WISE: Process based E-Commerce

Amaia Lazcano Heiko Schuldt Gustavo Alonso Hans Schek
Institute of Information Systems
Swiss Federal Institute of Technology (ETHZ)
ETH Zentrum, CH-8092 Zürich, Switzerland
{alonso, lazcano, schek, schuldt}@inf.ethz.ch

1 Introduction

Electronic commerce is a business practice that is experiencing an extraordinary growth. Unfortunately, there is a severe lack of adequate software tools. The WISE project (Workflow based Internet SErvices) at ETH Zürich is an attempt to address this problem by providing a software platform for process based business to business electronic commerce. The final objective of the project is to develop a coherent solution for enterprise networks that can be easily and seamlessly deployed in small and medium enterprises. As a first step in this direction, we have developed a simple but powerful model for electronic commerce to be used as the overall design principle [LASS00]. To support this model, we have extended OPERA [Hag99, AHST97], a process support kernel built at ETH that provides basic workflow engine functionality and a number of programming language extensions, with the capability to implement trading communities that interact using virtual business processes. In what follows we explain the model in detail and outline the architecture of the system along with one concrete application.

2 E-Commerce Model

Business processes are used to model the most relevant activities within an organization. They can be seen as a set of procedures and rules, in graphical or textual form, describing the steps that must be taken in order to accomplish a given business goal. In practice, business processes are used to both document everyday procedures and as the basis for automating and optimizing such procedures. From here, we can provide a more concrete definition of electronic commerce by linking the business objects with the technology used to implement them. Thus, we define business to business electronic commerce as *the incorporation of information and communication technology into the business process so as to expand it beyond the corporate boundaries*. The first step towards making this definition a practical reality is to specify how to go beyond the corporate boundaries. This specification is based on the notions of virtual business processes, virtual enterprises, and trading communities.

A *virtual business process* is used to define concrete business goals and describe the corresponding activities. Unlike normal processes, in a virtual business process the definition and enactment is not tied to a single organizational entity. In a way, the virtual business process can be seen as a meta-process: its building blocks are the subprocesses provided by the participating companies. A virtual business process cannot be defined without

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

a context, i.e., without a set of goals, rules, requirements, constraints, and resources. This context is what we termed the *virtual enterprise*. Alternatively, a virtual enterprise can be seen as an organization based on virtual business processes. The concept of virtual enterprise is not gratuitous. Everything that cannot be resolved at the level of the component processes must be resolved at the virtual enterprise level, that is, within the context of the virtual process. Naming this context explicitly allows us to have a much better perspective of the tools to develop and how they should interact between them. For instance, it allows to specify what to do in case of exceptions at the virtual process level. To define the actors in the scenario, we use the notion of *trading community* which can be best described as the set of companies participating in a virtual enterprise. Alternatively, a trading community could be defined as the set of companies which provide the building blocks of the virtual business process. These two definitions are roughly equivalent: we consider a 1:1:n mapping between the trading community, the virtual enterprise and the virtual business process. That is, each virtual enterprise has one trading community and can run a number of virtual business processes. From a practical standpoint, defining the trading community is the first step towards defining access rights, responsibilities, authentication and encryption mechanisms, and the configuration of the underlying distributed system.

