17

A formal security design approach for information exchange in organisations

R. Holbein,
S. Teufel,
K. Bauknecht
Department of Computer Science
Winterthurerstr. 190, 8057 Zurich, Switzerland
+41 - 1 - 257 43 11; {holbein, teufel, bauknecht}@ifi.unizh.ch

Abstract

This paper introduces a formal security design approach for information exchange in organisations. The formal approach provides for automation of a security design method which supports security authorities in the design of individual security models. An individual security model is a fully customised specification of access control information for information exchange within a particular business environment. We introduce transaction based business process models (BPM) and utilise these models for a formal transformation to "need-to-know" authorisations. Therefore, we allocate information from BPMs which can be transformed to access control information and derive a specification of an organisation's individual security model. Our approach provides transparency of security design because the design method ensures that a security model is directly related to the business. Moreover, security effort and costs will be reduced because BPMs must not be specified for security reasons and security design can be automated. BPMs are a result of management activities and therefore, existing resources from a security point of view.

Keywords

Business process models, design method, need-to-know principle, role based access control, security design, security model

1 INTRODUCTION

Information security in organisations must be designed and implemented according to individual needs. Unfortunately, specifying an individual security model which is suitable for implementation with IT security mechanisms is a tremendous and extensive engineering task. Security design effort depends on the organisational environment as well as the complexity of security mechanisms which must be applied in order to implement a security policy, e.g. access control mechanisms. Specification of authorisation structures, for example, is part of security design and may require more than a year of work [Pohl et al., 1993]. We will introduce a formal security design approach for information exchange in organisations in order to automate individual security design. Our approach automates a security design method which allows to

D. L. Spooner et al. (eds.), *Database Security IX*

bridge the gap between development of powerful and complex access control mechanisms and application or handling of these mechanisms within organisations.

When a security architecture is applied in an organisation, security controls must be customised for systematic and flexible application concerning the individual environment. Customisation is an engineering activity which we call *security design*[1]. In this paper, we consider security design concerning the individual definition of global access rights (global authorisation) in organisations.

In our *previous work* we have analysed generic security requirements for information exchange and derived a security framework which consists of security criteria [Holbein, 1994] [Hofmann et al., 1994]. With respect to these criteria a "need-to-know" access control policy aims to restrict access rights to the smallest subset that is necessary to fulfil a user's actual task(s) in an organisation (also called least privileges) [Steinke et al., 1992]. A security policy usually provides an informal description of how security must be achieved. This informal description must be transformed to a formal security model in order to apply security mechanisms, i.e. to automate security policy enforcement [ISO, 1991]. For that reason, we have developed a security design method [Holbein et al., 1996]. Our goal is to automate the design method introduced in which reduces complexity and increases transparency within the *specification of individual security models* [Neumann, 1992].

In this paper, we will introduce a formal approach for automation of the security design method dealing with global authorisation for information exchange. We use transaction based business process models (BPM) to derive an organisation's individual security model for information exchange.

The remainder of this paper is structured as follows: In chapter 2 we give a brief overview on transaction based business process modelling which provides the principles that we will use for security design. However, for further details we refer to [Holbein et al., 1996]. Chapter 3 is the major part of this paper where the formal security design approach will be explained. There are two steps of our security design method that will be considered within a formal approach for automation of security design: a formal transformation of model attributes as well as structuration principles for security information. Finally, we draw some conclusions and provide an outlook for our future research.

2 BUSINESS PROCESS MODELLING

2.1 Business processes and security design for information exchange

Business processes take place in and between organisations. Business processes are directed to an organisation's goals and consist of well defined and coordinated business activities suitable to reach these goals. Information exchange takes place within the business activities and therefore, business processes define the context of information exchange in organisations [Abdallah et al., 1993].

Business process models (BPM) result from management activities where BPMs are of paramount and still increasing importance regarding strategy and business planning [Davenport, 1993]. Consequently, a BPM is a particular model of the organisation. However, it may be specified on different levels of abstraction and aggregation. Granularity of a process

---

[1] The OSI Access Control Framework refers to security design as allocation of security information for an establishment of access control policy representations. It is characterised as "... an engineering design activity that necessarily precedes the other access control activities ..." ([ISO, 1991], p. 6).

description depends on the purpose of the model and the complexity of agents that perform the process steps. The more knowledge about the process, the more details can be included in the process description [Curtis et al., 1992].

We intend to utilise *BPMs for security design* concerning information exchange in organisations. As mentioned above, useability of BPMs for different purposes depends on the specification method. Therefore, we have selected a prominent and well established approach for business process modelling so called transaction based business process modelling which is suitable to support security design for information exchange. This approach combines issues of process and organisation with a special emphasis on interpersonal relationships of agents which are established during task fulfilment. Hence, it provides a wide range of information suitable for security design - *security information*. Consequently, our security design method is based on a process oriented view on the organisation instead of considering static organisational structures.

In summary, BPMs are suitable to support security design and therefore, the application of security mechanisms. Process orientation allows to consider information flow as well as establishment of interpersonal relationships and obligations during task fulfilment for comprehensive security design. Further on, BPMs are suitable to increase transparency of a resulting security model.

2.2 Transaction based business process modelling

A BPM is a process oriented representation of an organisation's business. We propose one central construct for business process modelling. This construct refers to a business transaction and therefore, is called a *business transaction construct* or in short transaction construct. Transaction constructs represent process oriented responsibilities in organisations [Picot, 1994] and integrate the relevant attributes of business processes for comprehensive representation of functions (activities), course of events, organisation and information. Moreover, transaction constructs include task descriptions and interpersonal relationships within business processes.

Transaction based business process modelling is established in practice due to a number of case studies and successful applications, primarily in US companies, over the last ten years [Ferstl et al., 1993b] [Leinberger, 1994] [Medina-Mora et al., 1992] [Scherr, 1993] [Ferstl et al., 1993a]. The work of Winograd and Flores in the mid 80s established a theoretical basis for this approach [Winograd et al., 1986] [Winograd, 1988] [Auramaki et al., 1992].

The basic idea behind the concept of business transactions is to describe a business process as "transaction between organisational entities". A transaction construct represents a *customer-supplier relationship* between two agents in a business transaction. Consequently, a business process is a number of business transactions where agents are responsible for goal oriented task fulfilment in a customer-supplier relationship. Customer and supplier are represented by abstract role descriptions of the responsible agents. Information which must be exchanged between the customer and the supplier is defined in terms of document types and object references. Using transaction constructs, complex business processes can be described as a composition of similar customer-supplier transactions. This is a network of interrelated customer-supplier transactions. The network components consist of identical structures regarding the *basic course of events* because there is one fundamental course of events for every kind of customer-supplier transaction[2] [Medina-Mora et al., 1992] [Scherr, 1993]. The basic course of events within every transaction consists of 4 phases (Figure 1):

---

[2] More traditional approaches which are based on procedures and activities result in arbitrary structures because there is no basic course of events and activities may be combined in any way [Scherr, 1993].

