The WISE approach to Electronic Commerce

A. Lazcano G. Alonso H. Schuldt C. Schuler
Information and Communication Systems Research Group
Swiss Federal Institute of Technology (ETH)
ETH Zentrum, Zürich CH-8092, Switzerland
E-mail: wise@ccic.inf.ethz.ch
http://www.inf.ethz.ch/department/IS/iks/research/wise.html

May 26, 2000

Abstract

The growing interest in Electronic Commerce practices has lead to a wide variety of models trying to capture the subtleties and complexities of the electronic marketplace. In this paper, we discuss a model based on trading communities, virtual business processes and virtual enterprises. These concepts are at the heart of the WISE (Workflow based Internet SErvices) project, where we have used them to drive the design and implementation of software tools for business to business electronic commerce. The paper briefly describes the model and shows how it is being used in practice as part of the WISE research effort.

1 Introduction

Electronic commerce is a long established practice among companies which use information and communication technology to drive their everyday business transactions. In fact, some decades old retail chains are the direct result of electronic commerce practices. In spite of this proven success, electronic commerce has not been widely adopted until very recently due to the cost and overhead involved [1, 3]. The surge of interest in electronic commerce of the last few years has been triggered by the pervasiveness of inexpensive computers and the widespread use of the Internet. Today, electronic commerce appears in a wide variety of forms: catalogues, auctioning, advertising, direct sales, retailing, marketing, customer services, etc. Of all these varieties, business to business is the most successful [1], which is easily explainable by the trading volume involved and the faster adoption of technology by companies.

When looking in more detail at this particular form of electronic commerce, two basic approaches can be distinguished. The first reflects traditional practices and is mainly geared towards automating the supply chain. A typical scenario involves a large company which auctions its purchase orders and assigns them to the best bidder among a pre-determined set of suppliers. Such an approach has proven to be quite successful and has been in operation for a long time, albeit implemented over leased lines, mainframes, ad-hoc application code and, therefore, available only to large companies with well established commercial ties. The second approach is based on the notion of *enterprise networks*, where different companies pool together their services to offer a more complex, value-added product. This approach is more challenging given its dynamic nature and the greater degree of cooperation required from all parties involved. On the other hand, it is also the most promising approach for small and medium enterprises and the natural model for the Internet.

Unfortunately, and unlike for the first approach, the software support for enterprise networks is entirely inadequate. Existing tools are, in the best case, only partial solutions and difficult to integrate into a coherent whole. Moreover, many of these tools support only a very concrete and narrow aspect of the whole picture (e.g., security, transactional interaction, collaboration), lacking an overall design principle for the entire environment. As a result, very few of such enterprise networks are actually operative and those which are do not necessarily make an optimal use of existing technology. As a matter of fact, the key cost factor today is not the hardware but the effort necessary to put together a working system out of a disparate and heterogeneous set of software tools [3].

To address this concern is the main goal of the WISE project (Workflow based Internet SErvices), its final objective being to develop a coherent software platform for enterprise networks that can be easily and seamlessly deployed in small and medium enterprises. As a first step in this direction, and also as part of WISE, we have developed a simple but powerful model for electronic commerce to be used as the overall design principle. This model is based on three concepts: virtual business processes, virtual enterprises, and trading communities. The notion of virtual business process allows us to qualify and properly identify the activities to be carried out, i.e., how the participating companies will interact with each other. It also allows us to determine to which extent these activities take place across corporate boundaries and where and what the interfaces between these boundaries are. Similarly, the concept of virtual enterprise identifies and gives meaning to the virtual business process by placing it within the context of concrete goals and a well defined environment. The idea of trading community provides the cooperation framework for the participating companies and establishes which of these companies are the involved parties in each step of the virtual business process.

These three concepts, when taken together, provide a very useful starting point for designing and developing software tools for electronic commerce. Thus, the tools to develop are those necessary to support the different aspects of the interaction among different parties (the trading community) who share or provide services (the building blocks of the virtual enterprise) which can be invoked or triggered via some form of Application Programming Interface (API) and linked into a coherent whole (the virtual business process) using a communication network.

In this paper, we describe more precisely both the model and the software platform we have developed. We consider our main contribution to be the fact that WISE encompasses the whole life cycle of a virtual enterprise and addresses all practical aspects of deploying a realistic solution. While many innovative ideas have been developed as part of WISE, our measure of success has been not how well we could address isolated issues but how well we could bring the solutions to the many issues involved into a single system suitable to be used in real applications. Accordingly, in this paper we present WISE as a complete system and discuss concrete aspects of its architecture not by themselves but in relation to the whole. Readers interested in more detail about particular issues can find more information elsewhere [3, 17, 19, 18, 25]

The paper is organized as follows. In Section 2, we present the WISE model for electronic commerce and define the notions of *trading communities*, *virtual business processes* and *virtual enterprises*. Section 3 discusses the goals of the WISE research project. Section 4 presents the architecture and discusses its more relevant aspects. Section 5 examines in more detail key functionality implemented as part of the WISE system. Finally, Section 6 concludes the paper.

2 The WISE Model

Business processes are often used to model, represent, and formalize the most relevant activities within an organization. They can be seen as a set of procedures and rules, expressed in a more or less formal language, in graphical or textual form, describing the steps that must be taken in order to accomplish a given complex task or business goal. Examples of such tasks are to open a new bank account, to obtain a credit, to purchase a computer, to find out the current location of a parcel, to resupply shops, etc. In practice, business processes are also used to both document everyday procedures and as the basis for automating and optimizing such procedures. Moreover, almost any form of electronic commerce can be modeled using business processes. From here, we can provide a more concrete definition of electronic commerce by linking the business objects

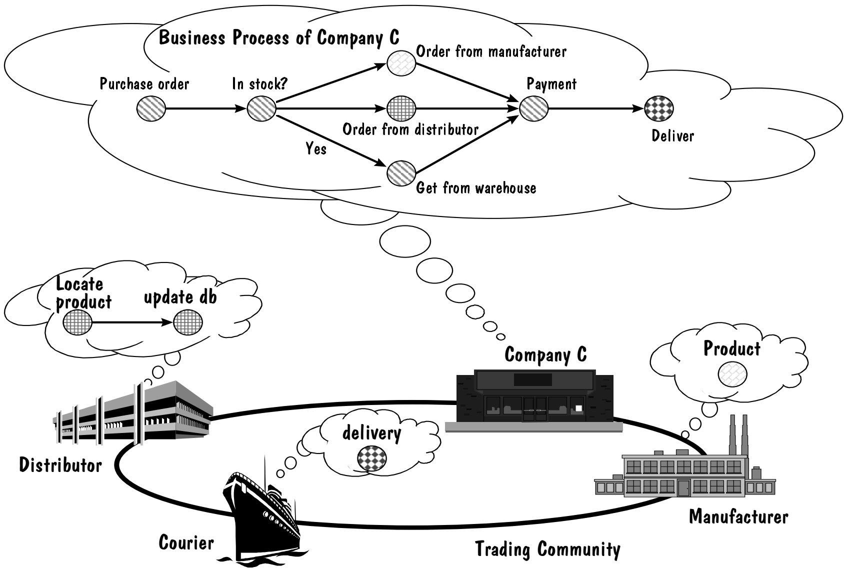

Figure 1: A company incorporating a virtual process as part of its own business processes

with the technology used to implement them in practice. Thus, for the purposes of this paper, 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.*

2.1 Virtual Business Processes, Virtual Enterprises, Trading Communities

The first step towards making this definition a practical reality is to specify how to go beyond the corporate boundaries. This specification, which forms the basis for the WISE model, is based on the notions of virtual business processes, virtual enterprises, and trading communities. These notions can be briefly characterized as follows.