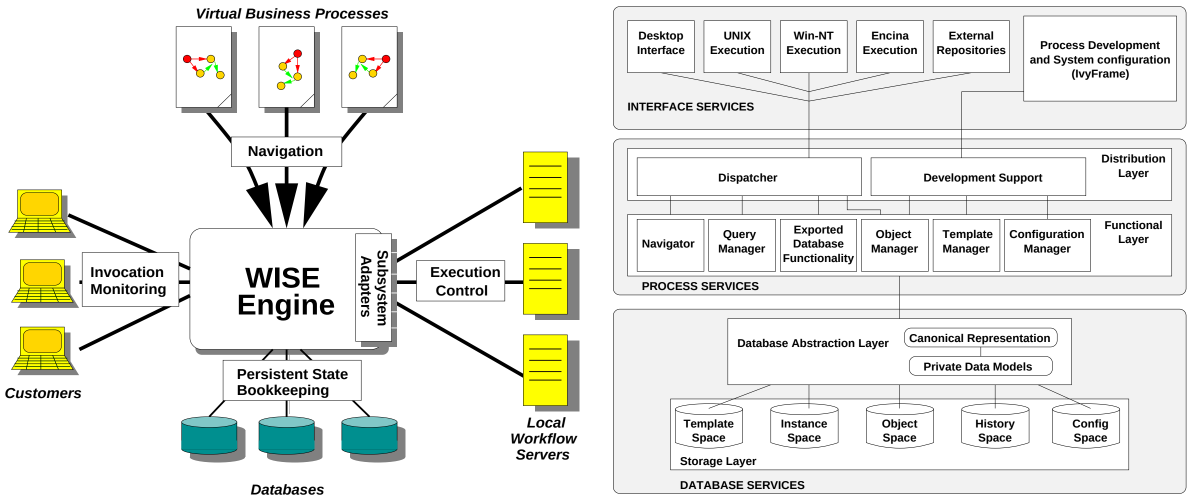
3 WISE Architecture

The challenge in electronic commerce is how to build a software tool capable of supporting the entire life cycle of a virtual business process. By this we mean that virtual business processes should be seen as valuable assets to be maintained for as long as they are in use. Support for this life cycle can only be provided through a generic framework which can be used to develop virtual business processes without a significant amount of expertise or development cost. This framework should provide technical solutions to problems such as how to incorporate the services of different companies as part of a single business process, how companies can advertise their services and make them available to other companies, or how a virtual business process can be enacted and its execution monitored.

To implement this model, WISE [AFH+99] builds upon OPERA [Hag99, AHST97]. OPERA can be seen as the kernel of a workflow management engine capable of executing business processes. OPERA was extended to support trading communities that interact based on *virtual business processes*. This support consisted in making WISE act as a coarse granularity programming language and its development environment plus a high level distributed operating system designed to work over a heterogeneous, geographically separated cluster of computers and conceptually based on the notion of process. Instead of the traditional system level calls, our building blocks are already existing applications. Instead of conventional programs, we work with processes.

WISE has both a runtime component and a development environment associated to it. WISE is organized around three service layers (Figure 1.b): *database services*, *process services* and *interface services*.

The database service layer acts as the storage manager. It encompasses the storage layer (the actual databases used as repositories) and the database abstraction layer (which makes the rest of the system database independent). The storage layer is divided into five *spaces*: template, instance, object, history, and configuration, each of them dedicated to a different type of system data. Templates contain the structure of the processes. When a process is to be executed, a copy of the corresponding template is made and placed in the instance space. This copy is used to record the process' state as execution proceeds. Storing instances persistently guarantees forward recoverability, i.e., execution can be resumed as soon as the failure is repaired, which solves the problem of dealing with failures of long lived processes [ST96, DHL91]. Instances also constitute the basic unit for operations related to process migration and backup facilities [HA99b] (changes are asynchronously sent to two databases so that one acts as a backup for the other). The history space is used to store information about already executed instances (as in, for instance, [GMWW99]). It contains a detailed record of all the events that have taken place during the execution of processes, including already terminated processes. Finally, the configuration space is used to record system related information such as configuration, access permissions, registered



a) b)

Figure 1: a) General architecture of WISE, b) Internal organization of the engine

users, internet addresses, program locations, and so forth. The database abstraction layer implements the mechanisms necessary to make the system database independent. It translates the process representation to the private representations of the underlying repositories (SQL, C++, system calls) as required by the underlying spaces.

The process service layer contains all the components required for coordinating and monitoring the execution of processes. The most relevant components for the purposes of this paper are the *dispatcher* and *navigator* modules. The dispatcher deals with physical distribution and acts as resource allocator for process execution. It determines in which node the next step will execute, locates suitable nodes, checks the site's availability, performs load balancing, and manages the communication with remote system components. The navigator acts as the overall scheduler: it "navigates" through the process description stored in the main memory, establishing what to execute next, what needs to be delayed, and so forth. Once the navigator decides which step(s) to execute, the information is passed to the *dispatcher* which, in turn, schedules the task and associates it with a processing node in the cluster and a particular application. The dispatcher then contacts the *program execution client* (PEC); this is a small software component present at each node responsible for running application programs on behalf of the WISE server. Users interact with the system via *desktop interfaces*, which are also used to inform the user of any activity that they need to execute as part of a process (similar to worklists in workflow engines).

As an example of added-value process services, WISE allows to provide higher level database functionality [SBG$^+$00] for processes (*exported database functionality*) following the ideas of transactional process management [SAS99], i.e., guaranteeing correct concurrent and fault-tolerant process execution.

The development environment allows users to specify processes via a process definition tool. The tool we use is *Structware/IvyFrame* [Ivy98, Lie98], which is internally based on Petri-nets and supports not only the modeling of business processes but also sophisticated analysis of its behavior (bottlenecks, average execution times, costs, delays, what if analysis, etc.). Using the process definition tool, it is possible to perform *process creation* and *configuration management*. The configuration management allows users to specify the hardware and software characteristics of the computer infrastructure to be used in the execution of a process (IP addresses, type of OS, CPU specifications, etc.). The process creation element allows users to create processes by combining individual activities and subprocesses, and specifying the flow of control and data among them.

