# A Model for Security in Agent-based Workflows

*Henrik Stormer, Konstantin Knorr, Jan H.P. Eloff*

*With the rise of global networks like the Internet the importance of workflow environments is growing. However, security questions in such environments often only address secure communication. Other important topics are role-based access control and separation of duty. This paper shows how mobile agents can be used to implement these and other security features in workflows.*

*Keywords:* agent, fraud, role, security, separation of duties, workflow

## 1 Introduction

Workflow environments have hugely benefited from the technical advancements made available by the Internet over the last years. Many workflow environments today are implemented over public networks such as the Internet. Because workflow environments in most situations represent the "bread-and-butter" of a company, the implementation thereof has raised serious information security problems. Organizations are concerned about their privacy on the net as well as of the privacy of client information. Similar to other systems the information security requirements of a workflow system are modelled on the ISO 7498-2 standard. This standard proposes the following information security services: identification and authentication, authorization (access control), confidentiality, integrity, and non-repudiation. Mechanisms for each of these services must be employed to secure a workflow environment.

Identification and authentication, confidentiality and non-repudiation services are implemented similarly to those in non-workflow environments. The services of authorization (with the main focus on access control) and integrity require special design considerations and implementation details. For example access control requires the modelling of access based on the type of tasks to be performed on the objects travelling around in a workflow environment. A unique feature of integrity in a workflow environment is to preserve the contents of objects according to business rules. These business rules are linked to the operational characteristics of an organization. There is a need for new approaches modelling the design and implementation of the access control and integrity services in workflow environments.

Currently available research results in the area of access control are dominated by models of role-based access control (RBAC). RBAC shows good potential to be successfully employed in a workflow system. The information security principle of separation of duties (SoD) is important in the modelling of integrity in a workflow environment. A physical and logical separation of tasks can improve the prevention of fraudulent activities.

Agent technology shows great potential in the field of workflow systems. Furthermore, information security aspects like RBAC and SoD can be considered in the agent-based implementation of workflows.

Therefore, the primary aim of this paper is to give an architectural model and a framework for implementing access control and integrity in a workflow environment. Intelligent and mobile agents are applied to current workflow technology to meet access control and integrity requirements.

The remainder of the paper has the following structure: Section 2 gives an introduction to workflow environments and agents. Section 3 describes a sample process which will be used for illustration purposes throughout the paper. An architecture for agent-based workflows and its different agent types are discussed in Section 4. Section 5 introduces the notions of RBAC and SoD and gives a formal model for them within the workflow environment. Implementation of several security features in an agent-based workflow system is the topic of Section 6. Section 7 gives a conclusion.

***Henrik Stormer*** has studied computer science at the University of Saarbrücken (Germany). He is currently a PhD student at the Department of Information Technology at the University of Zurich. His research interests are mobile agents and agent-based workflow systems. stormer@ifi.unizh.ch

***Konstantin Knorr*** studied mathematics at the universities of Mainz and Frankfurt (Germany) and is currently a PhD student at the Department of Information Technology, University of Zurich. His research interest are formal models of security and security in workflow environments. knorr@ifi.unizh.ch

***Jan Eloff*** is a professor in computer science at Rand Afrikaans University, South Africa. He is currently a visiting professor at the University of Zurich, Switzerland. He is chairperson of the Special Interest Group in Information Security and is chairperson of the International Working Group 11.2 of IFIP specialising in small systems security. He is an evaluated researcher from the National Science Foundation, South Africa. eloff@rkw.rau.ac.za

## 2 Background

This section gives background information on workflow management issues and mobile agents.

Workflow management is an essential research area in computer science. A workflow is an executable business process whose modelling and execution is supported by a software system called workflow management system (WfMS) [Georgakopoulos et al. 95]. Before a workflow can be executed, it has to be described in a way the WfMS is able to understand. This description is called a *process definition*. The definition has to be made during *build time* before a workflow can be executed. During run time of the system many instances of the workflow are generated according to the process definition. The main elements of a process definition are tasks, objects, subjects, roles, and the control flow. A process consists of several tasks whose chronological and logical order is given through the control flow. To describe a task, it has to be specified which roles are granted access to which objects. Subjects can be associated with persons but also with machines and computer programs. [Leymann/Altenhuber 94].