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. Two examples of such processes are shown in Figures 1 and 2. In both cases, the virtual business process appears as a normal process except for the fact that some steps within the process correspond to individual activities or entire subprocesses at different organizations. 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. For instance, in Figure 1, a company incorporates as part of one of its own processes activities to be carried out at other companies. In this case, the company acts as a dealer in merchandise that either it has in stock or obtains directly from other distributors or the manufacturer. It also uses a fourth company for the delivery of the merchandise. As the figure shows, the virtual business process is the one at Company C since it is the only one reaching across the participating companies. Note, however, that it is not the only process involved: the distributor, manufacturer and courier may implement their steps as business processes themselves. This illustrates one important aspect of our notion of virtual business processes. In order to build such a process, we do not necessarily need to know the details of the component processes. Much like encapsulation and modular programming in modern programming languages, we only need to know the "interface" to the component process in order to incorporate it into the virtual business process. Note also that the level of nesting is not limited, i.e., the component process itself could be another virtual business process.

Another important aspect to consider is that virtual business processes are independent of the language

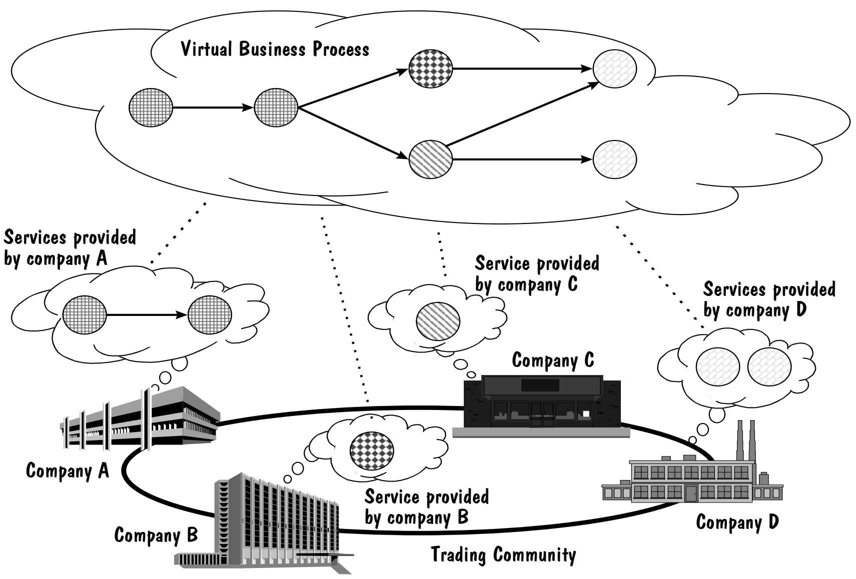

Figure 2: A virtual process as a value added process combining the services of different companies

used to represent either the virtual business process or the component processes. In fact, since what is needed is only the interface, virtual business processes and component processes could use entirely different representations. We only need a way of making the interface public, along with a specification of the characteristics of the component process, and ensure that the interface follows a given agreed format. How this is done within WISE will be described when the mechanisms used for process definition are discussed. Similarly, because what is seen from the outside is only an interface to a component process, the participating companies do not need to expose the internal details of their activities, which often is proprietary information. In general, defining the interface is not a significant problem since it is usually specified as part of the contract binding the companies. As a last point, there are many organizational and formal aspects of interest related to virtual business processes. Some of them will be covered when the implementation is discussed, however, for reasons of space and clarity, all aspects related to the formalism behind virtual business processes have been omitted.

A virtual business process cannot be defined without 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, independently of whether there is a real organization behind the virtual enterprise or not. For instance, in Figure 1 the virtual enterprise is part of Company C while in Figure 2 the virtual enterprise is indeed virtual, in the sense that there is not necessarily a physical organization behind it.

The concept of virtual enterprise is not gratuitous. In practice, the context of a virtual business process is extremely important and the determining factor in terms of feasibility. 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 identify or define the virtual enterprise is in some cases straightforward – as in Figure 1 – while in other cases it can become a major endeavor from the organizational and legal point of view – as it tends to happen in scenarios like the one depicted in Figure 2. Typical issues which arise at this stage are who owns the information about the virtual process (one of the participants?, all of them?, who manages this information?, who has the right to sell this information as a value added service?), where to locate the software platform (fully decentralized?, in one of the participants?, in a neutral organization?, in an intermediary company offering the virtual enterprise as a service to the trading community?), etc. All these tend to be organizational and legal issues beyond the scope of the paper but which should be kept in mind since an adequate software platform will simplify them while, as it is the case with existing tools, a poor design will only make the problem even more complex, greatly detracting from the potential of the enterprise network idea.

Once we have defined what to do (the virtual business process) and the context in which it should be done (the virtual enterprise), we need to define the actors in the scenario. For this purpose, 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.

2.2 The model as a whole

How the model is used in practice can be best seen with an example. Consider, for instance, the scenario shown in Figure 3. In this scenario, the trading community is formed by two different departments of an insurance company (policies department and claims department) and a loss adjuster company. Each member of the trading community provides services (check customer record, payments and entitlements, damage assessment) which are used as building blocks for the virtual process. Based on these services, the virtual enterprise can be created by defining a virtual process in which individual activities correspond to services provided by the participants. Note that there are several ways to interpret this virtual process. One is to see it as totally virtual, as shown in Figure 2, in the sense that the virtual process does not belong to one company within the trading community. Another possibility is that in which a company within the trading community incorporates the services of other companies as elements of its own business processes, as shown in Figure 1. In both cases, the concept is the same and poses the same type of challenges and difficulties, but for the sake of argument we will concentrate on an example following the pattern shown in Figure 2.

The flow in this process is to be interpreted as follows. The insurance company defines a virtual business process to handle insurance claims. In the first step of the process, a clerk in the "claims department" receives the claim and collects all the necessary information about the claim itself, the customer, the involved parties, etc. This information is processed in the "policies department", which uses SAP R/3 as the supporting tool. In this department, the data provided is correlated with the information available in the database, i.e., whether the customer is up to date in payments, whether it is a case covered by the existing policies, and so forth. Once this step is completed, the information is forwarded back to the claims department where the claim is classified, i.e., the specific type of claim (burglary, flood, fire, car accident, damages by third parties, etc.) is determined.

For the purposes of this example, we will consider only two types of claims: burglary and fire. In case of burglary, the claim is returned to the policies department where, based on the police report, the total value of the objects is calculated, the payment limit established and an estimate is made of how much the insurance should pay. In case of fire damage, the process is more complicated. To deal with such cases, the insurance company resorts to a loss adjuster company which will be the one responsible for making an estimate of what needs to be paid. In the example, the loss adjuster, who uses a workflow engine to drive its business processes, provides an entry point similar to an API which the insurance company can invoke. Through this interface, the loss adjuster receives the necessary data and triggers its own business process.

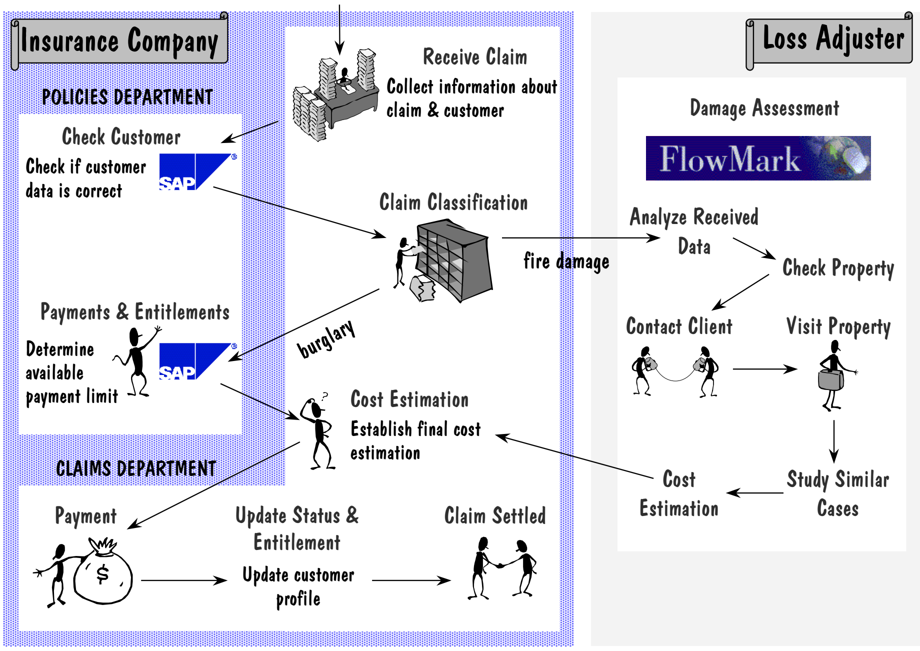

Figure 3: Example of a virtual enterprise process

This process consists of checking the property, i.e., who is the legal owner of a building, arrange a meeting with the client, visit the damaged property, compare with similar cases or, in case of major disasters like floods or earthquakes, determine what other sources of payment may need to be considered, and a cost estimate is made. The cost estimate is then forwarded to the insurance company, which using a similar mechanism can incorporate this step into its own business process. After the estimation is completed, the payment is made, the corresponding records updated (so that a customer is not paid several times for the same claim), and the claim settled.

