IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4218

A Methodology For Development of Trusted Cooperative Information Systems

Maria Grazia Fugini and Pierluigi Plebani
Politecnico di Milano, fugini@elet.polimi.it, plebani@fusberta.elet.polimi.it

ABSTRACT

In building distributed information systems, a methodology for analysis, design and implementation of security requirements of involved data and processes is essential for obtaining mutual trust between cooperating organizations. Moreover, when the information system is built as a cooperative set of e-services, security is related to the type of data, to the sensitivity context of the cooperative processes and to the security characteristics of the communication paradigms. This paper presents a methodology to build a trusted cooperative environment, where data sensitivity parameters and security requirements of processes are taken into account. The phases are illustrated and a reference example is presented in a cooperative information system and e-applications. An architecture for trusted exchange of data in cooperative information system is proposed.

INTRODUCTION

Recently, the widespread use of information technology and the availability of networking services have enabled new types of applications, characterized by several geographically distributed interacting organizations. In particular, the term *Cooperative Information Systems* (*CoopIS*) is used to denote distributed information systems that are employed by users of different organizations under a common goal [Mylopoulos et al. 1997].

In this paper we refer to a recent extension of the concept of *CoopIS*, namely *e-applications* [Mecella et al. 2001], consisting of *e-services* provided by different organizations on the net.

In addition to geographical distribution and inter-organization cooperation, in e-applications some security issues arise, such as (i) cooperating organizations may not know each other in advance and (ii) e-services can be composed both at design and run-time. Whereas in traditional "closed" CIS mutual knowledge and agreements upon design of applications are the basis for the cooperation, the availability of a complex platform for e-services [Mecella et al. 2001] allows for "open" cooperation among different organizations that may not know and/or trust each other.

A major obstacle in securing information systems, and CoopIS in particular, lies in the lack of concepts and methods that allow security developers to identify, design, and implement security requirements policies [Chung et al. 2000]. For CoopIS, few requirements and policies are known at design time: at run time, policies need to be negotiated among the cooperating processes or new policies must be added. In these cases, determining the suitable requirements and policies is based on the identification of the "normal" behavior of the system users [Mukkamala et al., 1999], known as user profiling methods. The need arises to identify, verify, and strengthen the security policies in order to allow the e-services to securely authenticate each other and to exchange data in a trusted way.

This paper proposes a policy-and mechanism-based security methodology where we organize application data and control data of e-services in a *sequence of development steps*. Moreover, an architecture for secure exchange of information among e-services in the CoopIS based on the security level is proposed.

Some proposals for architectures for e-services and workflow-based environments have been presented in the literature [Casati et al. 2001, Mecella et al. 2001]. The concept of *cooperative process* [Schuster et al. 2000] defines as a complex business process involving different organizations. An e-service represents a contract on which an organization involved in the cooperative process agrees.

In order to integrate features of *data security* in a CoopIS/e-application we have to handle:

- security of data handled *within* one e-service/process;
- security of data exchanged *among* cooperating e-services/processes and therefore of the *cooperation* among individually trusted e-services.

The methodology presented in this paper extends the traditional waterfall method by providing a set of steps that, starting from the user security requirements and policies, brings to the development of a *secure*, or *trusted*, CoopIS (we will use one of these terms indifferently).

Working Assumptions

In the remainder of the paper, we make a set of assumptions that are now shortly listed.

1. First, we assume that each e-services has been designed (e.g., as a workflow) according to internal security requirement and policies and is therefore a *trusted* or secure (set of) *process*(es). The focus of the methodology is on data and process security during the *cooperation* among processes.
2. Secondly, we consider that organizations cooperating in CoopIS/e-applications can be of two types:
   - *trusted organizations*: data transmission occurs among organizations which trust each other in a network due to organizational reasons (e.g., homogeneous work groups in a departmental structure, or supply-chain relationships among organizations in a virtual enterprise);
   - *untrusted/external organizations*: data are transmitted among cooperating entities in general, possibly accessing external data sources.