- Preparation: The customer or supplier proposes work to be performed by the supplier.
- Negotiation: The customer and supplier come to an agreement about the work to be performed, i.e. make commitments about the conditions for satisfying the agreement.
- Performance: The supplier performs the work and reports completion.
- Acceptance: The customer evaluates the work in terms of the conditions of satisfaction and declares satisfaction or declines to accept it.

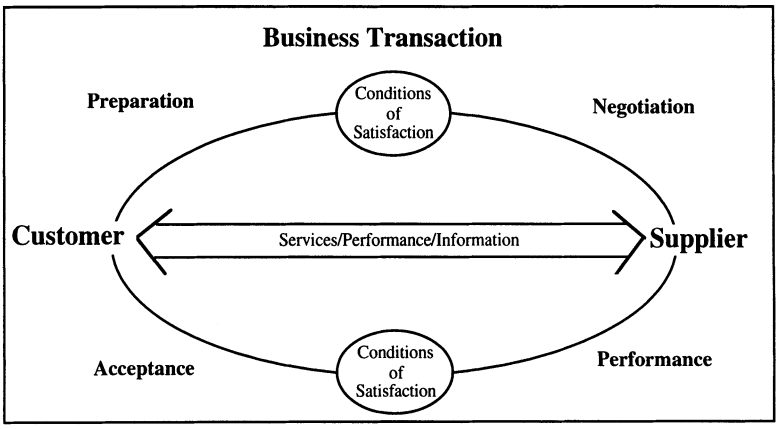

Figure 1 Customer-supplier transaction

This course of events can be refined by additional customer-supplier transactions within each of the transaction phases. Relationships between a transaction and its sub-transactions are represented by transaction links which connect phases of transactions.

In summary, transaction constructs provide extensive information about information exchange in organisations including the business context in which information exchange takes place. Therefore, transaction based BPMs are appropriate to derive extensive security information.

3 SECURITY DESIGN BASED ON BUSINESS TRANSACTIONS

We will now explain *what* kind of security information transaction based BPMs provide and *how* we use this information to build an individual security model. As already mentioned, the transaction construct integrates all relevant aspects of a business process. Thus, security information is either explicitly represented or can be derived from this construct and therefore, is directly related to the business process for transparency and reasoning

Basically, we refer to the transaction construct which was introduced in the previous chapter to transfer attributes from a business process model to a security model. It is a fundamental requirement for business operation that agents hold rights which correspond to their tasks and responsibility within business process steps [Heilmann, 1994]. For that reason, we utilise transaction constructs to specify an agent's "need-to-know" within business transactions and to define access rights which the agent needs in order to fulfil his responsibility (need-to-know policy) [Collins et al., 1993]. This part of the security design process is called a transformation of model attributes. We introduce a formal representation of transaction based business processes which allows this transformation to be automated. As a result, the transformation

defines access control information which represents a "need-to-know" model and allows an implementation of the model with role based access control mechanisms [Ting et al., 1992]. However, we do not consider implementations of the resulting "need-to-know" models with particular role based mechanisms in this paper.

Subsequently, to transformation of model attributes we introduce structuration principles concerning the resulting access control information which allow efficient administration of rights and access control. Extensive structuration depends on the capabilities of a particular access control mechanism. Therefore, our considerations are restricted to structuration principles which can be implemented using basic facilities which are usually provided by role based mechanisms.

3.1 Formal representation of business process models and role based access rights

In this section we will prepare for the transformation of model attributes. Therefore, we use mathematical set theory to introduce a formal representation of *transaction based business process models* and *role based access rights*. First of all, the following basic sets provide a bottom-up definition of transaction based business processes:

A: set of activities;
O: set of objects;
BTP: set of business transaction phases;
BT: set of business transactions;
OE: set of organisational entities.

Additionally, we define a set of documents which is a subset of O:

D: set of documents, where $D \subseteq O$.

Business transaction activity - bta (basic unit for BPM modelling):

In a first step of formal business process representation, we define business transaction activities which are the basic units for specifying business processes. A bta is a tuple consisting of an object $o \in O$, an activity $a \in A$ concerning the object and a business transaction phase btp $\in$ BTP in which the activity takes place. Therefore,

$BTA \subseteq A \times O \times BTP$, consists of elements $bta = (a, o, btp)$[3],

BTA: set of business transaction activities.

A relationship of subordination is used to represent that a business transaction activity may precede another one. Therefore, (BTA, $\leq$) defines a poset (partial ordered set) where $bta_1 \leq bta_2$ denotes that bta_1 takes place prior to bta_2 or at the same time.

Business transaction task - btt:

Second, business transaction tasks accumulate business transaction activities according to particular business transactions. A business transaction task represents the overall activities concerning a business transaction which can be delegated to agents in order to define responsibility for task fulfilment:

$BTT \subseteq P(BTA)$, consists of elements $btt = \{bta_1, ..., bta_n\}$,

BTT: set of business transaction tasks.

---

[3] Note, that document processing is a subset of these business transaction activities because there may also be activities concerning other object types, for example activities which change the state of a business transaction and consequently, a business process. Another example would be physical activities like driving a nail into the wall.

Business transaction responsibility - btr:

A business transaction responsibility is a tuple consisting of one business transaction task and one organisational entity, i.e. an organisational entity is associated with the business transaction task. This association defines which organisational entity is responsible for task fulfilment:

$BTR \subseteq OE \times BTT$, with btr = (oe, btt), BTR: set of business transaction responsibilities.

Business process unit - bpu:

The components above allow to define business process units which correspond to a business transaction construct as introduced in the previous chapter (cf. Figure 1). A business process unit associates two business transaction responsibilities btr_C, $btr_S \in BTR$[4] with a common business transaction. This 3-tuple represents a business transaction with the corresponding customer and supplier as well as their tasks within the transaction. Therefore,

$BPU \subseteq BT \times BTR \times BTR$, consists of elements bpu = (bt, btr_C, btr_S),

BPU: set of business process unit.

There is also a relationship of subordination between business process units to represent that a business transaction may be subtransaction of another one. Therefore, (BPU, <) defines that BPU is a set with a strict partial order where $bpu_1 < bpu_2$ denotes that bpu_1 is subtransaction of bpu_2.

Business process - bp:

Finally, a business process is an accumulation of a particular set of business process units.