This example shows how to introduce the loss adjuster process as one more element of the overall claim processing procedures, even if the loss adjuster is an entirely different company. The key to the WISE approach to virtual enterprises is to treat such interoperability problems as process encapsulation problems where, as long as there are well defined input and output parameters, the rest can be treated as a black box.

The practical questions which arise when implementing such a virtual process can be best answered by following the proposed model. Thus, the overall goals for the process are part of the virtual enterprise. For instance, if the goal is a reduction of the claim processing time, this can only be expressed in relation to the virtual enterprise. The monitoring mechanisms cannot work if limited to one participant, therefore, they should be part of the global agreement between the participating companies. All these agreements and the way information is distributed and accessed by the partners form the virtual enterprise. Similarly, when concrete queries arise, the system needs to have some sort of user identifier so that the information is given to authorized parties. Who the authorized parties are is part of the description of the trading community. The same can be said of the physical distribution, where each element in the process is specified as part of the API's and the addresses of the partners listed in the trading community.

2.3 Applicability of the model

We believe trading communities, virtual enterprises and virtual processes are a very powerful approach to interpret and identify the needs of a wide range of electronic commerce practices. For instance, in the case of retailing, a company can provide a much more sophisticated product by outsourcing aspects of the operation which are not central to its activities. A common example are companies offering a product (books, CD, flowers) without actually handling (producing, storing or delivering) the product themselves. Most of the handling is left to companies providing specialized services, which allows to significantly reduce the operational costs. The virtual enterprise model naturally captures such scenarios by simply having the distribution and delivery services incorporated as activities within the business processes of the company selling the product as shown in Figure 1.

The same type of scenarios can be created for retailing, brokering, customer services, and supply chain electronic commerce [1]. In all these cases, we are particularly interested in the establishment of a higher order entity which allows different companies to collaborate and exchange their services. Thus, in the case of retailing (e.g., buying a computer using an online interface that allows to put the system together starting from a collection of basic components), we are interested in implementing mechanisms to allow a company to outsource part of its services. In the case of brokering and customer services, we are interested in scenarios where a company bridges the gap between several others and provides value added services based on those provided by the other companies. Finally, in the general case of business-to-business electronic commerce, we are looking at ways to build applications spawning tasks beyond the corporate boundaries.

3 The WISE Approach: from Model to System Specification

From our point of view, the real challenge in electronic commerce is how to provide a complete solution, that is, how to build a software tool capable of supporting the entire life cycle of a virtual business process. By life cycle, we mean that virtual business processes should not be seen as a one time programming effort (today's approach) but as valuable assets to be maintained for as long as they are in use. Furthermore, the support for this life cycle can only be done 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. Without solutions to such problems, the notions of virtual enterprise, trading community, and virtual business process described above may be conceptually appealing but become irrelevant for practical purposes.

3.1 The Virtual Business Process Life Cycle

To motive the approach we have taken in WISE, we need first to clarify what we see as the life cycle of a virtual business process (Figure 4). This life cycle is centered around the process since it is the determining factor shaping the collaboration between the partners in the trading community and the overall nature of the virtual enterprise.

In this life cycle, a trading community starts by providing a number of well defined services which are made known to all participants. Using these services as basic building blocks, a virtual business process is defined. Once defined, the process can be executed within the context of a virtual enterprise. Based on this context, the execution of the process can be observed and the result of these observations used to improve and fine tune the virtual business process. Finally, the virtual business process also serves as the basis for the communication and coordination of partners (what we call context based communication in the sense that the interactions are determined by the participation of the partners in the process). The partners can use this ability to optimize the execution of ongoing processes or tailor their services for the creation of future virtual business processes. In more detail, the interactions between the three elements of the model are as follows:

- *Interaction Trading Community - Virtual Business Process.* This essentially involves mechanisms

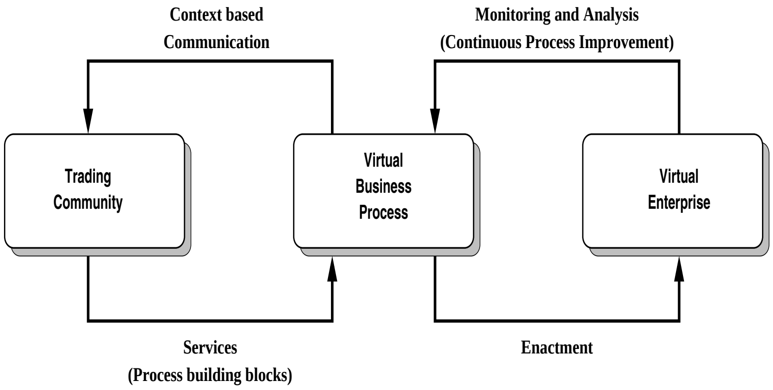

Figure 4: Interactions between the elements of the WISE model

whereby companies can advertise their services, other companies can look at them and, finally, incorporate these services into their own business process without requiring ad-hoc development. The services should appear as a catalogue in which the entries represent activities that can be incorporated into a business process. The catalogue should also provide important information about the activities so that they can be directly incorporated from the catalogue into the virtual business process.

- *Interaction Virtual Business Process - Virtual Enterprise.* It involves actually running the virtual business process across the Internet and bridging the interoperability differences between the participating systems. Here it needs to be pointed out that enactment is not just an interoperability problem. In fact, our experience shows that interoperability is easier to address once the notion of virtual process is in place, since it provides a much narrower and well defined framework in which to develop a solution.
- *Interaction Virtual Enterprise - Virtual Business Process.* This is the first feed back loop in the life cycle. It allows to use the observed behavior of the process when being executed to improve its design and gather useful information about the virtual enterprise. Technically, it requires to keep track of executing processes so that useful information can be provided about them. This information includes overall execution time, delays, flow analysis, quality of service requirements, bottleneck identification, etc. Using this information, the virtual business process can be continuously improved.
- *Interaction Virtual Business Process - Trading Community.* This is the second feedback loop. Its purpose is to allow the partners in the trading community to establish communication with other partners based on their responsibilities within the virtual business process. The idea is that a partner A should not communicate with a partner B since this means partner A must have a lot of implicit information about the process, its state, configuration, and so forth. Instead, a partner should communicate with whomever is responsible for a given activity in the virtual business process. The system should resolve who is actually the responsible party and then set up a communication channel. This idea establishes a framework for context specific collaboration, i.e., one in which communication is not based on point-to-point routing but based on the dynamics of the process execution.

3.2 A Software Platform for Electronic Commerce

From these ideas, we can derive a basic specification of the software platform to develop. Since the main abstraction we use is the process, the specification revolves around how to support processes. We do this based on four modules (Figure 5) which can be best described as follows:

- *Process definition* includes all the functionality related to the definition and specification of a virtual business process. This involves a language in which to specify the process and the corresponding sup-

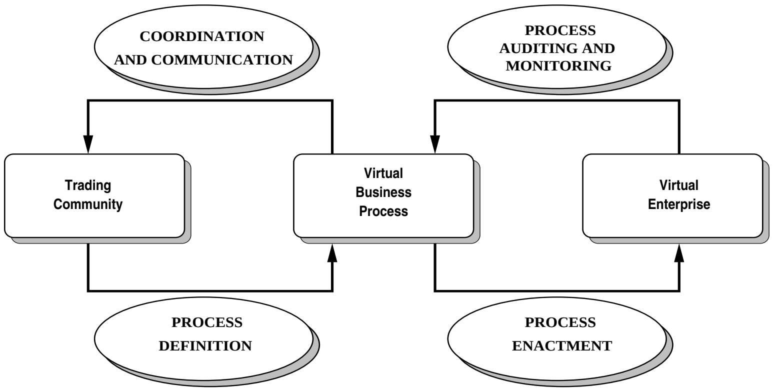

Figure 5: Schematic specification of the WISE system

port environment. Our approach to process definition is not just to build a more or less sophisticated interface but to provide an expressive process definition language incorporating features such as exception handling, interprocess communication, event management, and execution guarantees. We strongly believe this functionality is sorely needed since a business process, virtual or not, is just a description of an evolving practice. Applying known software engineering techniques will certainly be of practical advantage.