   Every time mutual knowledge among organizations participating in CoopIS/e-applications is not given in advance, mechanisms are needed to ensure that mutual trust be established dynamically, during cooperative process executions.
3. *Trust* regards mainly two aspects:
   - the sensitivity of data being managed within a process, and
   - a secure *information exchange* to guarantee sensitive information.

   1. Sensitivity concerns both correct authentication of cooperating organizations and the process of guaranteeing that only authorized organizations can read, use, and generate data in the cooperative process. To guarantee sensitivity of information, security technologies and mechanisms must be used, e.g., based on the use of digital certificates and signatures, to allow the cooperating organizations to establish a secure communication environment.

The paper is organized as follows. In Section 2, we present the methodology schema through its steps and introduce a reference example. In Section 3, we detail the phases of the methodology. Finally, in Section 4 we draw the final remarks and discuss the envisioned use of the framework in order to cope with development issues aspects that can regard other types of distributed Information Systems, such as Mobile Information Systems or Multi-Channel Information Systems that use different communication channels.

METHODOLOGY SCHEMA

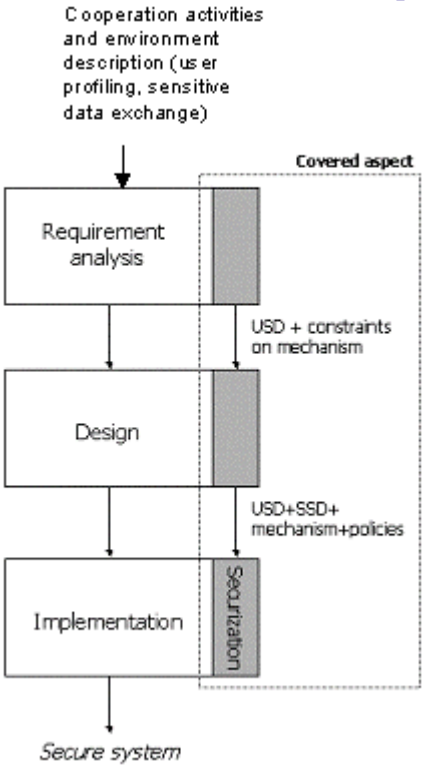
Security requirements and policies are the knowledge of a specific domain about features of interest for the identification and verifica-

This paper appears in *Issues and Trends of Information Technology Management in Contemporary Organizations,* the proceedings of the Information Resources Management Association International Conference.

tion of security issues and mechanisms in the CoopIS. Starting from captured requirements and policies, the *sequence of phases* is identified as follows:

- A *requirements analysis* phase, where the sensitivity of *user data* is stated for those data (mainly databases and business data such as customer directories or collections of documents) pointed out as core-business data for the CoopIS.
- A *design phase*, where *system security* data (such as password, security files, log files, authentication data and so on) are designed and ways for exchanging data in a secure way are identified.
- A *securization* phase, where user data and system security data are implemented on a security platform made of selected products and solutions for authentication, access control, audit, communication, and so on.

Figure 1: Sequence of steps in the methodology



A Reference Example

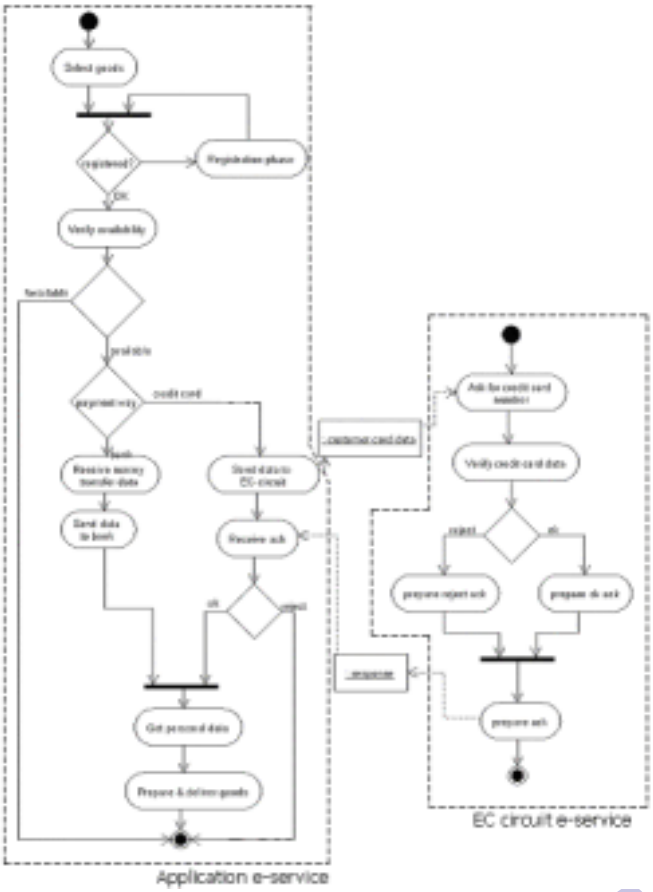
To illustrate the development methodology, in the paper we refer to a "Goods purchase and payment" case study, whose UML activity diagram is shown in Fig. 2. This e-application is composed of two e-services: a good ordering e-service and a payment e-service, which can possibly interact by exchanging the customer card data.