$BP \subseteq P(BPU)$, consists of elements $bp = \{bpu_1, ..., bpu_n\}$,

BP: set of business processes, with

$$\forall\, bpu_1, bpu_2 \in BPU : bpu_1 < bpu_2 \wedge bpu_2 \in bp_1 \Rightarrow bpu_1 \in bp_1.$$

Obviously, a BPM is an *abstract representation* of a type of business processes (bp), i.e. a class of business process instances, and therefore requires to specify activities, objects, business transactions, business transaction phases as well as organisational entities on an abstract level. For that reason, we specify these components by the name of a corresponding class representing the overall set of instances belonging to that class. We use activity names (an) to represent classes of activities, object names (on) to represent classes of objects, i.e. object types, business transaction names (btn) to represent classes of business transactions, business transaction phase names (btpn) to represent classes of business transaction phases, role names (rn) to represent classes of organisational entities and business process names (bpn) to represent classes of business processes. Hence, we substitute the basic sets above and use the following basic sets for specifying BPMs on an abstract level:

AN: set of activity names;
ON: set of object names, with
    DN: set of document names, where $DN \subseteq ON$;
BTN: set of business transaction names;
BTPN: set of business transaction phase names, with
    BTPN = {Preparation, Negotiation, Performance, Acceptance};
RN: set of role names;
BPN: set of business process names.

The following representations for classes of business transaction activities, business transaction responsibilities, business process units and overall BPMs result from this substitution:

$BTA \subseteq AN \times ON \times BTPT$, consists of elements bta = (an, on, btpt),

---

[4] Index C, S represent customer and supplier of a business transaction.

$BTR \subseteq RN \times BTT$, with btr = (rn, btt),

$BPU \subseteq BTN \times BTR \times BTR$, consists of elements bpu = (btn, btr_C, btr_S).

$BPM \subseteq P(BPU)$, consists of elements bpm = $\{bpu_1, ..., bpu_n\}$, with

BPM: set of business process models.

A business transaction which is not subtransaction of any other business transaction within a business process, is called the core transaction. This business transaction represents the whole business process on a highest level of aggregation. Therefore, the corresponding business transaction name is interpreted as business process name as well and therefore, a member of the overall set of business process names (BPN):

$$BPN = \{ btn \in BTN \mid \exists bpm \in BPM; btt_C, btt_S \in BTT; rn_C, rn_S \in RN; \neg\exists bpu \in bpm: ((btn, (rn_C, btt_C), (rn_S, btt_S)) \in bpm \wedge (btn, (rn_C, btt_C), (rn_S, btt_S)) < bpu)$$

The formal representation of transaction based BPMs is suitable for a transformation onto role based access rights. In general, *role based access rights* define a relationship between an abstract authorisation unit which represents a set of active entities within a system, a protection object and an operation that can be executed on the protection object. Basically, we use the following formal representation of access rights which consists of tuples according to these constituents:

$AR \subseteq AU \times PO \times OP$, consists of elements ar = (au, po, op), with

AR: set of access rights,

AU: set of authorisation units,

PO: set of protection objects,

OP: set of operations concerning protection objects.

Obviously, a role based specification of authorisation units makes use of abstraction and implies mandatory properties within discretionary access rights. Authorisation units are not restricted to identities of system users, but include roles which represent sets of active entities within a system [Jonscher et al., 1993]. In the following, authorisation units will be allowed to represent any set of active entities within a system. For that reason, we introduce subject expressions to specify authorisation units of different type referring to single active entities as well as sets of active entities within a system. Subject expressions provide means of abstraction as well as aggregation and refer to any active entity of a system which is assigned to this abstract expression at a particular time. We even allow authorisation units to be tuples of subject expressions (combined subjects) which refer to any active entity of a system that is assigned to both abstract expressions at a particular time. We define a basic set of subject expressions (SE) to represent elements of AU within transaction based security design:

SE: set of subject expressions.

Further more, an abstract specification of access rights requires to specify sets of objects within single access rights. Hence, we will specify elements of PO using protection object expressions which allow to represent object instances as well as object classes which refer to particular sets of protection objects. For transaction based security design we define a basic set of protection object expressions (POE):

POE: set of protection object expressions.

Consequently, authorisation units and protection objects within the formal representation of access rights will be replaced by subject expressions and protection object expressions. Further on, operations will be represented by the name of a corresponding type of activity concerning the protection object and we extend the 3-tuple with an additional component which allows to

define access conditions that must be valid during evaluation of access requests in system operation. The following representation of access rights results:

AR ⊆ SE × POE × OPN × AC, consists of elements ar = (se, poe, opn, ac), with

OPN: set of operation names,

AC: set of access conditions (predicates).

The access right holds true for every active entity within a system which is assigned to the subject expression at a particular time as well as every object of a system which is assigned to the protection object expression.

3.2 Transformation of business process models onto role based access rights

The following *transformation* of BPMs onto role based access rights is based on the fact that transaction constructs provide extensive information to define *global access rights* on an abstract level. BPMs provide for an abstract specification of authorisation units as well as protection objects in terms of subject and protection object expressions. Figure 2 shows a transaction construct. The numbered items (1)-(5) declare BPM information which we will now transform onto a "need-to-know" specification for role based access controls. In order to do so, we specify logical expressions which define role based access rights consisting of

- organisational roles and functional contexts concerning business processes (1),(2),
- document types concerning information exchange within business processes (3),
- activities concerning the document types (4) and,
- transaction states as access conditions that must be valid during access control (5).

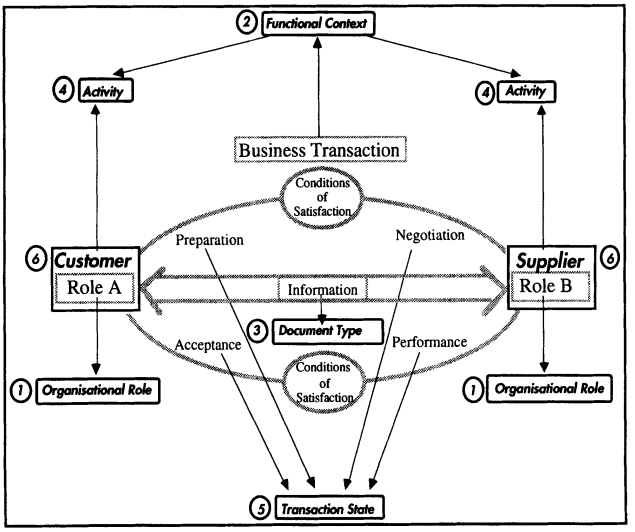

Figure 2 Transformation of attributes from a business transaction construct to a security model