The *Object Management Group* defines a software agent as "a computer program that acts autonomously on behalf of a person or organization" [Crystaliz et al. 97]. The following properties characterize agents:

- pro-active (support of the user's work)
- adaptive (learning the user's preferences or the ability to work on different platforms)
- autonomous (limited communication with its creator)
- intelligent (making 'intelligent' decisions [Ferber 99])
- mobile (can actively migrate in networks to different systems and move directly to the local resources, like databases or application servers)

Before agents can be used, each system needs to install a so called *agent-place* to create, delete and execute agents. Agents can migrate from an agent place to another performing the work locally. Lange and Oshima [Lange/Oshima 99] give reasons why to use agents: e.g. reduction of the network load, overcoming of network latency and encapsulation of protocols.

An agent-based workflow is a workflow in which agents perform, coordinate, and support the workflow [Huhns/Singh 98]. In an agent-based workflow system, there exist different agent types that manage the workflow. A process instance agent is responsible for controlling one process instance. Newer architectures further split the functionality of a workflow system:

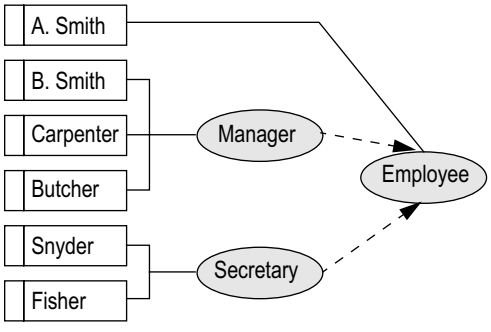


**Fig. 1:** Sample role definition and hierarchy

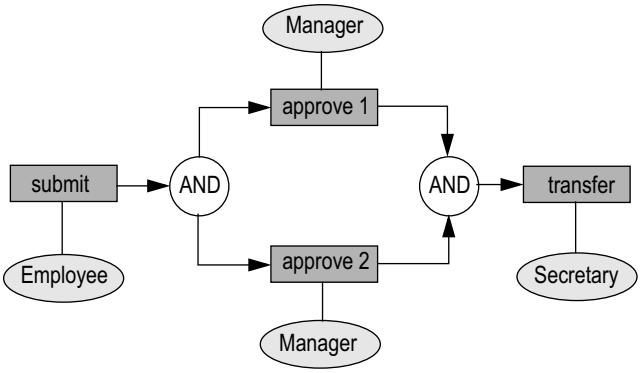


**Fig. 2:** Example of a process definition

task agents, which are mobile and migrate to subjects which perform some tasks of the workflow [Hawryszkiewycz/Debenham 98], and personal agents which act as an interface between the subject and other agents are introduced.

## 3 Example Workflow Environment

This section gives an example which will be used throughout the paper for illustration purposes.

Six persons, also referred to as subjects, *A. Smith*, *B. Smith*, who are brothers, *Carpenter*, *Butcher*, *Snyder*, and *Fisher* are working in a company. Figure 1 shows these subjects together with their assigned roles. Note that every *Manager* (or *Secretary*) is an *Employee*, too. E.g. *Butcher* is able to activate the manager or the employee role. The partial order of roles builds up a so called *role hierarchy* – a well known modelling approach [Scheer 94].

Figure 2 shows the process definition for a travel expense claim. The process starts when an employee submits a travel expense claim. Two Managers must approve this claim before the money transfer is done by a secretary. Note that the tasks are partially ordered, e.g. *submit* precedes all other tasks but no order is possible between *approve 1* and *approve 2*.

## 4 Architecture of the Agent-based Workflow System

The architecture of the proposed agent-based workflow system consists out of the following four agent types:

*Process Instance Agent (PIA)* The PIA is created by a subject, which has to provide a complete and correct process definition. The process instance agent represents and manages an instance of a workflow (according to the process definition given) and controls its whole execution. In the example a PIA is created for each travel expense claim of an employee (e.g. Claim 157 of Butcher).

*Task Instance Agent (TIA)* The TIA is responsible for one task in a process instance. It is created by the PIA and has to search for a subject, deliver the task description and objects items to the subject and the results back to the PIA. Referring to the example, a TIA is created for the transfer task of Claim 157 of Butcher.

*Worklist Agent (WLA)* The WLA stores a mapping of all subjects and their assigned roles (cf. Figure 1).

