an ATO package;
- Services of an ATO-Server can be **shared** concurrently by many users. ATO-Servers are configured as multi-user servers being able to handle asynchronous client requests of concurrent users in parallel by attaching individual threads of control to each request (**multi-thread server model**);
- The user can **configure the distribution** of the system in a flexible way, that is, the addition or removal of ATOs in his section is provided;
- The ICS implementation was redesigned (based on the Remote Procedure Call mechanism) to allow transparent interprocess **communication between clients and servers**.

However, despite these new characteristics, some still were needed to consider PROSOFT a cooperative SEE. The next section will present the guidelines of the development of **Cooperative PROSOFT**.

4. COOPERATIVE PROSOFT

The cooperative SEE main goal involves the concurrent use of software objects (diagrams, program listings, etc.), allowing a same object being manipulated by several developers.

Distributed PROSOFT environment represents a first step toward this purpose, providing mechanisms for information sharing. However, the support for cooperative synchronous object handling was not available in Distributed PROSOFT.

Cooperative PROSOFT is an extension to the PROSOFT kernel specified with the formalism presented at [NUN 95]. Thus, some ATOs were developed and integrated to the environment. [REI 97] presents its static and functional views, while the dynamic aspect of this work was specified using ISO/LOTOS (Language of Temporal Ordering Specification) and was influenced by We-Met [REK 93].

From an architectural point of view, Cooperative PROSOFT is a Distributed PROSOFT component, providing user, PROSOFT workstation and cooperative object management. Cooperative PROSOFT architecture is presented next by its functionality informal description.

4.1. COOPERATIVE PROSOFT ARCHITECTURE

Cooperative PROSOFT is a set of eight ATOs specified to manage the synchronous manipulation of cooperative PROSOFT objects. It is a middleware framework that also provides (figure 2):

- An orthogonal version-object model (based in the work published at [GOL 96]);
- Undo/redo facilities;
- Users, workstations and tools management;
- Fine grain object access rights.

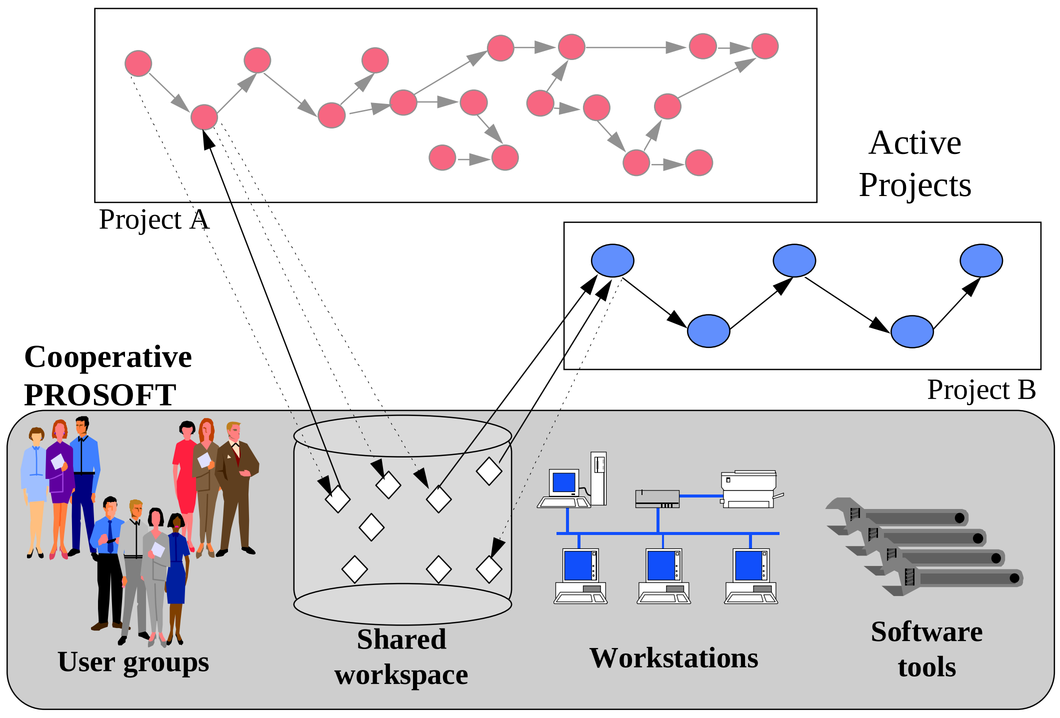
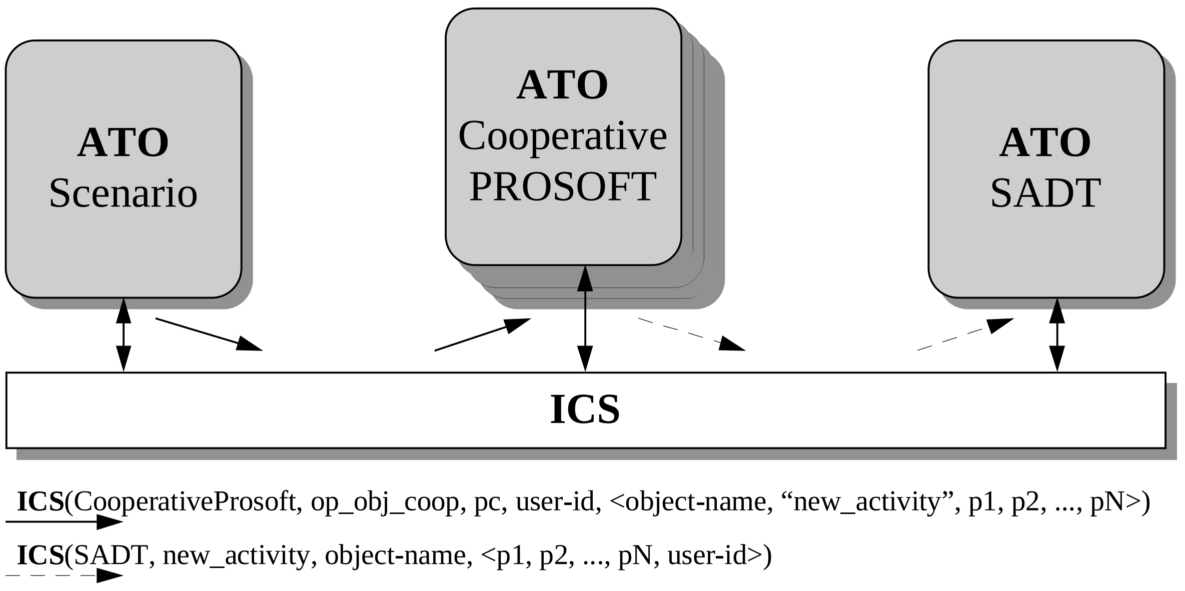