Customer and supplier within a business transaction construct represent a particular set of organisational entities which are specified by role names. These entities are responsible agents

and therefore, need to have access to information objects concerning the business transaction, i.e. have a "need-to-know". Further on, customer and supplier role names will be associated with different types of business transactions and therefore, describe an agent's overall context of activity within an organisation, i.e. a job type. In other words, the overall association of an organisational role and different types of business transactions defines a scope of responsibility consisting of a number of tasks within an organisation. Therefore, customer and supplier roles will be called *organisational roles* (1) within "need-to-know" modelling. Organisational roles can be used as authorisation units in order to specify the overall "need-to-know" access rights concerning all the corresponding business transactions. For transformation of model attributes we use a predicate which allows to define subject expressions according to the customer and supplier role names of the business transaction constructs belonging to a BPM:

$$\text{isSubjectExpression}(rn) \equiv \exists\, bpm \in BPM;\ btn \in BTN;\ btt_C, btt_S \in BTT;\ rn' \in RN :$$
$$((btn, (rn, btt_C), (rn', btt_S)) \in bpm \vee (btn, (rn', btt_C), (rn, btt_S)) \in bpm)$$

The following definition of RN' and SE' describes the overall transformation of BPM organisational roles onto subject expressions which we will use to specify "need-to-know" access rights:

$$RN' = \{\ rn \in RN \mid \text{isSubjectExpression}(rn)\ \}$$
$$SE' = SE \cup RN'$$

For the next transformation step we refer to business transactions as an agent's context of activity from a process oriented point of view. As previously explained, an agent's scope of responsibility consists of tasks which are assigned to business transactions. For that reason, an agent's task related "need-to-know" can be specified by referring to types of business transactions. Consequently, we will use business transaction names to represent a *functional context* (2) within "need-to-know" modelling in order to specify task related "need-to-know" access rights concerning types of business transactions. For transformation of model attributes we redefine the predicate above in order to define subject expressions according to business transaction names:

$$\text{isSubjectExpression}(x) \equiv \exists\, bpm \in BPM;\ btn \in BTN;\ btt_C, btt_S \in BTT;\ rn', rn'' \in RN :$$
$$((btn, (x, btt_C), (rn'', btt_S)) \in bpm \vee (btn, (rn', btt_C), (x, btt_S)) \in bpm \vee$$
$$(x, (rn', btt_C), (rn'', btt_S)) \in bpm)$$

The following definition of BTN' and SE'' describes the overall transformation of BPM business transaction names onto subject expressions which we will use to specify "need-to-know" access rights:

$$BTN' = \{\ btn \in BTN \mid \text{isSubjectExpression}(btn)\ \}$$
$$SE'' = SE' \cup BTN'$$

The final step in defining subject expressions deals with combining the previous ones to define combined subject expressions according to business transaction constructs. For transformation of model attributes we use a predicate which allows to define subject expressions by a combination of an organisational role and a functional context from the business transaction construct belonging to a BPM:

$$\text{isCombSubjExpr}(rn, btn) \equiv \exists\, bpm \in BPM;\ btn \in BTN;\ btt_C, btt_S \in BTT;\ rn' \in RN :$$
$$((btn, (rn, btt_C), (rn', btt_S)) \in bpm \vee (btn, (rn', btt_C), (rn, btt_S)) \in bpm)$$

The following definition of CSE and SE''' describes the overall transformation of BPM organisational role - functional context combinations onto combined subject expressions which we will use to specify "need-to-know" access rights:

$CSE = \{ (rn, btn) \in RN \times BTN \mid isCombSubjExpr(rn, btn) \}$

$SE''' = SE'' \cup CSE$

For further transformation of model attributes we consider the information which agents exchange during a business transaction. Information units are explicitly specified within transaction constructs as part of the agents' task descriptions. Agents of a business transaction communicate these information units as sender-initiated information exchange. Additionally, information units for receiver-initiated information exchange across business transactions or even across business processes (e.g. information retrieval: access to a shared databases) are specified as part of the agents' task descriptions. Hence, these information units will be defined as protection objects. For an abstract representation, the information units are specified by *document types* which define a classification scheme (3). These document types define the protection object classes concerning a business transaction construct, e.g. "Contract". For transformation of model attributes we use a predicate which allows to define protection object expressions according to document types within the business transactions constructs belonging to a BPM:

$isProtectionObjectExpr(on) \equiv$

$\exists\, bpm \in BPM; btn \in BTN; btt_C, btt_S \in BTT; rn_C, rn_S \in RN; an \in AN; btpn \in BTPN :$

$(on \in DN \wedge (btn, (rn_C, btt_C), (rn_S, btt_S)) \in bpm \wedge (an, on, btpn) \in (btt_C \cup btt_S))$[5]

The following definition of ON' and POE' describes the overall transformation of BPM document types onto protection object expressions which we will use to specify "need-to-know" access rights:

$ON' = \{ on \in ON \mid isProtectionObjectExpr(on) \}$

$POE' = POE \cup ON'$

A business transaction construct refers to two agents. These agents are responsible for different tasks within the business transaction. With respect to business transactions, the agents' task responsibilities also define process responsibilities in an organisation [Picot, 1994]. A description of the agents' task fulfilment consists of business transaction activities. These *activities* (4) specify operations concerning objects that will be necessary to accomplish the tasks. Again, it is important to note, that transaction based BPMs are not restricted to information processes. Hence activities may concern every kind of object even physical objects. Nonetheless, for IT security reasons we will focus on activities concerning information objects. These activities define the document processing operations that must be performed and we will interpret these operations as interface description of an object model for document processing [Bever et al., 1988]. Obviously, such a model is partially included within BPMs according to the association of objects and activities within business transaction activities. For that reason, the transformation of model attributes will result in a specification of access rights which reflect parts of a document processing model. However, a complete object model for document processing including a hierarchy of object classes will not be provided. For transformation of model attributes we use a predicate which allows to define operation names according to document processing activities within the business transactions constructs belonging to a BPM:

$isOpName(an) \equiv$

$\exists\, bpm \in BPM; btn \in BTN; btt_C, btt_S \in BTT; rn_C, rn_S \in RN; on \in DN; btpn \in BTPN:$

$((btn, (rn_C, btt_C), (rn_S, btt_S)) \in bpm \wedge (an, on, btpn) \in (btt_C \cup btt_S))$

---

[5] $on \in DN$ is used to select those elements of BTA which refer to document processing activities. This is necessary to restrict specification of access rights to document processing operations, i.e. to exclude physical activities like driving a nail into the wall where no access rights concerning information objects will be specified.