- *Process enactment* includes compiling the description of the virtual process so as to transform it into a representation suitable for executing the process and the actual execution environment. The process enactment engine should not limit itself to the actual execution of processes but also support industrial strength features such as back-up mechanisms, replication, load balancing, process migration, persistence, and recovery.

- *Monitoring and Analysis* involves keeping track and recording every step of the execution of the virtual process for load balancing and quality of service purposes as well as, later on, for analysis of the behavior of the process. The idea is to capture the entire execution history for both on-line use (load balancing, resource reservation, system administration) and off-line analysis (mining of the process data for business re-engineering, provisioning, and optimization purposes).

- *Coordination and Communication* represents all the functionality necessary to allow the different participants to communicate with each other in the context of the execution of a process. We do not expect all virtual business processes to be fully automated. In fact, in most cases human intervention will be necessary and the system should be able to facilitate this intervention by allowing to identify partners by the work done within the process. Queries such as "contact the person responsible for activity X" or "contact the company who entered this data item" should be readily answerable and result in the requested communication channel being established without the requester having to provide information other than that derived from the virtual process itself.

These four modules fully support the three concepts used in the model. For instance, the trading community can be specified as the users of the process enactment module and the sources of the entries in the catalogue of basic building blocks for the process. The virtual business process is the result of the process definition module and the virtual enterprise is that defined by the configuration of the process enactment and coordination modules. In what follows, we will discuss each of these modules separately. However, within WISE, we emphasize the fact that they are only different aspects of the same problem. A system delivering only one of these modules is only a partial solution that will be difficult to use in practice. A complete solution must provide the four modules and integrate them. In this sense, the best way to describe WISE may be as an integration effort, i.e., an attempt to combine several different technologies into a coherent whole. These ideas are summarized in Figure 6 which also contains some indications of the commercial tools used within WISE.

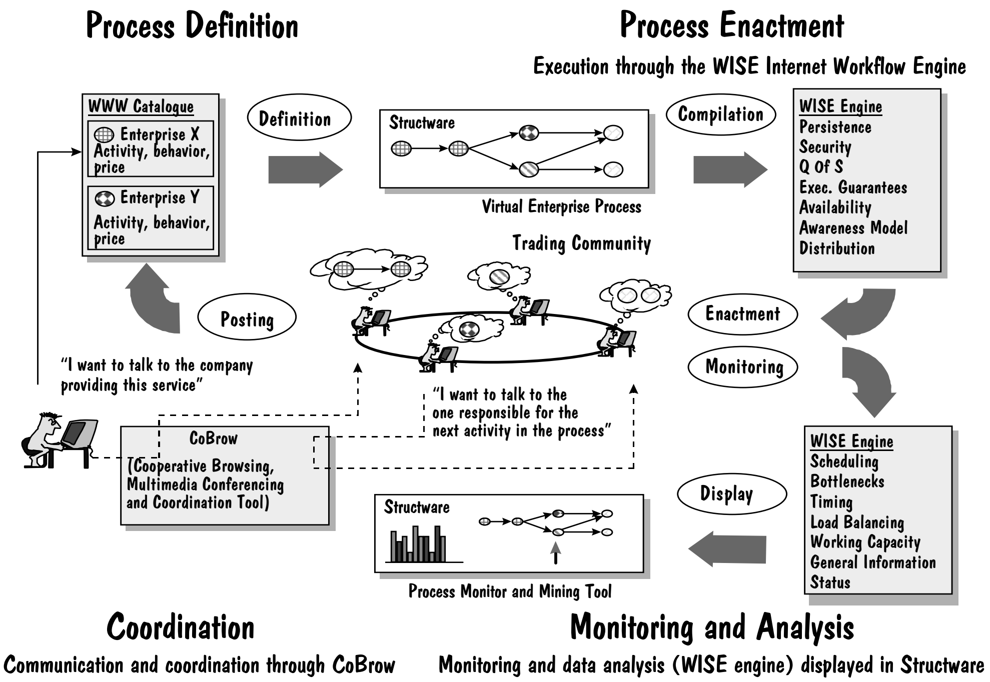

Figure 6: The WISE architecture

To complete the system specification, it is necessary to provide an overall strategy for the system. In WISE, this is based on the idea of distributed computing environment. Here is where the idea of process becomes relevant: the virtual business process can be seen as a distributed program running on some form of middleware linking together the resources of the trading community [5]. These resources are the concrete applications or services offered to the virtual enterprise by the trading community and are used as the basic building blocks of the distributed program (the virtual business process). From here, the type of software to develop is the type of software that would be needed to support the definition and execution of such a coarse grained distributed program.Taking this analogy a step further, it can be said that any realistic solution to electronic commerce must take into account the true complexity of the problem. We see electronic commerce processes as valuable assets needing to be properly specified, designed, developed, tested, debugged, and maintained in an effort not unlike software life-cycles. In order to do this, the system has to provide adequate tools, otherwise these tasks become extremely difficult and largely ad-hoc endeavors as it is the case today.

4 The WISE Architecture

The specification above can be seen as that of 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. This is the same idea behind the OPERA process support system, a system developed in the Information and Communication Systems Research Group of ETH Zürich with the objective of providing a generic process tool which can be tailored to different applications [4]. As part of the WISE project we have extended OPERA to implement the specified four modules.

4.1 Process Definition

In WISE, virtual business processes are constructed by using the services offered by different companies (part of the trading community) as building blocks. Each one of these services can be a process in itself although beyond providing the necessary interfaces, the nature of these services is orthogonal to WISE. The virtual business process integrates the services of the different companies establishing the order of invocation, the control logic and the data flow between the participants in the same way a workflow process orchestrates business models within a single corporation. In order to do this, there are two elements that WISE must provide as part of the first module. The first element is a mechanism for the participants to publish their services. The second is a way to define a process based on such services. For these purposes, WISE uses a WWW catalogue and a commercial business process modeling tool.

The WWW catalogue uses Java applet/servlet technology to allow companies in the trading community to advertise their services and to "see" the semantics of the services provided by other companies [22]. While a list of mere "read-only" pointers would seem to suffice, in practice companies need to understand the behavior of a service before they can incorporate it into their business process. The catalogue, instead of just URL's, contains objects encapsulating the behavior of a each service. Then IvyFrame is used, which is a Java version of a business modeling tool supporting simulation and analysis (Structware, commercially available product of IvyTeam, one of the industrial partners in the project). IvyFrame allows a company to see the exact characteristics of each entry in the catalogue, not only in terms of the steps it takes but also in terms of its functional specification: cost, average duration, guarantees, requirements, side effects, etc. When a company wants to make an entry in the catalogue, it specifies the service using the modeling tool. This generates code that is inserted in the catalogue and executed in the simulation and analysis tool every time another company is interested in using the service as a step within a virtual business process. Note that the specification of the service does not represent additional overhead. Such specifications are necessary independently of the way the virtual enterprise is implemented.

From the catalogue, a drag and drop type of interface is used to build the virtual business process. The tool we use for process definition is the already mentioned *Structware/IvyFrame* [20, 22], 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.). This analysis capability is the one used in the catalogue, but it can also be used to analyze the behavior of the whole virtual business process once it is constructed. In Figure 7, the virtual process example of Figure 3 has been modeled using Structware. The services provided by the policies department of the insurance company (Check Customer and Payments & Entitlements, with SAP R/3 as the supporting tool) and the loss adjuster company (damage assessment, a business process in itself, executed by a workflow engine) are incorporated as steps within the virtual business process.