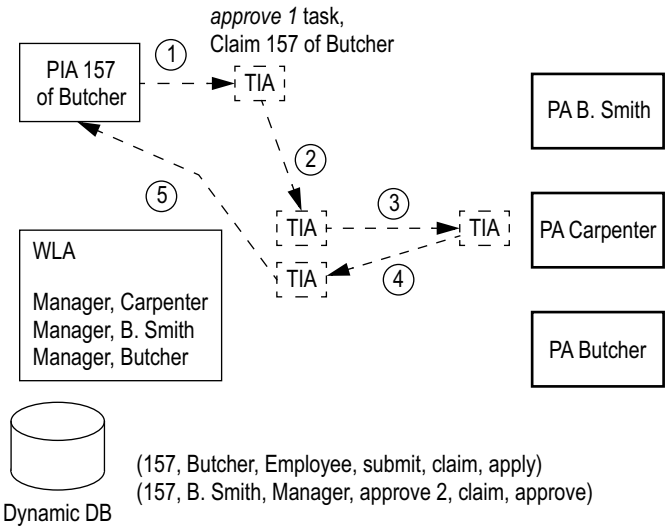


**Fig. 3:** Life cycle of a TIA

*Personal Agent (PA)* A personal agent is the interface between the subject and the incoming TIAs. It is immobile, controls the incoming task requests from the TIAs, and coordinates the communication between subject and TIA.

The basic idea in the handling of a workflow task is to create a TIA for each task in the underlying process instance. The PIA creates all TIAs according to the temporal and logical prerequisites of the workflow. E.g., the parallel execution of several tasks can be achieved by the creation of several TIAs simultaneously.

When the PIA creates a TIA for fulfilling task *t*, all and just those objects and privileges which are needed for the task's execution are instantiated as part of the TIA. Next, the TIA has to find a subject for task *t*. Therefore, it migrates to the WLA to get a list of possible subjects to interact with *t*, qualifying by means of role allocations and possibly other constraints. The TIA chooses a subject *s* randomly from the list and migrates to *s*: This is another security feature because the choice of the TIA is not predictable and therefore fraud is complicated. Clark and Wilson [Clark/Wilson 87] identified SoD as one of the two major mechanisms that can be implemented to ensure data integrity. They purposed the random selection of task participants in order to ensure that any attempted conspiracy is inherently unsafe. More elaborate choices are possible but are not further investigated as part of this paper.

After the migration of the TIA to the selected subject, the PA of *s* is informed that a new task is waiting to be executed. The PA must now inform the subject, for example by showing a message on the screen or playing a sound sample. Then, the subject performs the task using the objects and privileges which will be provided by the TIA. When the task is done, the TIA migrates back to the WLA to pass information to the dynamic database of WLA. This information will contain *s*, *t*, the objects and privileges used plus other information like time stamps. Finally, the TIA migrates back to the PIA to pass control flow related information.

Fig. 3 illustrates the above procedure by means of the travel expense claim example. The PIA instantiates the task *approve1* in the process instance 157 of Butcher (1). The TIA migrates to the WLA to get a list of potential subjects who qualify for the execution of the task (2). The WLA creates the subject list based on rules and on the dynamic database (cf. Section 5.3). In this example the list contains only one subject, Carpenter, because the dynamic database shows that B. Smith has done the other approve task and Butcher cannot approve his own claim. Therefore, the TIA migrates to PA of Carpenter and the task is executed. Next, the TIA migrates back to the WLA to pass information to the dynamic database. Finally, the TIA migrates back to the PIA.

## 5 Security Aspects

The focus of the paper is to show that SoD and RBAC mechanisms are feasible for implementation in an agent-based workflow system. Therefore, this section gives a brief introduction to RBAC and SoD. Then, a formal model is proposed facilitating the implementation of access control (by means of RBAC) and integrity (by means of SoD) in an agent-based workflow environment.