The following definition of AN' and OPN' describes the overall transformation of BPM document processing activities onto operation names which we will use to specify "need-to-know" access rights:

$AN' = \{ an \in AN \mid isOpName(an) \}$

$OPN' = OPN \cup AN'$

Finally, we will use *transaction states* (5) to define access conditions (restrictions) within the specification of access rights. Up to now, we have defined authorisation units according to the agents' scope of responsibility (organisational role) as well as their different tasks concerning business transactions (functional contexts). In general, a task related specification of access rights may be done on different levels of detail. If a detailed task specification is available, it will be possible to specify access control conditions where single activities are directly associated with transaction states or even the content or processing state of a document. It is important to note that relationships between activities and transaction states provide more accurate specifications of an agent's "need-to-know". Access conditions will be expressed in terms of logical expressions. We introduce a partial function called state to identify elements of BTPN according to elements of BTN: state : BTN ~> BTPN and a predicate to define access conditions according to the state of business transactions within the business transaction constructs belonging to a BPM:

isAccessCondition(btn, btpn) ≡

$\exists$ bpm $\in$ BPM; btt_C, $btt_S \in$ BTT; rn_C, $rn_S \in$ RN; an $\in$ AN; on $\in$ DN; btpn $\in$ BTPN:

$((btn, (rn_C, btt_C), (rn_S, btt_S)) \in bpm \wedge (an, on, btpn) \in (btt_C \cup btt_S) \wedge$

state(btn) = btpn)

The following definition of AC^{BPM} and AC' describes the overall transformation of BPM transaction states onto access conditions which we will use to specify "need-to-know" access rights:

$AC^{BPM} = \{ (btn, btpn) \in BTN \times BTPN \mid isAccessCondition(btn, btpn) \}$

$AC' = AC \cup AC^{BPM}$

Now, we can define "need-to-know" access rights using the components which have been defined during the previous transformation steps, i.e. we define who needs what kind of access to which information object for what purpose. We use the *organisational role* and *functional context* definitions as subject expressions which represent abstract authorisation units referring to the access context of active entities within a system. A combination of an organisational role and a functional context is necessary to represent the agents' task related context within business transactions, i.e. who and what purpose. Therefore, we combine the corresponding subject expressions with respect to our previous definition of combined subject expressions. The further components of "need-to-know" access rights will be *document types* which define protection object classes, i.e. which information, *activities* concerning document processing define what kind of operation and *transaction states* define access conditions. For transformation of model attributes we use a predicate which allows to define access rights according to business transaction activities within the business transactions constructs belonging to a BPM:

isAccessRight((se_1, se_2), poe, opn, ac) ≡

$\exists$ bpm $\in$ BPM; btn $\in$ BTN; btt_C, $btt_S \in$ BTT; rn_C, $rn_S \in$ RN; on $\in$ DN; an $\in$ DN;

btpn $\in$ BTPN:

$((btn, (rn_C, btt_C), (rn_S, btt_S)) \in bpm \wedge (an, on, btpn) \in (btt_C \cup btt_S) \wedge$

$(se_1 = rn_C \wedge isSubjectExpression(rn_C) \vee se_1 = rn_S \wedge isSubjectExpression(rn_S)) \wedge$
$s_2 = btn \wedge isSubjectExpression(btn) \wedge poe = on \wedge isProtectionObjectExpr(on) \wedge$
$opn = an \wedge isOpName(an) \wedge ac = (btn, btpn) \wedge isAccessCondition(btn, btpn))$

The following definition of AR^{BPM} and AR' describes the overall transformation of BPM business transaction activities onto "need-to-know" access rights:

$$AR^{BPM} = \{ ((se_1, se_2), poe, opn, ac) \in SE'' \times POE' \times OPN' \times AC' \mid isAccessRight((se_1, se_2), poe, opn, ac) \}$$

$$AR' = AR \cup AR^{BPM}$$

Up to now, we have specified "need-to-know" access rights according to business transaction activities independent from the BPM in which these business transaction activities occur. Usually, BPMs consist of several business transaction constructs (Figure 3) and there may be multiple BPMs including the same business transaction constructs, i.e. the same types of business transactions. In this case, business process names which refer to the type of business process as a whole, i.e. the name of the core transaction, can be used within access conditions for further refinement of a process related "need-to-know". In general, these logical expressions are means for refinement of access conditions according to individual constraints.

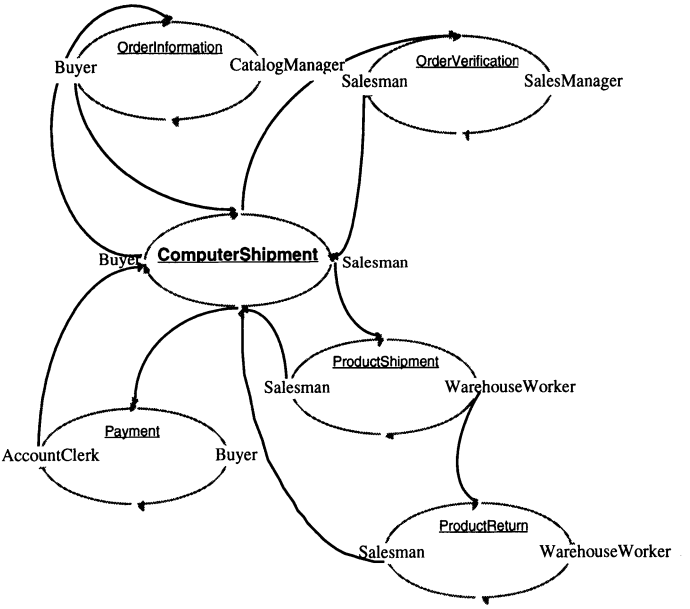

Figure 3 Business process example

3.3 Structuration of security information

Role based access rights which correspond to the "Need-to-know" of agents within an organisation define least privileges which these agents need in order to fulfil their tasks and therefore, provide the foundation of an individual security model. The previously described transformation of model attributes results in an explicit specification of generic rights.

Unfortunately, for complex business process models the number of explicitly specified rights will be quite large. One goal of role based access controls is to reduce the number of explicit rights in order to reduce administration effort and to increase efficiency during evaluation of access requests. It depends on the access control mechanism and its particular capabilities for implicit and structured representation of rights, whether the number of explicit rights can be reduced. In the following we will introduce some basic structuration principles for the previously defined security information. This structuration requires aggregation and classification capabilities within access control mechanisms to define:

- hierarchical relationships between authorisation units and top-down inheritance of rights,
- aggregations of authorisation units and protection objects which can be nested (definition of sets and subsets concerning authorisation units and protection objects),
- aggregation of operations (definition of functional classes).