This is a typical e-commerce application whose flow goes as follows. First, the customer browses the product's catalogue published on the company's web site. In order to buy the chosen goods, the customer has to register into the application providing his/her personal data. In this way, the system can identify each customer by using an unique indentifier (called username), and authenticate him/her using a password.

Upon request of goods by a customer, the system verifies the availability of the needed quantity of that good; the customer can then pay in two different ways: 1) by credit card or 2) using his/her own bank's payment circuit.

In the first case, the e-services cooperate as follows. Application control passes form the goods selection e-service to the *e-service* of

Figure 2: UML activity diagram of the "Goods purchase and payment" application



the credit card electronic circuit (EC-Circuit). The EC circuit's web site manages information about the credit cards and has the responsibility for checking the customer data and the card validity. As illustrated in the figure, the EC circuit's web site is activated by the dispatch of the customer data sent by the Goods e-service (this is usually modeled and implemented as a trigger in workflows). In this case, security of exchanged data is an issue handled by our methodology.

If the second payment mode (bank circuit) applies, the payment validity is checked within the same system process and no communication exists among e-services, hence no security issues that are treated by our methodology occur.

Once the payment task has been completed, the system prepares and sends to the customer the invoice and the goods. A set of information regarding users that access to the application are stored in order to authenticate them during the registration phase. They are stored in a profiling database.

THE METHODOLOGY PHASES

The basic idea of the methodology for secure CoopIS development is the following.

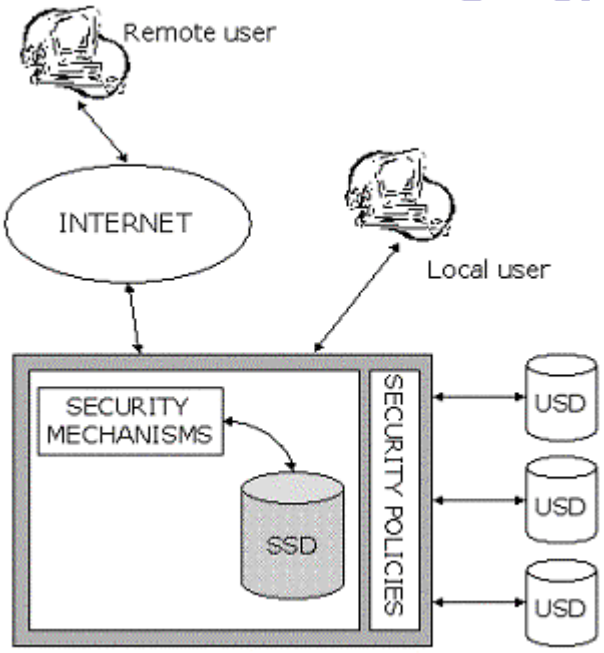
> If the completion of the steps of the methodology eventually leads to trusted agents/roles who handle sensitive data in a correct way (that is, according to the security requirements and policies and according to the implemented security mechanims), then data are handled, processed and exchanged in a secure way and therefore the CoopIS is secure.