Figure 2 Cooperative PROSOFT main components

Cooperative PROSOFT manages all the communication between the user and the groupware ATOs available in the environment. Figure 3 shows a diagram that explains the strategy adopted. An user requests (through the ATO Scenario, the PROSOFT ATO responsible for all end-user interactions) one operation (called *new_activity*) to modify an object (called *object-name*). All this information is sent to the Cooperative PROSOFT ATOs: they verify if the user has an access right to the object and make another ICS call to the ATO associated with the object (in this example, SADT[4]).



ICS(CooperativeProsoft, op_obj_coop, pc, user-id, <object-name, "new_activity", p1, p2, ..., pN>)

ICS(SADT, new_activity, object-name, <p1, p2, ..., pN, user-id>)

Figure 3 Cooperative PROSOFT acts as mediator between the user requests and the destination ATO

[4] The SADT ATO is responsible for the creation and execution of SADT ([ROS 77] [ROS 85]) diagrams.

4.2. USER MANAGEMENT AND OBJECT STATE

Every object in the environment is associated with a set of users and groups that have different access rights. The union of users' and public workspaces is the **shared workspace**. We defined an user taxonomy (based on the role played in this cooperation model):

- The **Creator** is the user who created an object;
- The **Editor** is an user that can modify the object;
- The **Visualizer** is an user that can watch a cooperative session (not allowed to do any action with the object);
- The **Manager** can act as a moderator in the cooperative session of the object, and, for example, is allowed to modify the state of the object, to specify an access right for an user and undo the operations made by some user in the object.

States were defined for the objects (or versions) revealing the maturity degree reached by the object. An object is created in the **unstable** state, where only the creator has the right to modify. Through a creator request, it is promoted to the **stable** state, allowing some users to work in a cooperative way. From a manager request, the object becomes **consolidated**. Further modification is not allowed (consolidated objects can have derived versions). Figure 4 shows the state transition diagram for PROSOFT objects.