Such capabilities imply additional mandatory properties within discretionary access rights because groups of authorisation units and objects will be included within single access rights and administration of rights does not require independent manipulation of a large number of rights. Transaction based security design supports *classification and aggregation of security information* according to generic characteristics of the transaction based security design approach as well as individual characteristics of the business environment under consideration:

(1) Authorisation units: The business transaction construct provides structural properties for a classification of authorisation units (see the corresponding components of the model in Figure 2). We have defined subject expressions according to organisational roles (1) as well as the functional context (2) which are different points of view on an agent's context of activity. Role based access rights have been introduced for an abstract specification of authorisation units according to an agent's access context. Therefore, we use the word role concerning the organisational as well as the functional context, i.e. a subject expression which refers to a functional context will be called functional role in the following. For structuration of the security information, we will first of all refer to the previously defined subsets of SE'': RN', BTN' which separate different types of authorisation units. We will call these subsets OrganisationalRoles and FunctionalRoles in the following.

For further structuration of authorisation units we consider the following relationship: Subject expressions have been defined according to the organisational role of an agent, i.e. an agent's overall scope of responsibility, or according to a functional context which is one part of an agent's scope of responsibility. Hence, each element of OrganisationalRoles defines a subset within FunctionalRoles which represents a scope of responsibility from an organisational point of view. Therefore, we introduce a partial function to identify subsets of FunctionalRoles by elements of OrganisationalRoles:

$$f_1\text{: } RN' \sim\!> P(BTN')$$

$$rn \rightarrow \{\ btn \in BTN' \mid \exists\ bpm \in BPM;\ btt_C, btt_S \in BTT;\ rn' \in RN:$$

$$((btn, (rn, btt_C), (rn', btt_S)) \in bpm \vee (btn, (rn', btt_C), (rn, btt_S)) \in bpm)\ \}$$

Moreover, a business process unit, i.e. a whole transaction construct, also defines subsets within OrganisationalRoles as well as FunctionalRoles. This is because two organisational roles are associated with a common business process unit and because business processes consist of multi-layered business process units, i.e. business transactions which are refined by secondary business transactions and so on ($bpu_i < bpu_k$). First, we will represent the relationships between organisational roles and business transactions by subsets within OrganisationalRoles referring to common business transactions. The following partial function identifies these subsets of OrganisationalRoles:

f_2: BPU ~> P(RN')
bpu → { rn ∈ RN' | ∃ bpm ∈ BPM; btn ∈ BTN; btt_C, btt_S ∈ BTT; rn' ∈ RN:
(bpu ∈ bpm ∧
((btn, (rn, btt_C), (rn', btt_S)) = bpu ∨ (btn, (rn', btt_C), (rn, btt_S)) = bpu)) }

Second, we represent the subordination relationships between business process units by nested subsets within OrganisationalRoles. The following partial function identifies these subsets of OrganisationalRoles:

f_3: BPU ~> P(RN')
bpu → { rn ∈ RN' | ∃ bpm ∈ BPM; btn ∈ BTN; btt_C, btt_S ∈ BTT; rn' ∈ RN:
(bpu ∈ bpm ∧
((btn, (rn, btt_C), (rn', btt_S)) = bpu ∨ (btn, (rn', btt_C), (rn, btt_S)) = bpu) ∨
(btn, (rn, btt_C), (rn', btt_S)) < bpu ∨ (btn, (rn', btt_C), (rn, btt_S)) < bpu)) },
with ∀ bpu_1, bpu_2 ∈ BPU : bpu_1 < bpu_2 ⇒ $f_3(bpu_1) \subset f_3(bpu_2)$

Third, we represent the subordination relationships between business process units by nested subsets within FunctionalRoles. The following partial function identifies these subsets of subject expressions:

f_4: BPU ~> P(BTN')
bpu → { btn ∈ BTN' | ∃ bpm ∈ BPM; btn ∈ BTN; btt_C, btt_S ∈ BTT; rn_C, rn_S ∈ RN:
(bpu ∈ bpm ∧
((btn, (rn_C, btt_C), (rn_S, btt_S)) = bpu ∨ (btn, (rn_C, btt_C), (rn_S, btt_S)) < bpu)) },
with ∀ bpu_1, bpu_2 ∈ BPU : bpu_1 < bpu_2 ⇒ $f_4(bpu_1) \subset f_4(bpu_2)$

Combined authorisation units have been introduced to specify task related access rights concerning business transactions. In the following, we interpret customer and supplier as meta-roles concerning business transaction constructs (see (6) in Figure 2) which classify combined authorisation units. In order to group combined authorisation units according to this classification as a tuple consisting of OrganisationalRoles × FunctionalRoles we define generic subsets of SE using the following predicates:

isCustomer(rn, btn) ≡
∃ bpm ∈ BPM; btt_C, btt_S ∈ BTT; rn_S ∈ RN:
((btn, (rn, btt_C), (rn_S, btt_S)) ∈ bpm ∧ isSubjectExpression(rn) ∧
isSubjectExpression(btn))
isSupplier(rn, btn) ≡
∃ bpm ∈ BPM; btt_C, btt_S ∈ BTT; rn_C ∈ RN:
((rn_C, btt_C), (rn, btt_S)) ∈ bpm ∧ isSubjectExpression(rn) ∧
isSubjectExpression(btn))
Customers = { (rn, btn) ∈ RN' × BTN' | isCustomer(rn, btn) }
Suppliers = { (rn, btn) ∈ RN' × BTN' | isSupplier(rn, btn) }

Consequently, combined authorisation units consist of an organisational as well as a functional point of view concerning an agent's context of activity and represent either the customer or the supplier within a business transaction. Now we extend SE''' with the previously defined subsets of authorisation units:

$$SE'''' = SE''' \cup \{ \times \in P(SE''') \mid \exists\, rn \in RN;\ bpu \in BPU :$$
$$x = f_1(rn) \vee x = f_2(bpu) \vee x = f_3(bpu) \vee x = f_4(bpu) \} \cup$$
$$\text{Customers} \cup \text{Suppliers}$$

(2) Hierarchical relationships between authorisation units: We will now introduce a 'special kind of' hierarchical relationship between authorisation units which can be derived from the transaction model. This hierarchy will be established between combined subject expressions according to relationships between the corresponding business process units of a BPM. The hierarchy refines an organisational hierarchy and provides for *"need-to-know" inheritance of rights*. Nonetheless, an organisational hierarchy must be defined and will usually be available in an organisation, e.g. the manager of a sales department is supervisor of a salesman. We define a poset (RN, ≤) with $x < y : (x \leq y \wedge x \neq y)$, to represent this organisational hierarchy where $rn_1 < rn_2$ denotes that rn_1 is subordinated to rn_2. In order to specify hierarchical relationships between authorisation units we define (SE'''', <) to be a set with a strict partial order.