This methodology can be applied when:

- a new Information System is built;
- the CoopIS is built out of existing systems that are aggregated to compose a *federated system* and this is the most frequent case;
- part of an existing system (e.g., a new e-services or a new version thereof) is developed.

In the two latter cases, the methodology must take into account the different policies of the existing systems and mediate them into a *common federated security policy* for data protection [Castano et al., 97].

Figure 3: Methodology data, policies and mechanisms



In the methodology, we classify the sensitive data in two classes:

a) User Sensitive Data (USD) - Data that are sensitive for the user (databases, data files, documents, and so on)

b) System Sensitive Data (SSD) - Data that constitute the security mechanisms and hence are sensitive for the system (e.g., password files, crypto keys, protected databases, log files, audit traces) and arise from the choice of the mechanisms that will ensure the protection of the USD.

This classification leads to a *recursive* definition of sensitive data. This recursion consists in the fact that some sensitive data (USD) are protected through other sensitive data (SSD), which are in turn to be protected. To stop the recursion, the methodology has to be based on a set of *security policies* able to protect the sensitive data independently of technological (hardware or software) issues or devices. This is shown in Fig. 3, where the methodology block denoted by a gray background) represents the target of the methodology, which is composed of two sub components: the security mechanisms, that use the SSD, and the policies. In this way, access to the sensible data of USD type occurs only through the gray-background module that will be implemented through the methodology.

We assume that, in a preliminary requirement collection phase, interviews and checklists have been submitted to the users in order to capture the core business of the CoopIS and list the basic categories of data that need to be protected.

For example, in our reference example, a requirement list is shown in Table 1. External payment means payment via EC credit card. Agent means the operator of the e-service. The numeration assigns a letter to the business targets, a number to functional and non functional requirements and gives a "dot" level to the security policies regarding the subtarget and the requirements. A hierarchy in a tree can be constructed form this table to study derived policies. Finally, we have listed some USD and SSD data for this application.

Requirements Analysis Phase

The goals of the requirements analysis phase are:

1) The identification of the *USD*. The emphasis of the study here is on data that need to be protected: indirectly, security information is derived on who (agents or roles or Organization Units) is authorized to operate on these data, from which sites/workplaces, under which policies (e.g. separation of duties, need-to-know, binding of duties [Castano et al., 95]).

2) The identification of *constraints on security mechanism (and therefore on SSD)*: these data will be derived precisely in the design phase. Here only constraints on what the user wants or does not want (e.g., password only instead of digital certificates) are considered. Thanks to the identification of constraints, we can apply the methodology presented in this paper not only to a new system, but also to an existing system, which is represented basically in terms of constraints.

Once the USD have been identified, we classify them according to the four "traditional" *sensitivity properties*:

- integrity
- confidentiality
- non repudiation
- authenticity.

In fact, the need to preserve these properties leads to different levels of security in data protection and the four properties constitute the *Security Level* of both the USD and the SSD data. They are used for secure storage and secure transmission of data.

Referring to the reference example, let us consider the case where we are building a new system which has to communicate with the e-service provided by the EC credit card circuit system. There exists a

Table 1: Sample checklist with requirements and policies used as methodology input

	Core Business Description	Funct.l req.s	Non Funct.l req.s	Security Policies
Target	A.Goods selling via e-commerce	A.1.Separate order from payment phase	A.1. Efficiency: goods delivered within 5 days	A.1.1Customer must be registered in customer db
		A.2.Allow for external and internal payment modes	A.2.Provide fast Web access	A.2.1.Keep audit data regarding both orders and browsing session
				A.2.2.Authentication required between e-services in case of external payment
				A.2.3 Same agent can not handle order and payment
		A.3. Filter accesses	A.3 Keep filtering transparent	A.3.1 Use firewall with web bastion host
	B. Fidelization of customers	B.1.Manage credit line data	B.1 Provide fast access to credit data by agents	B.1.1 Keep secrecy and integrity of credit line data
	C. Expand business	C.1.Examine audit data about browsing	C.1 Limit amount of stored analysis data	C1.1 Keep secrecy of data about customers
				C1.2 Treat data statistically only
USD	Customer database, Order database, OrderHistory database, CreditCard Database, AgentIds			
SSD	Passwords, Clients(browser) authentication data, Agent authentication data			

set of sensible data which are transmitted over the communication channel, say via Internet, and, in order to protect these data, we have to create a secure channel to ensure both the integrity and the confidentiality of the data. The USD of the application could be the *personal data* used by the system to deliver the acquired goods. The Security Level SL of these data is represented by the values of the four items SL=<integrity, confidentiality, authenticity, non repudiation>, where each can assume the Top, High, Confidential, Unrelevant value. Infact, the privacy law requires the use of a set of mechanisms to guarantee the confidentiality and the integrity of the personal data.