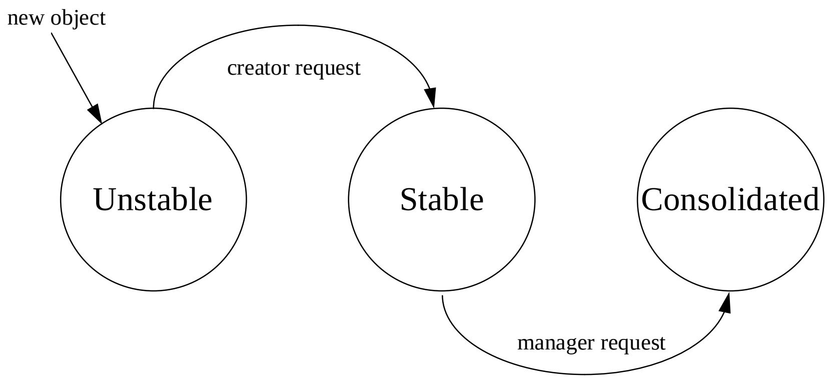


Figure 4 The cooperative object state transition diagram

4.3. GROUPWARE DEVELOPMENT IN PROSOFT

We developed the Cooperative PROSOFT framework with a main goal: provide an easy to follow upgrade path for the existing PROSOFT ATOs. This was achieved though the minimization of changes needed to provide cooperative object management in an existing ATO.

First, each ATO operation should be classified as **viewer** (do not modify an object) or as **modifier** (an operation that modifies the object). To develop a Groupware-ATO, we need to treat only modifier operations: all the cooperative users should be immediately notified about the modification of a stable object.

In groupware applications is often needed to identify which user made a modification in an object. So, the second step is to extend each modifier operation to use a new feature added with Cooperative PROSOFT: every performed operation is associated with the identification of the user who requested it (as shown in figure 3, the last argument of the function call to the Groupware-ATO is the user-id). This additional information can be used to differentiate the effect of an user operation in a shared object. Finally, the undo and redo services (provided by this framework) can be added to the Groupware-ATO graphical user interface.

The SADT ATO was modified to support these new features. Figure 5 depicts the cooperative use of a shared diagram: specific users use this feature to identify with different colors and texture the graphical objects included.

4.4. RELATED WORK

Cooperative PROSOFT major contribution for the CSCW field is the new approach adopted to attack the problem: the use of formal specification techniques to describe and validate groupware applications. The literature describes several cooperative SEEs and CASE tools [VES 95] [TOP 95] [ARA 97]. However, besides the benefits of the adoption of formal methods in the software development life cycle, its use in the specification of groupware application is limited.

The validation of groupware applications is often the most difficult issue to achieve mainly because a lot of observation is required. The use of software tools to test and verify the specifications, as provided by the tools available at the PROSOFT environment [NUN 95], help the developer to focus only in the important aspects of groupware construction.

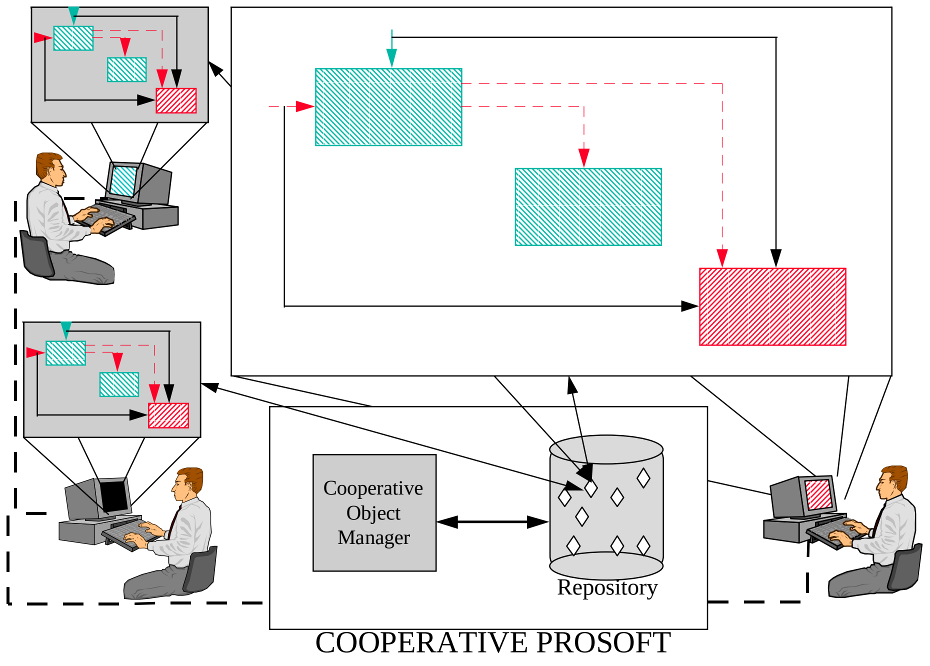


Figure 5 Cooperative use of a shared SADT object

5. CONCLUSIONS AND FUTURE WORK

This paper showed the evolution of the PROSOFT SEE to support the development of groupware applications. The PROSOFT main objectives were presented, showing that the use of formal methods is on emphasis in this environment.