However, we do not inherit rights along an organisational hierarchy within transaction based security design because we aim to realise a "need-to-know" policy according to the business processes within organisations, i.e. relationships between task responsibilities according to business processes[6] [Wilke, 1992]. Hence, we define hierarchical relationships between combined subject expression to establish a task related inheritance of rights concerning "need-to-know" within business processes, i.e. inheritance depends on the organisational hierarchy as well as task relationships within business processes. Task related authorisation concerning business processes has been realised by specifying authorisation units as combination of subject expressions representing an organisational role and a functional context: $(s_1, s_2) \in SE''''$, with isCombSubjExpr(s_1, s_2). Now, we define the following condition for a "need-to-know" inheritance of access rights between such authorisation units: authorisation units inherit rights from other authorisation units if they are represented by a combined subject expression where one organisational role of the combined subject expression is higher in organisational hierarchy than the other one and a subtransaction relationship between the functional context components of the subject expressions exists. The following expression describes this "need-to-know" hierarchy of combined subject expressions:

$$\forall\, x, y \in SE'''' : x < y \Leftrightarrow$$
$$\exists\, rn_1, rn_2 \in RN';\ btn_1, btn_2 \in BTN':$$
$$(x = (rn_1, btn_1) \wedge y = (rn_2, btn_2) \wedge rn_1 < rn_2 \wedge (btn_1 < btn_2 \vee btn_2 < btn_1))$$

The following expression describes the implication of a hierarchical relationship between authorisation units concerning inheritance of access rights:

$$\forall\, s_1, s_2 \in SE'''';\ poe \in POE;\ opn \in OPN;\ ac \in AC :$$
$$((s_1, poe, opn, ac) \in AR' \wedge s_1 < s_2 \Rightarrow (s_2, poe, opn, ac) \in AR').$$

(3) Protection objects: Business transaction constructs define groups of information units with common "need-to-know" characteristics, i.e. a number of access rights may refer to identical authorisation units, operations and access conditions but different document types (protection objects) within one business transaction construct. For structuration of security information, we define sets of protection objects according to the business transaction constructs of a BPM, i.e. group protection objects according to the functional context in which the objects are used. The following partial function identifies these subsets of protection objects

---

[6] Probably, agents may have general rights which do not depend on a "need-to-know" concerning task fulfilment within business transactions. Such rights are not for further consideration here, but may be directly assigned to organisational roles and inherited along the organisational hierarchy.

which we call transaction folders. Business transaction folders are additional protection object expressions which represent all the objects belonging to the set:

f_5: BPU ~> P(ON')
bpu → { on ∈ ON' |
∃ bpm ∈ BPM; btn ∈ BTN; btt_C, btt_S ∈ BTT; rn_C, rn_S ∈ RN; an ∈ AN;
btpn ∈ BTPN:
(bpu ∈ bpm ∧
((btn, (rn_C, btt_C), (rn_S, btt_S)) = bpu ∧ (an, on, btpn) ∈ (btt_C ∪ btt_S))) }

Further on, we define subset relationships among those sets of protection objects which refer to business process units with a subordination relationship. As a result there are nested transaction folders according to subordination relationships between business transactions constructs. The following partial function identifies these subsets of protection objects:

f_6: BPU ~> P(ON')
bpu → { on ∈ ON' |
∃ bpm ∈ BPM; btn ∈ BTN; btt_C, btt_S ∈ BTT; rn_C, rn_S ∈ RN; an ∈ AN;
btpn ∈ BTPN:
(bpu ∈ bpm ∧
((btn, (rn_C, btt_C), (rn_S, btt_S)) = bpu ∧ (an, on, btpn) ∈ (btt_C ∪ btt_S) ∨
(btn, (rn_C, btt_C), (rn_S, btt_S)) < bpu ∧ (an, on, btpn) ∈ (btt_C ∪ btt_S))) }

Now we extend POE' with the previously defined subsets of protection objects:

POE'' = POE' ∪ { x ∈ P(POE') | ∃ bpu ∈ BPU : x = f_5(bpu) ∨ x = f_6(bpu) }

(4) Functional classes: Functional classes allow to group operations concerning documents. As already explained, these operations depend on the interface description of an object model for document processing. Functional classes must be customised to combine rights with document processing operations which refer to the same protection object (document) and are 'typically' (often) granted together for one authorisation unit, i.e. establish a scope of activity concerning document processing. Consequently, functional classes must be defined individually and manually. This may be done by analysing the task descriptions of business transaction constructs belonging to BPMs in order to identify scopes of activity. Otherwise, the explicitly specified access rights which have been derived from the BPMs must be analysed concerning different operations which are often granted for identical authorisation units and protection objects.

The structuration principles introduced in this section allow to reduce *the number of rights* that must be explicitly specified within a "need-to-know" security model and therefore, allow to reduce the effort for administration of rights and access control. A basic set of explicitly specified rights AR' results from the transformation of model attributes. Application of the structuration principles for specification of access rights requires the identification of common characteristics within that basic set of explicit rights. In order to do so, explicitly specified access rights within AR' must be grouped to identify rights which have identical components except of one. Subsequently, the distinct components must be compared with the various subsets of access right components that have been defined by the previous structuration principles. If the distinct components match with one of these subsets, then an AR'' can be defined where the identified set of explicitly specified access rights can be substituted by a single access right which includes the whole set of matching access right components. Obviously, matching and substitution can be formally specified and therefore, may also be

automated. With respect to the different access right components this substitution will have different results depending on the sequence in which the access right components are considered. Therefore, the sequence of substitution which is best suited depends on the strategy for administration of rights and evaluation of access requests of a particular role based access control system which will be used to implement the "need-to-know" security model.

For example, a certain number of rights may grant generation of an object type "Contract" (opn = generate, poe = Contract) to all the members of Suppliers. These rights can be substituted with one access right including se = Suppliers. If multiple access rights are identical, except the protection object components and these protection object components are all members of a particular transaction folder, then a single access right can be specified referring to the entire transaction folder.

As a result of the whole security design process we have defined security information: SE"", POE", OPN', AC', AR" which can be used to initialise role based access control systems, i.e. to implement the "need-to-know" security model for individual system operation with a "need-to-know" access control policy. Moreover, the relationships which are established by the previous structuration principles allow to verify *consistency of a security model* concerning the principles of transaction based security design:

1. Organisational roles have been used to define subsets of FunctionalRoles. According to the transaction construct, every element of FunctionalRoles must be member of different subsets of FunctionalRoles according to the organisational roles of the transaction construct.
2. Combined subject expressions must either represent a customer or a supplier and therefore, must be assigned to the corresponding generic subset of SE"".