In terms of process definition, Structware supports the standard flow control primitives of a workflow tool. It is possible to define conditional branching, nested processes, and assign additional information to each task within the process. This last point is important from the point of view of WISE since it allows to use this additional information as the configuration information necessary to enact the process. In this way, no additional interfaces are necessary. The designer can define everything that is needed for the execution of the process using a single tool. Moreover, since the tool is only a simulation tool and not an enactment tool, we can incorporate as well the services available at the catalogue without having to worry at this stage about how they will be invoked when the process needs to be executed. The necessary invocation parameters are provided by the company posting the service. This separation of service definition and

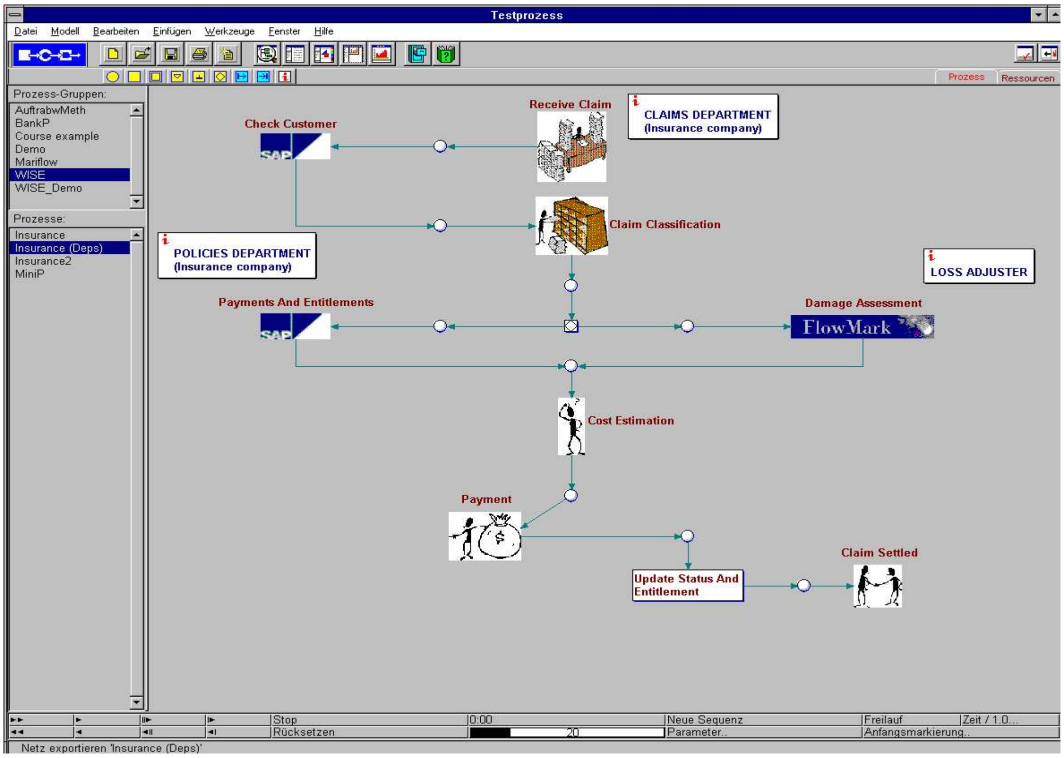

Figure 7: The virtual process of Figure 3 modeled in Structware

service enactment gives us the same advantages as the separation of interfaces and code in modular and object oriented programming. In fact, we see this entire procedure as a form of high level, coarse grained programming. We have successfully applied this idea of *workflow programming* within WISE and other projects in order to provide sophisticated language primitives not available in commercial workflow tools. Some of this functionality will be discussed in Section 5.

Perhaps one of the most critical aspects of our approach is the language used to define and execute processes. The problem here is that the real issues are not related to the language itself but to the overall flexibility of the approach. Therefore, our research efforts have not been focused on a particular abstraction (such as Petri-Nets, State-Activity charts, or transaction models) but to the functionality the language should support. Moreover, our experience with users shows that it is not sensible to impose a language on the users. Most companies already use their own business process modeling tools and will not be willing to change these tools for others. As a result, what we support in terms of language is an internal language into which we compile many other languages [18]. This internal language is called OCR (Opera Canonical Representation) and it is optimized for enactment purposes (minimize overhead since the language is persistent, optimize flow to speed up the navigation through a process, and incorporate enough information for recovery and monitoring).

OCR is not intended to be used directly by users. In WISE, we use the business modeling tool Structware as the front end. Users define processes using this tool (Figure 7) and the WISE system compiles the representation of this tool into OCR (Figure 8), which is the representation used by the WISE system to work internally. This OCR representation is a complete description of the virtual business process, containing the specification of the different steps in the process, the applications or services invoked by each of them, and the control and data flow within the process. Some important aspects of OCR are described in Section 5, and additional information can be found in other publications related to OPERA [4, 18].

```
PROGRAM Receive_Claim()
        RETURNS (customerName: String, policyNo: String,
                 amount: Integer) (
        COMPENSATION    SYSTEM,
        ROLLBACK        SYSTEM,
        RESTART         not-restartable,
        SYS             Unix,
        HOST            machine1,
        COMMAND         "receiveClaim.tcl"
);

PROGRAM Check_Customer(INNAME : String, INNR : String )
        RETURNS () (
        COMPENSATION    SYSTEM,
        ROLLBACK        SYSTEM,
        RESTART         not-restartable,
        SYS             SAP,
        HOST            sappc,
        CLIENT          "800",
        USER            wise,
        PASSWORD        *****,
        WORKFLOW        "Check_Customer"
);

PROGRAM Claim_Classification(customerName: String,
                     policyNo: String, clientOk: Integer)
        RETURNS (customerName: String, policyNo: String,
                 clientOk: Integer, claimType: String)(
        COMPENSATION    SYSTEM,
        ROLLBACK        SYSTEM,
        RESTART         not-restartable,
        SYS             Unix,
        HOST            machine2,
        COMMAND         "claimClassification.tcl %customerName%
                         %policyNo% %clientOk%"
);

PROGRAM Damage_Assessment(customerName: String,policyNo: String,
                  clientOk: Integer, claimType: String)
        RETURNS (customerName: String, policyNo: String,
                 clientOk: Integer, claimType: String,
                 paymentLimit: Integer) (
        COMPENSATION    SYSTEM,
        ROLLBACK        SYSTEM,
        RESTART         not-restartable,
        SYS             Flowmark,
        HOST            fmpc,
        USER            user1,
        PASSWORD        *****,
        SERVER          exmsrv,
        DATABASE        insurdb
);

. . .

PROCESS Insurance ()
        RETURNS ()
        WHITEBOARD (policyNo: String, clientOk: Integer,
                    customerName: String, claimType: String,
                    paymentLimit: Integer, amount : Integer),
        COMPENSATION      SYSTEM,
        ROLLBACK          SYSTEM,
        RESTART           not-restartable

TASKS
        PROGRAM  T_Receive_Claim: Receive_Claim ()
                STORE (amount > amount) (
                ACT initial (STARTUP),
                COND true
        );

        PROGRAM T_Check_Customer: Check_Customer
                  (INNAME = T_Receive_Claim.customerName ,
                   INNR = T_Receive_Claim.policyNo)
                STORE ( ) (
                ACT finished (T_Receive_Claim),
                COND true
        );

        PROGRAM T_Claim_Classification: Claim_Classification
                  (customerName = T_Receive_Claim.customerName,
                   policyNo = T_Receive_Claim.policyNo,
                   clientOk = "1")
                STORE (customerName > customerName,
                       policyNo > policyNo, clientOk > clientOk,
                       claimType > claimType ) (
                ACT finished (T_Check_Customer),
                COND true
        );

        PROGRAM T_Damage_Assessment: Damage_Assessment
                  (customerName = WB.customerName,
                   policyNo = WB.policyNo,
                   clientOk = WB.clientOk,
                   claimType = WB.claimType)
                STORE (customerName > customerName,
                       policyNo > policyNo, clientOk > clientOk,
                       claimType > claimType,
                       paymentLimit > amount) (
                ACT finished (T_Claim_Classification),
                COND T_Claim_Classification.claimType =
                      "Fire_damage"
        );

        . . .
     END TASKS

END PROCESS
```

Figure 8: OCR version of virtual process example

4.2 Enactment

The enactment of the virtual business processes is performed by the WISE engine which forms the core of the runtime infrastructure (Fig. 9). The engine has been built as an extension to the OPERA engine, a process support kernel also developed at ETH Zürich [17, 4]. It borrows many ideas from workflow management [12, 9], and uses known techniques for distributing this functionality [26, 21, 10]. In addition, a considerable amount of extensions have been introduced to make workflow a suitable foundation for electronic commerce (for a different approach to electronic commerce based on workflow technology see [23]).