Hence, in order to satisfy a given order request forwarded to the application, the step of authentication to the application can rely on the use of the login-password method. This is a typical *application constraint* belonging to those that we have called SSD constraints.

Design Phase

The input of this phase is the set of USD produced in the Requirement Analysis phase and the security constraints regarding the SSD.

In this phase we have to identify the suitable mechanisms that permit not only to protect the USD but also to meet the constraints. On the basis of the adopted mechanisms, a particular set of *security (or control) data* will be used to enforce it: these are the data that we have called SSD.

For example, if we chose to protect some stored data using the Access Control Lists (ACL) mechanism, there will be a list of users for whom access is denied. Only the system administrator can manage those data.

Finally, as done with USD, we have to define a Security Level SL for the SSD too. In particular, for data exchange, the design phase specifies an architecture that ensures security when data are transmitted between e-services. The architecture is composed of two standard virtual devices:

- a SecureSender;
- a SecureReceiver.

that, on the basis of the mutual trust between the communicating e-services and of the Security Level of the exchanged data, wrap the data in a *secure package*.

The presence of this abstract devices allows us to face the security design problems with no need to specify the characteristic of a particular data transmission channel.

Referring to the "Goods purchase and payment" example, in order to protect the USD (the personal data), the mechanism should allow access to these data only from the data owner and from the authorized system personnel.

First of all we have to create a mechanism that authenticates the user who wants to operate with the USD. In order to satisfy the existing SSD constraints, we create this secure mechanism on the basis of the login-password method, so that every login-password pair identifies a *subject*.

After identifying all the *objects* composing the system, we have to define the Access Control Lists for them, so that we can create a rule set able to permit or deny the access to an object by a subject.

In this context, information like login and password are the SSD. In order to explain the role of the policies in our methodology, let us consider the typical situation where, the identified SSD are to be stored into a dedicated protected database which is accessible only by a restricted set of operators, such as the system administrator or the account managers. These users are also *subjects* of the system, that can be authenticated in two different modes:

- Using the same login-password used for the normal system users. In this way, the authentication data related to these operators are located at same Security Level of the other users.
- Using different and stronger authentication methods, like smart-card based or digital certificates based. In this case, we create another set of SSD.

The design phase is completed when the protection of all the identified USD has been covered by a mechanism and when all the SSD are protected by security mechanisms or by a set of security policies.

Let us assume that the authentication phase for the account manager implies the use of a smart-card; in this context a security policy could define the behaviour of both the account manager and the system administrator. For example:

- "the operator must securely store his/her smart card every time he/she leaves the work place";
- "in case of theft/lost, the operator must inform immediately the system administrator to disable the smart-card".

Summarizing, the USD are protected by realizing an ACL that implies the definition of the login password pair (SSD). The SSD are protected using the same method that, now, adopts the smart card approach; therefore another SSD is introduced.

Eventually, all these SSD are protected by a correct behaviour of the user, who has to follow the security policies.

Securization

In recent year, quite a number of system and application software has suffered from several vulnerabilities, such as the buffer overflow vulnerability [Aleph 96], which have been exploited by the attackers to obtain system access privileges. In the securization phase, designed protection issues have to be implemented into a set of security mechanisms that allow one to flexibly meet the security policies-

Under the term "securization" we consider:

- The realization of the mechanisms selected in the design phase. It is important to note that the realization of the security mechanisms is a typical task of the implementation phase specified in the software waterfall model.
- The implementation of the whole software code using the secure programming tecnique [Wheeler 1999], in order to avoid the buffer overflow vulnerabilities as well as all other vulnerabilities deriving from wrong code.