4 CONCLUSION AND FURTHER WORK

In this paper we have introduced a formal security design approach for information exchange. The overall goal of this approach is to automate individual security design in order to support security authorities in the design of "need-to-know" security models and therefore, to reduce an organisation's security related effort. Our approach is founded on transaction based business process models and consists of a

- transformation of formal model attributes (section 3.2) and
- structuration of the transformed security information (section 3.3)

The resulting security model specifies the "need-to-know" of agents within business processes and can be used to initialise role based access control systems, i.e. to implement the "need-to-know" security model for individual system operation with a "need-to-know" access control policy. Transaction based BPMs are suitable to reduce security design effort because these models are existing resources from a security point of view. Transparency of the security design process results from the direct relationship between a security model and the BPMs of an organisation. Moreover, security design is more efficient and bridges the gap between development of complex security mechanisms and customisation of such mechanisms for individual application in a business environment.

Actually, we are evaluating the security design method within case studies and we are working on a prototype implementation of a security design environment with automated transformation of BPMs onto access controls. This prototype will be implemented with Ingres Windows4GL™. Based on the ActionWorkflow™ business process modelling tools from ActionTechnologies we automatically transform model attributes to a "need-to-know" representation with Argos role based access control mechanisms [Jonscher et al., 1993].

REFERENCES

Abdallah T. S., Holbein R., Scheidegger P., Schmidt S. (1993): Offene Bürokommunikation - Inner- und zwischenbetrieblicher Informationsaustausch, Institut für Informatik, Institutsbericht, Initialpapier der Projekte OBI/OBS, Nr. 93.33.

Auramaki E., Hirschheim R., Lyytinen K. (1992): "Modelling offices through discourse analysis: the SAMPO approach", Computer Journal, Vol. vol. 35, nr. 4, S. 342-352.

Bever M., Ruland D. (1988): "Aggregation and Generalisation Hierarchies in Office Automation", in: Allen R. B. (Hrsg.): Conference on Office Information Systems, Palo Alto, California, ACM Press, S. 250-264.

Collins B. S., Mathews S. (1993): "Securing Your Business Process", Computers & Security, Vol. Vol. 12, No.7, S. 629-633.

Curtis B., Kellner M. I., Over J. (1992): "Process Modeling", Communications of the ACM, Vol. Vol. 35, No. 9, S. 75-90.

Davenport T. H. (1993): Process Innovation - Reengineering Work through Information Technology, Harvard Business School Press, Boston.

Ferstl O. K., Sinz E. J. (1993a): Der Modellierungsansatz des Semantischen Objektmodells (SOM), Universität Bamberg, Bamberger Beiträge zur Wirtschaftsinformatik, Nr. Nr. 18.

Ferstl O. K., Sinz E. J. (1993b): "Geschäftsprozessmodellierung", Wirtschaftsinformatik, Vol. 6/93, S. 589-592.

Heilmann H. (1994): "Workflow Management: Integration von Organisation und Informationsverarbeitung", Theorie und Praxis der Wirtschaftsinformatik HMD, Vol. 31, Nr. 176, S. 8-21.

Hofmann H. F., Holbein R. (1994): "Reaching out for Quality: Considering Security Requirements in the Design of Information Systems", in: International Conference on Advanced Information System Engineering CAiSE*94, Utrecht, Netherlands, S. 105-118.

Holbein R. (1994): "Secure Information Exchange in Organisations", in: IFIP TC11 Tenth International Conference on Information Security SEC'94, Curacao, electronically published on the IFIP TC11 WWW home page: http://www.iaik.tu-graz.ac.at/tc11_hom.html.

Holbein R., Teufel S., Bauknecht K. (1996): "The Use Of Business Process Models For Security Design in Organisations", in: submitted to IFIP SEC96 TC 11 Twelfth International Conference on Information Security, Samos, Greece.

ISO (1991): ISO 10181-3: Information technology - Open Systems Interconnection - Security frameworks in open systems - Part 3: Access Control, International Organisation for Standardization ISO, DIS, Nr. ISO/IEC DIS 10181-3.

Jonscher D., Dittrich K. R. (1993): A Formal Security Model Based on an Object-Oriented Data Model, University of Zurich, Department of Computer Science, Nr. 93.41.

Leinberger A. (1994): "Workflow-Managementkonzept von Action Technologies - Der Motor aller Arbeitsprozesse heisst Zufriedenheit", Office Management, Nr. 11/94, S. 70.

Medina-Mora R., Winograd T., Flores R., Flores F. (1992): "The Action Workflow Approach to Workflow Management Technology", in: Proceeding of the ACM Conference on Computer Supported Cooperative Work, Toronto, S. 281-288.

Neumann P. (1992): "Trusted Systems", in: Jackson K. M., Hruska J., Parker D. B. (Hrsg.): Computer Security Reference Book, Butterworth-Heinemann Ltd, Oxford, S. 837-862.

Picot A. (1994): "Restrukturierung von Unternehmen und Beschäftigungsperspektiven", Office Management, Nr. 11/94, S. 10-14.

Pohl H., Weck G. (1993): "Stand und Zukunft der Informationssicherheit", Datenschutz und Datensicherung, Vol. 1 und 2/93, S. 18-22; 78-86.

Scherr A. L. (1993): "A New Approach To Business Processes", IBM Systems Journal, Vol. Vol.32,No.1, 1993, S. 80-98.

Steinke G., Jarke M. (1992): "Support for Security Modeling in Information Systems Design", in: Thuraisingham B. M., Landwehr C. E. (Hrsg.): IFIP WG 11.3 Workshop on Database Security, VI: Status and Prospects, Vancouver, Canada, Elsevier Science Publishers B.V., S. 125-141.

Ting T. C., Demurjian S. A., Hu M.-Y. (1992): "A Specification Methodolgy for User-Role Based Security in an Object-Oriented Design Model", in: IFIP WG 11.3 Sixth Working Conference on Database Security, Simon Fraser University Burnaby, Vancouver, British Columbia, S. 351-378.

Wilke H. R. (1992): "Zugriffsschutz in UNIX-gestützten Bürokommunikationssystemen mit UniDesk - Lösungen und Erfahrungen aus Anwendungsprojekten in Behörden", in: Lippold H., Schmitz P. (Hrsg.): BIFOA-Kongresses SECUNET 92 - Sicherheit in netzgestützten Informationssystemen, Vieweg, S. 173-181.

Winograd T. (1988): "A Language/Action Perspective on the Design of Cooperative Work", in: Greif R. (Hrsg.): Computer Supported Cooperative Work: A Book of Readings, Morgan Kaufmann Publishers, S. 623-653.

Winograd T., Flores F. (1986): Understanding Computers and Cognition, Ablex Publishing Corp., Norwood, New Jersey.