During execution, the engine functions as an interpreter, with the virtual business process being the program to execute. Since these programs are constructed using the services of the trading community, the engine can be seen as an "engine of engines" in the sense that it manages the interaction and flow of information between the computer systems of the partners in the trading community.

Internally, the WISE engine provides the usual range of functionality to be found in state-of-the art workflow systems. Processes are stored persistently and every step of the execution is recorded in the system's database for both monitoring and recovery purposes. This allows, for instance, to resume work where it was left off instead of having to start at the beginning of the process after site failures. The engine performs navigation based on the information derived from the process and can also make decisions about what to do next based on the status of the system (this is what we call *awareness model*). Such decisions involve load balancing, redistributing work to other servers, restarting failed servers, notifying missed deadlines or predictions about missed deadlines, etc. The engine can also make predictions about future load and reserve network bandwidth accordingly in order to guarantee a certain throughput (in this

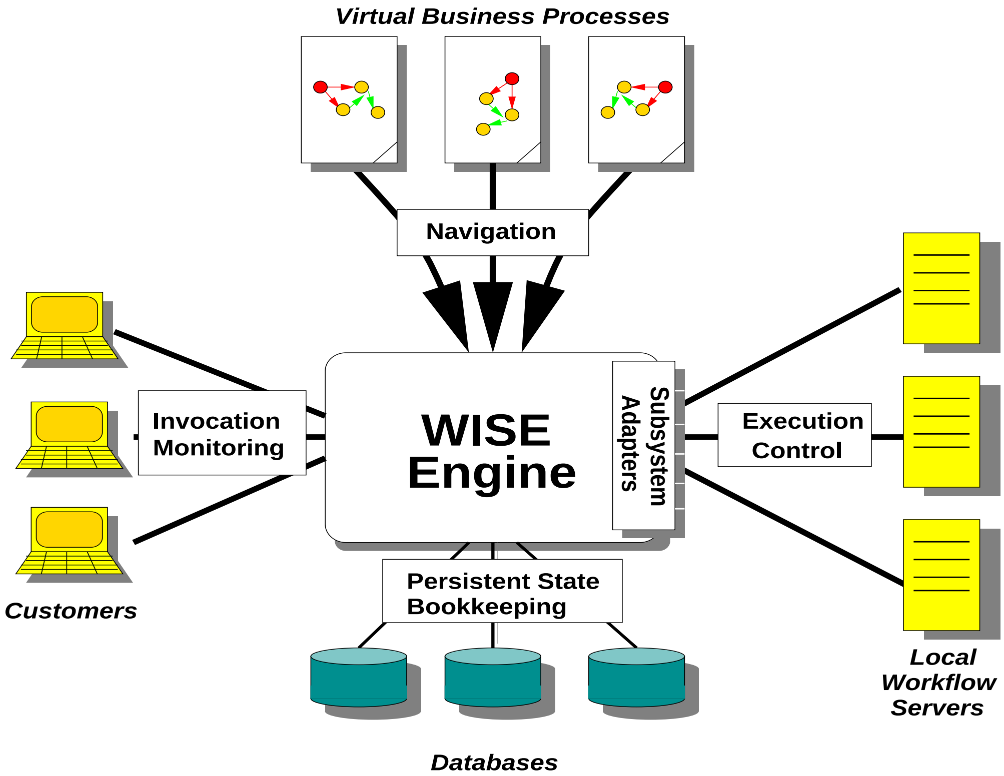

Figure 9: The WISE enactment framework

case, number of processes per unit of time).

In terms of the interactions with external systems, the engine uses *subsystem adapters* to encapsulate the knowledge necessary for the interaction with a particular system. These adapters are created filling out a structured form in the design interface. They allow to add new components to the system in a very user-friendly manner (the information required in the adapter is nevertheless of a technical nature: how to establish connections, which protocol to use, how to pass parameters, what type of interface should be used, underlying operating system, exit codes, etc.). Interaction with users takes place via *customer interfaces* which are based on Java technology, allowing them to run on a wide variety of platforms, including standard web browsers. These interfaces are of a special kind when they are used for monitoring purposes (see below), otherwise they resemble a standard worklist.

Of the many improvements incorporated into the engine, there are three which deserve special attention: security, quality of service, and increased fault tolerance. Given the nature of the data exchanged between the different participants in the trading community, WISE incorporates the necessary security mechanisms in the form of encryption of data for transmission over the network as well as a complete set of authentication measures for both execution, access, and monitoring of the process. Also, to make the notion of trading community viable given the current limitations of bandwidth, the WISE engine incorporates quality of service guarantees based on execution statistics and network characteristics, an approach which is described in detail later in this paper. The increased fault tolerance guarantees of the WISE framework address the problem that a single server will necessarily be a single point of failure, rendering all virtual business processes unavailable if it fails. To overcome this limitation, a cluster concept has been added to the WISE engine which allows to extend the system into a distributed architecture. On top of this architecture, backup mechanisms are used to guarantee the continuous availability of processes.

4.3 Audit and Monitoring

Process design is a difficult task. It is not reasonable to expect the designer to reach the best possible solution at the first attempt. In particular, in virtual enterprise environments it is difficult to foresee all possible eventualities until some example runs are available. Process design is rather an iterative procedure where WISE can be of great help by providing accurate measurements of all the characteristics affecting the execution of a process: overall duration, bottlenecks, relative duration of each task with respect to the duration of the entire process, loads at each participant site, deadlines missed, and so forth.

For modeling purposes, the WISE system uses a *history space* where information about all already executed processes is stored and organized in a way that facilitates its analysis. This process history is exploited within WISE for two purposes. The first one is to create a system model for using in on-line decision making regarding the configuration and status of the system. The second is to act as the basis of a process history data warehouse which coupled with a process data mining tool will allow to analyze the data off-line. Therefore, the history space will act as a repository where designers can explore existing data to find out relevant information about process execution.

Additionally, it is not reasonable to expect a virtual enterprise process to execute blindly. Accordingly, WISE provides tools to find out the status of any process in the system for monitoring purposes in order to allow users to keep track and troubleshoot them when necessary. In order to provide this functionality, WISE incorporates the necessary modules within the execution engine to keep track of executing processes.

For displaying the status of currently running processes, we take advantage of the capabilities of IvyFrame. Aside of the modeling and simulation functionality, IvyFrame allows process models to be exported as Java applets which already provide a graphical user interface to display process execution information and can be embedded in our administration and monitoring tool. This tool offers a couple of administrative services such as the instantiation of processes, an overview over currently loaded process templates and currently running instances of processes in the virtual enterprise and allows to display their status graphically. In the same way an IvyFrame process is compiled into a format understandable by the WISE engine, the information about the state of an active process instance produced by the WISE engine at runtime is translated into a set of tokens that can be displayed using the interface of the Petri net-based IvyFrame applet.

Finally, WISE also includes an awareness model [13] that allows the engine to make decisions based on its own status and that of the participants. The intuition behind this awareness model is that WISE needs to have enough information about what is taking place in the system in order to make informed decisions about configuration, quality of service, resource reservation, load balancing, and so forth. This functionality goes beyond what conventional workflow systems provide, which limit themselves to keep track the status of any process and store the resulting data. The goal here is to generate near-real-time information to be able to asses the state of the system at any point in time. At this stage, we have the necessary functionality to generate this information and gather it in a centralized location. We are currently developing the algorithms implementing load balancing, resource reservation, and quality of service guarantees, which are the ones making more use of this information.