Some requirements that influenced the development of Cooperative PROSOFT were presented, as a natural extension to the Distributed PROSOFT. The framework presented in this paper provides a groupware development environment based on formal specification. Groupware ATOs can now be developed without compromising any PROSOFT feature. Our experience in adjusting existing ATOs shows that Cooperative PROSOFT is adequate for building effective Groupware ATOs to provide a higher level of cooperation between users. A consequence of this work is the integration of the advantages found in formal specification techniques to the groupware development. This combination can provide the development of higher quality groupware applications than those obtained with the use of traditional techniques.

The PROSOFT software process manager (described in [LIM 98]) provides coordination and control capabilities that can be used to guide the cooperative interaction between software engineers. However, PROSOFT support for asynchronous interaction is still in progress.

PROSOFT is an environment in constant evolution. Now the efforts are directed to build an interface between this environment and the Java programming language (as pointed at [SCH 97b]). A new wave of distributed Internet enabled ATOs are expected to appear.

6. REFERENCES

[BJØ 78] BJØRNER, D.; JONES, C.B. **The Vienna Development Method:** the meta-language. Berlin: Springer-Verlag, 1978. (Lecture Notes in Computer Science)

[GOL 96] GOLENDZINER, L.G.; SANTOS, C.S. **Versions and configurations in object-oriented database systems: a uniform treatment.** Prêmio Compaq de estímulo à pesquisa e desenvolvimento em informática, 1., São Paulo: Instituto Uniemp, 1996.

[LIM 98] LIMA, C.A.G. **Um Gerenciador de Processos de Software para o Ambiente PROSOFT**. Porto Alegre: CPGCC-UFRGS, 1998. M.Sc. Thesis. (in Portuguese)

[NUN 89] NUNES, D. J. **PROSOFT: Um ambiente de desenvolvimento de software estendível.** UFRGS-II, Brazil, 1989. (Internal publishing in Portuguese).

[NUN 92] NUNES, D. J.. Estratégia Data-Driven no Desenvolvimento de Software. BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, 6., Gramado. **Proceedings...** Porto Alegre: Instituto de Informática/UFRGS, 1992. p. 81-95. (in Portuguese)

[NUN 95] NUNES, D. J. **PROSOFT**. UFRGS-II, Brazil, 1995. (Internal publishing in Portuguese).

[PRE 93] PRESSMAN, R. **Software Engineering:** A practitioner's approach. Third Edition, McGraw-Hill, 1993.

[REI 97] REIS, R.Q.; LIMA, C..; NUNES, D. Cooperative Software Development and the PROSOFT Environment. In: INTERNATIONAL SYMPOSIUM ON COMPUTER AND INFORMATION SCIENCES, 12., 1997, Antalya, Turkey. **Proceedings....** Antalya: Bogaziçi University, Oct. 1997.

[REI 98] REIS, R.Q. **Uma Proposta de Suporte ao Desenvolvimento Cooperativo de Software no Ambiente PROSOFT.** Porto Alegre: CPGCC-UFRGS, 1998. M.Sc. Thesis (in Portuguese)

[REK 92] REKERS, J., SPRINKHUIKEN-KUYPER, I., Leiden Univeristy, 1992. **A LOTOS specification of a CSCW tool.** The Netherlands. Internet: ftp:// ftp.wi.LeidenUniv.nl/pub/CS/TechnicalReports/1992/tr92-28.ps.gz

[ROS 92] ROSENBERRY, W.; KENNEY, D.; FISHER, G. **Understanding DCE.** Open Software Foundation, O'Reilly & Associates, 1992.

[SCH 95] SCHLEBBE, H. **Distributed PROSOFT.** Department of Computer Science, Tecnical Report, University of Stuttgart, Germany, 1995.

[SCH97a] SCHLEBBE, H.; NUNES, D.J. **PROSOFT:** An Object-based Distributed Software Development Environment. Department of Computer Science, Technical Report, University of Stuttgart, Germany, 1997.

[SCH97b] SCHLEBBE, H.; SCHIMPF, S. **Reengineering of PROSOFT in Java.** Department of Computer Science,Technical Report, University of Stuttgart, Germany, 1997.

[TOP 95] TOPPER, A. Tools for group development. **Object Magazine**, London, v. 4, n. 8, Jan. 1995.

[VES 95] VESSEY, I.; SRAVANAPUDI, A.. CASE Tools as Collaborative Support Technologies. **Communications of the ACM**. New York, v. 38, n. 1, p. 83-95 Jan. 1995.

[WAT 91] WATT, D. **Programming Language Syntax and Semantics.** Prentice-Hall, 1991.