### 5.1 RBAC

In a workflow environment, usually tasks are not linked directly to subjects. The concept of *roles* forms a middle layer between the subjects and the tasks, cf. Figure 4. As an example consider the *Manager* role in the travel claim example. Furthermore, access rights are enforced on roles and not on subjects. This simplifies the security administration. There is a considerable amount of research about RBAC going on.[1]

### 5.2 SoD

"SoD is a policy to ensure that failures of omission or commission within an organization are caused only by collusion among individuals and, therefore, are riskier and less likely, and that chances of collusion are minimized by assigning individuals of different skills or divergent interests to separate tasks" [Gligor et al. 98]. In a workflow context, SoD has to be divided and extended into static SoD (SSoD) and dynamic SoD (DSoD). SSoD enforces certain rules during build time of the workflow, i.e. before process instances of the workflow are instantiated. Example: The process definition in Figure 2 requires that different tasks are performed by different roles (the *transfer* task by a secretary and the *approve 1* task by a manager). SSoD rules are applied to a process definition to guarantee their enforcement. In contrast DSoD can only be enforced during run time of the workflow, i.e. during the execution of a single instances of a process. DSoD can be enforced on different layers. Consider the following examples from the scenario introduced in Section 3:

1. A manager should not be allowed to submit for (first task) and approve (in a later task) his own travel expense claim. In this case DSoD should be enforced on the role and subject layer.

1. http://www.acm.org/sigsac/rbac2000.html

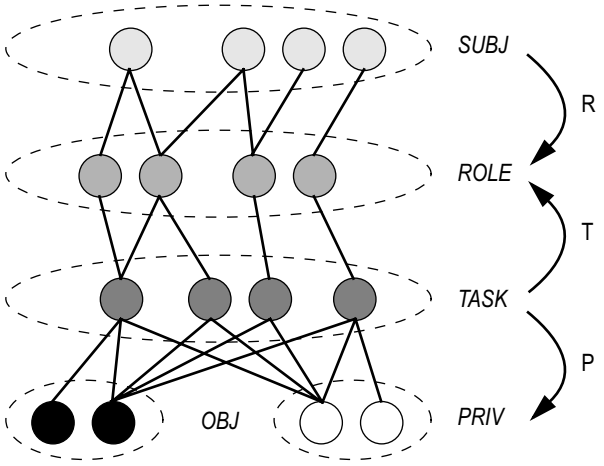


**Fig. 4:** Sets and mappings in the DSoD model

2. The following example is on a subject layer: Two brothers (A. Smith and B. Smith) are working in the same company. To prevent fraud, one of the brothers should not be allowed to approve the travel expense claim of his brother.
3. As a third example considers a document which is used in a first task by a subject in a certain role. In the travel claim example there could be the policy that the subject should not get any privileges to this document for the remaining tasks in the process instance even in a different role. This could be necessary to prevent any (ex post) manipulation of the document. Suppose Snyder submits a travel expense claim in her employee role. Then, Snyder can not transfer the money even if her secretary role is activated. This example illustrates SoD on an object, subject and role layer.

There are more complex SoD policies ([Ahn/Sandhu 99, Bertino et al. 99]) which fall out of the scope of this paper. The model which will be introduced in the following subsection allows for defining dynamic policies on role, subject, task, object and privilege layer. To illustrate the model, the three examples given above will be used.

### 5.3 A model for DSoD

Our model uses set theory and functions using the following five sets:

*SUBJ* The set of all persons who are capable of executing a task.

*ROLE* The set of all roles in a process definition.

TASK This set includes all tasks that are defined in the process definition.

*OBJ* The set of all objects (e.g. a text document) which are needed to execute the tasks.

*PRIV* The set of privileges. A privilege is used on an object, for example a read permission on a text document.

In the workflow definition it has to be defined which subjects are allowed to activate which roles. Therefore, the function $R$ maps *SUBJ* to the power set of *ROLE*.

$$R:SUBJ \rightarrow 2^{ROLE}$$

In a next step, all task definitions have to include the role that is allowed to execute the task. The following function is defined:

$$T:TASK \rightarrow 2^{ROLE}$$

Finally, a function is needed to associate tasks with objects and privileges:

$$P:TASK \rightarrow 2^{OBJ} \times 2^{PRIV}$$

Figure 4 shows the relationship between the five sets and the three functions. Information constrains in the above mentioned sets must be defined in the process definition. At run time, it is possible to implement DSoD using these sets together with a natural number, indicating the process instance.