4.4 Coordination and Collaboration

Unlike in conventional workflow engines, WISE will operate in an environment where neither the different participants nor the different elements of the process are necessarily in a position to easily exchange information among themselves. Note that, as the concept of trading community implies, each participant could be not only on a different location but in an entirely different company. It is nevertheless important for the participants to be able to communicate in order to resolve the unavoidable inconsistencies and minor problems associated with any process. These context based difficulties may require the communication and collaboration among persons who may only be identifiable through a role definition. If, for instance, there is a problem with some of the process data (wrong insurance number, wrong address, invalid code, etc.), a participant may request to send a message or to contact the person who introduced the data. It is in this sense that the communication and coordination will be context based.

To achieve this goal and in order to avoid having to develop the communication infrastructure as part of WISE, we use the results of CoBrow (Collaborative Browsing in Information Resources) [27], a project of the Communication Systems Research Group at ETH Zürich. CoBrow provides mechanisms supporting interactive (e.g., video conferencing) and non-interactive (e.g., group calendars) collaboration on a WWW-based platform. By integrating CoBrow within WISE, WISE can offer a variety of communication possibilities using the virtual business process as context.

5 WISE Functionality

WISE incorporates several novel features. In terms of the language used, interesting aspects include exception handling [18], event management, and inter-process communication mechanisms [19]. In terms of overall execution, relevant issues are atomicity and quality of service guarantees. It is this functionality, separately and as a whole, that differentiates WISE from any, to our knowledge, of the existing approaches to both electronic commerce and process management.

5.1 Exception Handling

In electronic commerce, where a large number of processes are executed, having exceptions is bound to be a normal occurrence. Any programming tool intended for large, complex applications has to face this problem. These tools usually incorporate exception handling mechanisms to separate the failure semantics from the program logic and thus facilitate the design of readable, comprehensible code. First in OPERA and then in WISE we have incorporated mechanisms for failure handling which are based on well-known programming language concepts [18]. Among these, we take advantage of the language extensions for structured exception handling first presented by Goodenough [15] and subsequently adopted in a number of programming languages (CommonLisp [28], C++ [29] and Java [11]). The key aspect of our exception model is separating exception detection from exception handling. The process description is decomposed into the process itself, which contains only the business logic, and the exception handlers, which are activities or subprocesses triggered when another activity reaches an exception. Exceptions can be generated by external programs or as semantic exceptions internal to WISE. The latter are like ordinary activities and, therefore, can occur anywhere inside a process. The former are based on user-provided predicates that define under which circumstances a given exception is raised. Like in structured programming languages, exception are propagated upwards along the invocation hierarchy until an appropriate handler is encountered, a very useful feature when dealing with nested processes. It is also possible to define different behaviors after the exception has been dealt with: return to the signaler, abort the signaler and return to the invoker, etc. With this idea, the process language of WISE adds a significant degree of flexibility when defining virtual business processes and greatly increases the overall fault tolerance of the system.

5.2 Event management and inter-process communication

Current process specification languages usually support the definition of nested processes, which helps to achieve some modularity by defining blocks of activities as subprocesses. In general, the data and behavior of a process in encapsulated within its own execution scope, preventing other processes from accessing this information until the process terminates. As a result, each subprocess appears necessarily as a black-box, and structuring a process by means of subprocesses greatly decreases the degree of parallelism since progress in the top level process is not possible until the complete subprocess has finished. We have addressed this problem in WISE by means of interprocess communications based on event- and rule-based (ECA rules) mechanisms [24, 31, 14]. Our approach differs from standard ECA rule management practice in that we do not use an active database system or a distributed event engine as the underlying platform. This requirement is often not practical since neither active databases nor distributed event engines are widely available today. Moreover, such architecture introduces a clear dependency on the functionality of the underlying system, thereby limiting the possibilities of generalizing the idea. We opt instead for an architecture in which interprocess communication is implemented as an additional module of the execution engine [19]. In this architecture, interprocess communication is based on the exchange of events between concurrently running processes. Events are typed and parameterized signals which can be raised by a running activity to inform other processes of certain situations reached or produced during its execution. The parameterization allows to pass arbitrary context data with the event. At runtime, an activity can signal only the events previously registered with the system as part of the activity declaration. In principle, the visibility of an event is limited to the block or process an activity belongs to (similar to local variables in a programming language). To enable inter-process communication, a subscription mechanism is used. This mechanism supports the propagation of events over several levels of nesting. Within its implementation of interprocess communication, WISE also incorporates recovery and exception handling features related to events, in order to deal with the effects of aborted processes and consequently revoked events. These new language primitives enhance the flow-oriented features of process languages with event mechanisms, giving the process designer the option to arbitrarily combine both paradigms. We believe that the interprocess communication facilities implemented in WISE can lead to better designed virtual business processes, since designers do not have to use workarounds like using flat, unstructured process models in order to accommodate the complex information flow between the different parts of a process.

5.3 Quality of Service (QoS)

To make the notion of trading community viable given the current unpredictable variations of available bandwidth in the Internet, the WISE engine incorporates quality of service guarantees based on reservation of network resources. The necessary information for reserving resources and providing guarantees is derived from a combination of execution statistics and the awareness model created by the monitoring and analysis component. Furthermore, we assume that many Internet Service Providers will support resource reservation and RSVP (Resource Reservation Protocol) in the near future due to the upcome of multimedia application [7].

When combined with admission control policies, resource reservation allows WISE to provide delay and throughput guarantees for the exchanges between partners in the trading community. Resource reservation is, however, expensive both in itself and also because it is likely that the amount of resources reserved will not always be exactly what is needed. To minimize the cost and avoid wasting resources, reservations are dynamically adjusted using statistical analysis of the past history of processes and the awareness model in order to calculate the resources that will be needed in the immediate and near future. Note that since the structure of the processes is known, it is possible to calculate future demands based on the current status of the process. Using information about what happened in the past and the process structure allows WISE to derive very precise estimates of what it is going to happen as the processes evolve. Bottlenecks, potential surges of demand, or even blocking behavior can be predicted in advance and dealt with beforehand by readjusting the reservations. Moreover, since the prediction process also incurs in certain overhead, a user can select among three process categories: critical, important, and normal. For each one of them, WISE provides different guarantees at different costs.

For a critical process, WISE guarantees delay and throughput. That is, WISE guarantees that the execution of a process will not be delayed beyond a certain deadline and that the system will be able to process a given amount of processes at the same time. This is accomplished by reserving bandwidth among all partners involved so that the network cannot cause any delay in process execution. It is for critical processes that predictions are made about future loads and the reservations adjusted accordingly. For an important process, WISE provides statistical guarantees regarding the delay in the execution but no throughput guarantees are made. The techniques used include increasing the execution priority, attending to load balancing parameters, and providing early warnings if necessary. While these guarantees are not as strong as for critical processes, they still allow to bound the probability a process will be delayed. Finally, normal processes are executed in best effort mode, that is, without guarantees regarding possible delays or throughput.

Monitoring of the system for QoS purposes is done by recording and logging information about network utilization and usage of previously reserved resources. When the monitoring system detects that a given measurement goes beyond a pre-specified thresholds, an awareness event is multicast to all relevant components which can then readjust the behavior of the system. In addition, and in order to determine if and how much of the reserved resources can be used for important processes, network characteristics like round-trip-time and packet-loss-rate for traffic in best effort mode are also recorded and logged. This near-real-time information is then used to assess the network state for early warnings, increasing execution priority of processes likely to be delayed and to support comprehensive load balancing.