The graphical representation produced by the process definition tool is compiled into a language called *Opera Canonical Representation* (OCR) [Hag99], that is used internally by WISE to create process templates. OCR supports advanced programming constructs such as e.g., event handling [HA99a], exception handling [HA00], spheres of atomicity [HA98], and high availability [HA99b].

4 Applications of WISE: Payment protocols

The correct and reliable accomplishment of payments is a crucial feature of e-commerce interactions, both in the business to business (b2b) and in the business to customer (b2c) case. In general, payments in e-commerce have a well-defined structure [MWW98] and, due to the transfer of sensitive information, come along with a couple of correctness requirements:

Atomicity Since payments take place in a highly distributed and heterogeneous environment, various aspects of atomicity can be distinguished: *money atomicity* which addresses the transfer of money between customer and merchant [CHTY96, Tyg98], *goods atomicity* which accounts for the atomic and correct delivery of the merchandise [CHTY96]), and finally *distributed purchase atomicity* [SPS99] which considers the atomic combination of different, originally independent interactions of a customer with several merchants.

Provability and Verification All participants must be able to *prove* –after a payment is settled– that the goods sent (received) are those they agreed upon in the initial negotiation phase. This requirement stems from the fact that fraudulent behavior has to be considered. Moreover, all participants have to be supported by appropriate mechanisms to *verify* the properties of a payment protocol w.r.t. atomicity and provability.

Concurrency Control When payments are processed concurrently, there has to be support for concurrency control, for instance to prevent that identical electronic cash tokens are multiply used (double spending).

In what follows, we show how dedicated e-services supporting payments with all these execution guarantees can be seamlessly provided by exploiting the special capabilities of the WISE system, i.e., by embedding all interactions into appropriate processes and by enforcing their correct concurrent and fault-tolerant execution. To this end, we make intensive use of the execution guarantees WISE provides for processes, following the theory of transactional process management [SAS99]. In particular, we take into account that certain steps of a process cannot be compensated (or are too expensive to be compensated) once they have been executed successfully while, at the same time, it is not possible to defer them until the end of a process due to control flow and/or data flow dependencies. Rather, we take into account that alternative process steps might exist which are guaranteed to succeed (i.e., which are retriable according to the terminology of the flex transaction model [ZNBB94]). The property of a process to terminate in a well-defined state even in the case of failures and even in the presence of non-compensatable steps by the existence of appropriate alternatives is called *guaranteed termination*. Essentially, this is a generalization of the traditional all-or-nothing semantics of atomicity in that exactly one out of several alternatives terminates correctly. Finally, by considering processes as transactions at a higher semantical level, transactional process management also considers correct concurrent process executions by applying ideas of the unified theory of concurrency control and recovery [AVA$^+$94].

In general, e-commerce transactions differ in terms of the (number of) participants, the type of goods, and/or the payment mode. All this information is subject to the negotiation which precedes the actual payment. Hence, a predefined process template is not appropriate to account for the particularities of individual payments. Rather, a generic process template has to be provided and to be automatically configured based on the results of the initial negotiation such that the resulting template reflects the payment of a concrete e-commerce transaction. This generic payment process template follows the ideas of anonymous atomic transactions [CHTY96] but extends and generalizes the latter to the case of multiple participants and supports both trading of digital goods [SPS00] and non-digital goods (via electronic contracts) as well as different means of payment.

The architecture of the payment coordinator (highlighted in dark gray) which is based on the WISE system and which controls the execution of process-based payment e-services, as well as the structure and the different steps of a payment process are depicted in Figure 2. In general, e-commerce transactions consist of three phases. The first phase encompasses bilateral negotiations between customer and merchants (1) and the transfer of encrypted goods to the customer (2). Due to this encryption, the merchandise cannot be used until the key transfer –which is an integral part of the payment process– is effected successfully. The second phase consists of the customer's combination of several independent interactions with different merchants into one