Consider a tuple consisting of the Cartesian product of the following six sets where N is the set of all natural numbers and n indicates the process number:

$$(n, s, r, t, o, p) \in N \times SUBJ \times ROLE \times TASK \times OBJ \times PRIV$$

Not all of these tuples are meaningful. Therefore, the notion of soundness is introduced. A tuple $(n, r, s, t, o, p)$ is *sound* if and only if

1. $r \in R(s)$ ,
2. $r \in T(t)$ , and
3. $(o, p) \in P(t)$

hold. The first inclusion says that the subjects $s$ should be allowed to activate role $r$. Next, the role $r$ should be allowed to execute task $t$. Finally, the object $o$ and the privilege $p$ should match with the task $t$. Note that the tasks have to be in a chronological and logical order. We therefore assume that (TASK, ≤) is a partially ordered set. $t_1 \leq t_2$ says that $t_1$ is executed in parallel with or before $t_2$.

The WLA uses a database called $DB_{dyn}$ for storing all tuples and rules to enforce DSoD. The rules for our examples above are:

1. A manager who is not allowed to approve his own claim. $n$ is a process instance id, $s$ a subject, $r$ the manager role, $t_1$ is the submit task, $t_2$ the approve 1 or 2 task, $o$ is the travel claim and $p$ are the submit privilege.
   $(n, s, r, t_1, o, p) \in DB_{dyn} \Rightarrow (n, s, r, t_2, \#, \#) \notin DB_{dyn}$
   # symbolizes any possible element of the corresponding set.

2. Consider the "brother" example, where $s_1$ is A. Smith and $s_2$ is B. Smith, $t_1$ is submit, $t_2$ is approve 1 or 2 task, $o$ is the travel claim document, $p_1$ is the submit and $p_2$ the approve privilege.
   $(n, s_1, \#, t_1, o, p_1) \in DB_{dyn} \Rightarrow (n, s_2, \#, t_2, o, p_2) \notin DB_{dyn}$

3. Last example: $s$ is Snyder, $r$ is Employee, $t_1$ is submit and $o$ is the travel claim.
   $(n, s, r, t_1, o, \#) \in DB_{dyn} \Rightarrow (n, s, \#, t_2, o, \#) \notin DB_{dyn}$
   for all $t_1 < t_2$

It is very important to check the rules for consistency. Contradicting rules may ruin the whole SoD mechanism. Also it might be possible to combine several rules to a single rule.

This process is called *pruning*. For more details see [Bertino et al. 99].

## 6 Implementation of DSoD in an Agent-based Workflow System

Section 4 introduced an architecture for an agent-based workflow system, Section 5.3 a formal model for DSoD rules. This section shows how these parts are interrelated. For illustration refer to Figure 3 to see which information from the DSoD model is used when and by which agent types.

The PIA has control over the whole process execution, i.e. the partially ordered set of tasks *TASK*, the roles *ROLE*, objects *OBJ*, and privileges *PRIV* associated with each task. In step (1) the PIA instantiates a TIA for task $t$ and passes information about the task. This information consists of $P(t)$ (the privileges, objects) and $T(t)$ (the roles) associated with $t$. Note that only the privileges and objects are passed which will be needed by the subject to perform the task implementing the security principle of least privilege.

The TIA migrates to the WLA and passes $T(t)$ to the WLA. The list of possible subjects is created by checking the elements of $T(t)$ and $R(s)$ for all tasks and subjects.

In step (3) the TIA decides randomly which subject performs the task and gives this information back to the WLA. The WLA will generate a subject list $[s_1,..., s_n]$ after enforcing the DSoD rules based on the entries in the dynamic database $DB_{dyn}$. This list consists of all subjects capable of executing the task and will usually contain more than one subject. The TIA uses a random function rand which maps the input $n$ to a random number in the set $\{1, \ldots, n\}$. The subject chosen will be $s_{rand(n)}$. This practice further decreases the possibility of fraudulent activities since the outcome of the assignment is not predictable.

Now the WLA blocks the subject for this instance. This is necessary for parallel execution, otherwise the $DB_{dyn}$ checking could fail. Before the TIA migrates to the subject the WLA inserts new tuples in the dynamic database. A tuple is $(n, r, s, t, o, p)$, where $n$ is the process instance number, $r$ the role associated with task $t$, $s$ the subject which performs $t$, and $p$ the privilege used on object $o$. Several tuples can be inserted into $DB_{dyn}$ for $t$, e.g. if several objects are used in the task. When the TIA arrives at the subject, the PA of $s$ gets the objects plus privileges contained in $P(t)$. Depending on the task, one or more objects may be created which have to be transferred via the TIA to the PIA. (4) takes the TIA back to the WLA where the subject is unblocked.