5.4 Execution Guarantees

One important requirement of processes is to terminate in a well defined state although failures and concurrent access to shared resources has to be considered. In databases, transactions are used to encapsulate database operations so as to provide *Atomicity*, *Consistency*, *Isolation*, and *Durability* [16]. Transactions provide clean semantics to concurrent executions and a powerful abstraction for optimization purposes [2]. We have applied these same ideas within WISE [25] leading to the notion of *transactional processes* so as to provide useful abstractions to reason about the correctness of the system, to express properties of the execution of processes, and to tune up the overall architecture. However, since we work with processes some form of *light-weight* transactions should be used. Thus, within a process, instead of using a unique construct encompassing all transactional properties, several separate constructs are used to group activities according to the desired semantics. We consider *spheres of atomicity* (atomic units with a more general semantics as the standard all or nothing), *spheres of isolation* (isolation units, much like critical sections in traditional operating systems), and *spheres of persistence* (determining whether the activities in the sphere are to be made persistent or not). A process is divided into spheres by the user and the system automatically checks whether the specification is correct. This is specially useful in regard to atomicity since whether the entire process is atomic or can be guaranteed to reach a consistent state depends on the applications with which it interacts. These spheres are then used during execution to decide what to do in case of failures, conflicts with other processes, rollback of already executed parts of the process, and determine what parts of the process are to be made persistent. Like with exceptions, we rely on user-provided information to decide the atomicity status of each activity within the process and whether two processes or activities within different processes conflict. Unless the user specifies otherwise, the process is assumed to be persistent in its entirety.

The need for execution guarantees is mostly in terms of atomicity as some other authors have pointed out [8, 30]. Using the notion of spheres of atomicity explained above, we can formally analyze the structure of a process and determine whether we will be able to reach a consistent point at any time or there are some execution paths that, once started, are irrecoverable (we cannot rollback, we cannot compensate, and we may not even be able to go forward). This analysis is likely to be very useful for process designers who can then identify problematic steps and try to solve them in a different form. Finally, with spheres of isolation, we are able to schedule concurrent processes correctly and to synchronize access to shared resources. These spheres of isolation together with the spheres of atomicity provide the necessary information which allows the WISE system to execute processes correctly with respect to both concurrency control and recovery simultaneously, based on one single correctness criterion.

Furthermore, we have also designed special wrappers, called *Transactional Coordination Agents* (TCA), that allow us to provide database functionality on top of application systems even if these applications do not support it. This database functionality like the atomicity of local operations, the support of a Two Phase Commit protocol (2PC) [6], or the compensation of previously executed operations is necessary to support the execution guarantees for processes denoted by the spheres of atomicity and the spheres of isolation. These TCAs have proven to be very useful in practice and have added a significant degree of failure resilience to the system.

6 Conclusions

Trading communities, virtual enterprises and virtual business processes are important notions in electronic commerce. The question is how to support these abstractions in a coherent manner. In this paper we have argued that virtual business processes can be interpreted and treated as persistent, coarse grained programming languages for distributed applications. Doing so allows to discuss the necessary functionality based on a common abstraction instead of looking at isolated problems. To prove our argument we have developed an specification for a system supporting electronic commerce processes and show how this specification can be implemented in practice using the WISE project as an example. We have also extensively discussed key functionality that needs to be implemented as part of the system: advanced language features, execution guarantees, Quality of Service, etc. While we do not expect all forms of electronic commerce to be expressed in the form of processes, we believe there are many application scenarios which could greatly benefit from the ideas here described.

Acknowledgments

WISE is a research project funded by the Swiss national Science Foundation under the thematic program "Information and Communication Systems". We would like to thank all the partners in the WISE project for their helpful comments and ideas. We would also like to thank all those students who have helped or are currently helping in the development of WISE: W. Bausch, C. Cañamero, S. Middendorf, A. Näf, F. Nussberger, D. Seitz, O. Separovic, K. Stöckli, M. Vinzens, and A. Weiss. More information about WISE can be found in the following URL: http://www.inf.ethz.ch/department/IS/iks/research/wise.html

References

[1] Survey: Electronic commerce. *The Economist*, http://www.economist.com /editorial /freeforall /14-9-97 /index_survey.html (1997).

[2] G. Alonso. Processes + Transactions = Distributed Applications. In *Proceedings of the High Performance Transaction Processing (HPTS) Workshop at Asilomar, California, September 14-17th, 1997.*, Also available in MiddlewareSpectra, vol. 11, no. 4, Spectrum Reports Inc./Ltd. and Financial MiddlewareSpectra, vol. 11, no. 4., Spectrum Reports Inc./Ltd (www.middlewarespectra.com). (1997).

[3] 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.* (1999).

[4] 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 (1997).

[5] G. Alonso and C. Mohan. Workflow management: The next generation of distributed processing tools. In Sushil Jajodia and Larry Kerschberg, editors, *Advanced Transaction Models and Architectures*, chapter 2. Kluwer Academic Publishers (1997).

[6] P.A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison Wesley (1987).

[7] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource Reservation Protocol (RSVP) - Version 1 Functional Specification. RFC 2205 (1997).

[8] L. Camp, M. Harkavy, D. Tygar, and B. Yee. Anonymous Atomic Transactions. In *In Proceedings of the 2nd Usenix Workshop on Electronic Commerce, pages 123 - 133.* (1996).

[9] F. Casati, P. Grefen, B. Pernici, G. Pozzi, and G. Sanchez. WIDE Workflow Model and Architecture. Technical Report 96-19, University of Twente (1996).

[10] Stefano Ceri, Paul W.P.J. Grefen, and Gabriel Sanchez. WIDE: A Distributed Architecture for Workflow Management. In *Proceedings 7th International Workshop on Research Issues in Data Engineering (RIDE'97)*, pp. 76–79, Birmingham, UK (1997).

[11] D. Flanagan. *Java in a Nutshell.* O'Reilly & Associates (1996).

[12] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, **3**(2):119–153 (1995).

[13] Dimitrios Georgakopoulos. Collaboration management infrastructure for comprehensive process and service management, Presentation in International Symposium on Advanced Database Support for Workflow Management, Enschede, The Netherlands (1998).

[14] A. Geppert and D. Tombros. Event-based distributed workflow execution with EVE. Technical Report 96.5, University of Zurich (1996).

[15] J.B. Goodenough. Exception handling: Issues and a proposed notation. *Communications of the ACM*, **18(12)**:683–695 (1975).

[16] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufman (1993).

[17] C. Hagen. Atomarität in Workflow- und Prozessunterstützungssystemen. In *9. GI-Workshop "Grundlagen von Datenbanken"*, Friedrichsbrunn, Germany. In German (1997).

[18] C. Hagen and G. Alonso. Flexible exception handling in the OPERA process support system. In *Proceedings os the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, Amsterdam, The Netherlands (1998).

[19] 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 (1999).

[20] IvyTeam. Structware'98 Process Manager. Available through http://www.ivyteam.com (1998).

[21] S. Jablonski and C. Bussler. *Workflow Management.* International Thomson Computer Press (1996).

[22] 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* (1998).

[23] P. Muth, J. Weissenfels, and G. Weikum. What Workflow Technology Can Do For Electronic Commerce. Technical report, University of the Saarland, Department of Computer Science, Saarbrücken, Germany.

[24] N.W. Paton, O. Diaz, M.H. Williams, J. Campin, A. Dinn, and A. .Jaime. Dimensions of active behaviour. In *Proc. 1st Int. Wshp. on Rules In Database Systems*, pp. 40–57. Springer-Verlag (1994).

[25] H. Schuldt, G. Alonso, and H.-J. Schek. Concurrency Control and Recovery in Transactional Process Management. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS'99)*, Philadelphia, PA (1999).

[26] H. Schuster, S. Jablonski, T. Kirsche, and C. Bussler. A Client/Server Architecture for Distributed Workflow Management Systems. In *Proc. of Third Int'l. Conf. on Parallel and Distributed Information Systems*, Austin, Texas (1994).

[27] G. Sidler, A. Scott, and H. Wolf. Collaborative Browsing in the World Wide Web. In *Proceedings of the 8th Joint European Networking Conference, Edinburgh, Scotland* (1997).

[28] G.L. Steele. *Common Lisp: The Language.* Digital Press, 2 edition (1990).

[29] B. Stroustrup. *The C++ Programming Language.* Addison Wesley, 2 edition (1991).

[30] D. Tygar. Atomicity in Electronic Commerce. *Internet Beseiged, ACM Press and Addison-Wesley. P. Denning and D. Denning, editors.*, pp. 389 – 406 (1997).

[31] J. Widom and S. Ceri. *Active Database Systems.* Morgan Kaufmann Publishers (1996).