Considering this definition of the securization phase, it is difficult to locate this phase with a specific role into the whole system implementation phase. A continuous cycle of hardware/software implementations and verifications has to be performed in order to meet the non functional requirements of the e-applications.

Moreover, during the implementation, other kinds of issues have to be taken into account, such as the characteristics of the transmission channel, or the authentication method adopted by existing e-services (in case a new service has to be integrated) or the costs and technique needed to securize the transmission channel.

Central to this phase is the implementation of the standard virtual devices *SecureSender* and *SecureReceiver* specified in the design phase. They can be viewed as sort of wrapper that mediate the various security level of the transmitted data among the e-services.

CONCLUDING REMARKS

In this paper, we have outlined the steps of a methodology for designing security in e-applications constituting a Cooperative Information System. Central to the methodology is the identification of User Security Data, that is, of information (records, databases, messages, etc.) that are perceived as sensible by the users. A second featuring element is the identification of System Security Data, which follow from User Security Data. These are expressed as constraints (e.g., a password scheme and a public-key encryption mechanism must be used to protect the data) and rule out the choice of security mechanisms that are both application compliant and cost/budget compliant.

An architecture for secure transmission of data between e-services has been sketched in the paper. More details on these wrapper-based architecture can be found in [Bertolazzi et al. 2001], where we have identified the wrapping techniques for data exchanges among e-services.

This work follows by drawing the functional and non functional specifications of the SecureSender and SecureReceiver. A further step consists obviously in expanding the details of the phases and, more significant, in expressing the security policies, requirements, and mechanisms for Mobile Information Systems. These are distributed informa-

tion systems that forward a user request on different communication channels, depending on the availability of the multichannel communication infrastructure.

REFERENCES

[Aleph 96] Aleph One, *Smashing The Stack For Fun And Profit*, Phrack Magazine Volume Seven, Issue Forty-Nine

[Bertolazzi et al., 2001] P. Bertolazzi, M.G. Fugini, M. Mecella, B. Pernici, P. Plebani, M. Scannapieco, "Supporting Trusted Data Exchanges in Cooperative Information Systems", submitted for publication, July 2001

[Casati et al., 2001] Casati F., Sayal M., Shan M.C., "Developing E-Services for Composing E-Services", Proc. 13th International Conf. on Advanced Information Systems Engineering (CAISE 2001), Interlaken, Switzerland, 2001

[Castano et al., 95] S. Castano, M.G. Fugini, G. Martella, P. Samarati, *Database Security*, Addison-Wesley Int. Publ., 1995

[Chung et al., 2000] Chung C., Gertz M., Levitt K., "Discovery of Multi-Level Security Policies", in Proc. 14th IFIP 11.3 Working Conference on Database Security, 2000

[Mecella et al., 2001] Mecella M., Pernici B., "Designing Wrapper Components for e-Services in Integrating Heterogeneous Systems", VLDB Journal, Special Issue on e-Services, 2001.

[Mylopoulos et al., 1997] Mylopoulos J., Papazoglou M. (eds.), "Cooperative Information Systems" , IEEE Expert Intelligent Systems & Their Applications, vol. 12, no. 5, Sept./Oct. 1997

[Mukkamala et al., 1999] Mukkamala R., Gagnon J., Jajodia S., "Integrating data minining techniques with intrusion detection methods", Proc. 12th IFIP 11.3 Working Conference on Database Security, 1999

[VLDB-TES 2000] Proc. of the 1st VLDB Workshop on Technologies for E-Services (VLDB-TES 2000), Cairo, Egypt, 2000.

[Schuster et al., 2000] Schuster H., Georgakopoulos D., Cichocki A., Baker D., "Modeling and Composing Service-based and Reference Process-based Multi-enterprise Processes", Proc. 12th International Conf. on Advanced Information Systems Engineering (CAISE 2000), Stockholm, Sweden, 2000

[Wheeler 1999] D. A. Wheeler, "*Secure Programming for Linux and Unix HOWTO"*, http://www.dwheeler.com/secure-programs/, 1999