As a final step (5) the TIA moves back to the PIA where the objects are passed over. Finally, the TIA is deleted by the PIA. If a complete process instance has finished (in the example, after the transfer task), the PIA can initiate a garbage collection at the WLA which removes all tuples from this process instance from $DB_{dyn}$. This information together with the objects created during the execution can be stored in an archive. Finally, the PIA is deleted, too.

## 7 Conclusion

This paper introduced an agent-based workflow environment consisting of four different agent types. The following table shows the four agent types and their most important characteristics.

| | Mobility | Intelligence | Security | Instance |
|---|---|---|---|---|
| PIA | - | + | + | + |
| TIA | + | + | + | ++ |
| WLA | - | - | - | - |
| PA | - | - | - | - |

The mobility of a TIA is rated high (+), since a TIA has to do several migrations during its life. The TIA will be an "intelligent" agent because several decisions such as the choosing of a subject from the subject list has to be done. The TIA furthermore enforces the strict least privilege principle (Security +). A TIA has to be created for every task in a process instance (marked ++ in the Instance column). The other agents are interpreted similarly.

This workflow environment was used to enforce different security features:

1. (Dynamic) access control: The access to objects is restricted. This is done dynamically because TIAs are created based on the state of the process instances.
2. Strict least privilege: A subject will just receive privileges to objects which are needed for the execution of a task.
3. SoD: The major focus of the proposed architecture is SoD. Its dynamic variant DSoD is realized through the interaction of PIA and WLA.
4. Random choice of subjects: This practice further decreases the possibility of fraudulent actions.

The work suggests that agents are a valuable resource in implementing security features in workflow environments.

Future work will deal with the development of a prototype to validate the presented architecture. Furthermore, the SoD model will be extended, e.g. to enforce rules between different workflow instances, e.g. when confidential data could be collected in different workflow instances by the same subject.

**References**

[Ahn/Sandhu 99]
Gail-Joon Ahn and Ravi Sandhu. The RSL99 Language for Role-Based Separation of Duty Constraints. In Proceedings of 99 ACM Conference on Role Based Access Control, 1999.

[Bertino et al. 99]
Elisa Bertino, Elena Ferrari, and Vijayalakshmi Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Transactions on Information and System Security, 2(1):65–104, February 1999.

[Crystaliz et al. 97]
Crystaliz, Inc., General Magic, Inc., GMD Focus, and IBM Coorp. Mobile Agent Facility Specification. Technical report, OMG, 1997.

[Clark/Wilson 87]
David D. Clard and David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. IEEE Symposium on Security and Privacy, pages 184–194, 1987.

[Ferber 99]
Jacques Ferber. Multi-Agent Systems: An Introduction to Artificial Intelligence. Addison Wesley Publishing Company, 1999.

[Gligor et al. 98]
Virgil D. Gligor, Serban I. Gavilla, and David Ferraiolo. On the

formal definition of separation-of-duty policies and their composition. IEEE Computer Society, IX:172–183, 1998.

[Georgakopoulos et al. 95]
Dimitrios Georgakopoulos, Mark Hornick, and Amith Sheth. An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure. Distributed and Parallel Databases, 3:119–153, 1995.

[Hawryszkiewycz/Debenham 98]
Igor Hawryszkiewycz and John Debenham. A Workflow System Based on Agents. In Gerald Quirchmayr, Erich Schweighofer, and Tervor J. M. Bench-Capon, editors, Database and expert systems applications, volume 17, page 688ff. Springer, 1998.

[Huhns/Singh 98]
Michael N. Huhns and Munindar P. Singh. Workflow Agents. IEEE Internet Computing, July/August 1998.

[Leymann/Altenhuber 94]
F. Leymann and W. Altenhuber. Managing Business Processes as an Information Resource. IBM Systems Journal, 33(2):326–348, 1994.

[Lange/Oshima 99]
Danny B. Lange and Mitsuru Oshima. Seven Good Reasons for Mobile Agents. Communications of the ACM, 42(3):88, March 1999.

[Scheer 94]
August-Wilhelm Scheer. Business Process Engineering. Reference Models for Industrial Enterprises. Springer Verlag, Berlin, 1994.