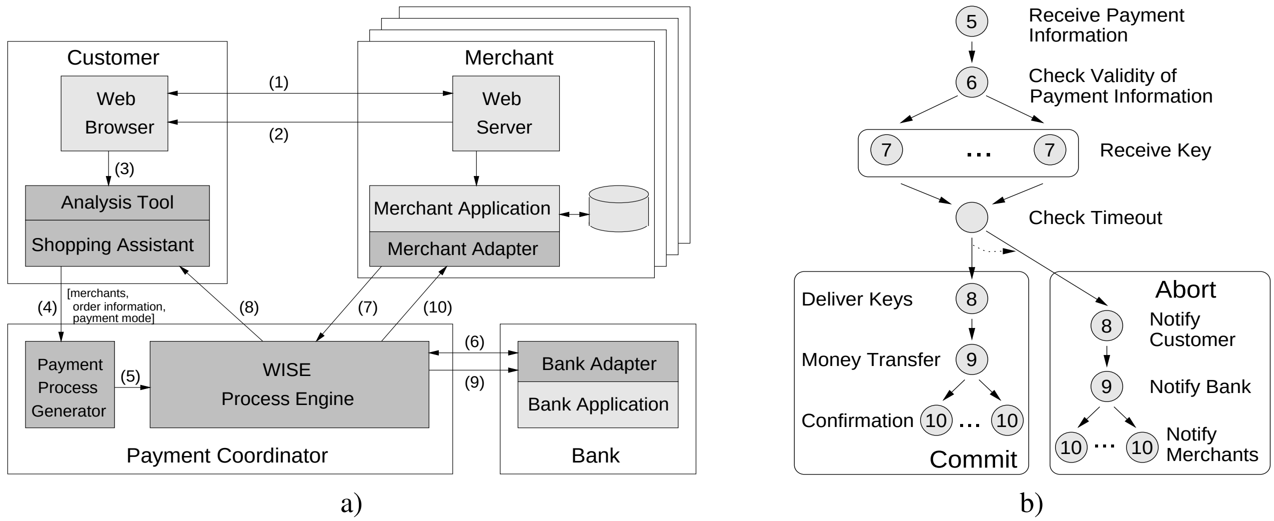


Figure 2: a) Architecture of the Payment Coordinator based on WISE, b) Structure of Payment Process

single e-commerce transaction, the analysis of the properties of the latter (3), and the generation of a concrete payment process reflecting this particular transaction (4). The third phase, steps (5)-(10), finally considers the execution of this payment process in which the cryptographic keys are collected from all merchants and delivered atomically together with the initiation of real money flow (commit branch) [SPS00].

In order to analyze whether these payment processes account for the previously identified execution guarantees, the process structure as well as the semantics of each step have to be considered. Money atomicity, goods atomicity, and distributed purchase atomicity are jointly present since all possible executions can be considered as correct: In the commit branch, money and goods are transferred for all purchases. This can be enforced since these steps are retriable (i.e., all keys have been received at that time and –after positive validation of the payment information– the money transfer is guaranteed by the bank). When some failure occurs (e.g., missing keys or exceeded deadlines), the abort branch guarantees that no information is transferred between customer and merchants (only appropriate notifications are sent). Verification is present in that the structure of the process template is known beforehand. Thus, all participants are aware of how payments will be processed and can check the guaranteed termination property of the generic process template, prior to the invocation of a concrete payment process. However, a crucial requirement is that the payment coordinator providing these payment e-services is regarded as a trusted instance by all participants. Hence, it has to be located at the site of a trustworthy and reliable instance (e.g., a certification authority or a clearing house).

While atomicity is inherent to each process, the properties of provability and concurrency control are tightly coupled to the architecture and the functionality of the WISE system. In terms of provability, the persistent bookkeeping is exploited to keep track of the individual behavior of participants. Finally, concurrency control for processes is seamlessly provided as a feature of the exported database functionality of WISE.

5 Conclusions

In this short paper, we have presented an electronic commerce platform that supports the implementation of intra- and inter-enterprise business processes. The WISE system does not only address the technological problems associated with e-commerce activities, but can also cover the business needs of different participants involved in a trading community by supporting several electronic services. We have presented the automatic generation of customized payment processes as an example of dedicated electronic services supported by the WISE system.

Acknowledgments

This work has been supported by the Swiss National Science Foundation as part of the projects WISE (Workflow based Internet Services) and INVENT (Infrastructures for Virtual Enterprises).

References

[AFH+99] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, and N. Weiler. WISE: Business to Business E-Commerce. In *Proceedings of the IEEE 9th International Workshop on Research Issues on Data Engineering. INFORMATION TECHNOLOGY FOR VIRTUAL ENTERPRISES (RIDE-VE'99). Sydney, Australia, March 23-24, 1999.*, March 1999.

[AHST97] G. Alonso, C. Hagen, H.J. Schek, and M. Tresch. Distributed Processing over Stand-alone Systems and Applications. In *Proceedings of the 23rd International Conference on Very Large Databases (VLDB'97)*, Athens, Greece, August 1997.

[AVA+94] G. Alonso, R. Vingralek, D. Agrawal, Y. Breitbart, A. El Abbadi, H.-J. Schek, and G. Weikum. Unifying Concurrency Control and Recovery of Transactions. *Information Systems*, 19(1):101–115, March 1994.

[CHTY96] J. Camp, M. Harkavy, D. Tygar, and B. Yee. Anonymous Atomic Transactions. In *Proceedings of the 2^{nd} USENIX Workshop on Electronic Commerce*, pages 123–133, Oakland, California, USA, November 1996. The USENIX Association.

[DHL91] U. Dayal, M. Hsu, and R. Ladin. A Transaction Model for Long-running Activities. In *Proceedings of the Sixteenth International Conference on Very Large Databases*, pages 113–122, August 1991.

[HA98] C. Hagen and G. Alonso. Flexible exception handling in the OPERA process support system. In *Proc. of the 18th Intl. Conference on Distributed Computing Systems*, Amsterdam, The Netherlands, May 1998.

[HA99a] C. Hagen and G. Alonso. Beyond the black box: Event-based inter-process communication in process support systems. In *Proc. of the 19th Intl. Conference on Distributed Computing Systems*, Austin, Texas, USA, May 1999.

[HA99b] C. Hagen and G. Alonso. Highly Available Process Support Systems: Implementing Backup Mechanisms. In *18th IEEE Symposium on Reliable Distributed Systems*, Lausanne, Switzerland, October 1999.

[HA00] C. Hagen and G. Alonso. Exception Handling in Workflow Management Systems. *IEEE Transaction on Software Engineering*, 26(9), September 2000.

[Hag99] C. Hagen. *A Generic Kernel for Reliable Process Support*. Ph.D. Dissertation, ETH Nr. 13114, 1999.

[Ivy98] IvyTeam. Structware'98 Process Manager. Available through http://www.ivyteam.com, 1998.

[LASS00] A. Lazcano, G. Alonso, H. Schuldt, and C. Schuler. The WISE approach to Electronic Commerce . *International Journal of Computer Systems Science & Engineering*, September 2000.

[Lie98] H. Lienhard. IvyBeans - Bridge to VSH and the project WISE. In *Proceedings of the Conference of the Swiss Priority Programme Information and Communication Structures, Zürich, Switzerland*, July 1998.

[GMWW99] M. Gillmann, P. Muth, J. Weissenfels, and G. Weikum. Workflow history management in virtual enterprises using a light-weight workflow management system. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises*, Sydney, Australia, March 1999.

[MWW98] P. Muth, J. Weissenfels, and G. Weikum. What Workflow Technology can do for Electronic Commerce. In *Proceedings of the EURO-MED NET Conference*, Nicosia, Cyprus, March 1998.

[SAS99] H. Schuldt, G. Alonso, and H.-J. Schek. Concurrency Control and Recovery in Transactional Process Management. In *Proce. of the 18^{th} ACM Symp. on Principles of Database Systems (PODS'99)*, pages 316–326, Philadelphia, PA, 1999.

[SBG+00] H.-J. Schek, K. Böhm, T. Grabs, U. Röhm, H. Schuldt, and R. Weber. Hyperdatabases. In *Proceedings of the 1^{st} International Conference on Web Information Systems Engineering (WISE'00)*, pages 14–23, Hong Kong, June 2000.

[SPS99] H. Schuldt, A. Popovici, and H.-J. Schek. Execution Guarantees in Electronic Commerce Payments. In *Proc. of the Intl. FMLDO Workshop: Transactions and Database Dynamics (TDD'99)*, pages 193–202, Dagstuhl, Germany, Sept. 1999.

[SPS00] H. Schuldt, A. Popovici, and H.-J. Schek. Automatic Generation of Reliable E-Commerce Payment Processes. In *Proc. of the 1^{st} Intl. Conference on Web Information Systems Engineering (WISE'00)*, pages 434–441, Hong Kong, June 2000.

[ST96] B. Salzberg and D. Tombroff. DSDT: Durable Scripts Containing Database Transactions. In *Proceedings of the 12th International Conference on Data Engineering*, New Orleans, Louisiana, USA, February 1996.

[Tyg98] D. Tygar. Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce. In *Proceedings of the 24^{th} International Conference on Very Large Databases (VLDB'98)*, pages 1–12, New York, USA, August 1998.

[ZNBB94] A. Zhang, M. Nodine, B. Bhargava, and O. Bukhres. Ensuring Relaxed Atomicity for Flexible Transactions in Multi-database Systems. In *Proceedings of SIGMOD'94*, pages 67–78, Minneapolis, Minnesota, USA, May 1994.