Chapter 4

Electronic Contracts

This thesis deals with the exchange and processing of rights expressions. Electronic contracts are rights expression with a particular semantic meaning and have importance in terms of their legal effect. As this thesis especially focuses on processing electronic contracts, this chapter addresses the particularities when dealing with electronic contracts.

As soon as parties agree to exchange digital or physical goods, a contract is concluded.

A contract is an agreement of two or more parties, i.e. a two or multilateral declaration of intent on the exchange of rights to goods or services under certain terms and conditions. The memorandum of an agreement is informal, i.e. can be stated verbally, or in writing, etc.

The rapidly growing interest in purchasing goods (e.g. music files, e-books, videos, or e–learning content) via the Internet is therefore accompanied by an increasing demand for contracts that are concluded via the Internet. Platforms that offer the exchange, respectively the purchase of goods or services are e.g. Amazon¹, eBay², or iTunes³. Every time customers desire to purchase goods and services offered via the Internet, they usually declare their intent to do so via a click-through-agreement [Ame01], resulting in a contract between seller and buyer. Parties that participate in

¹See: http://www.amazon.com/
²See: http://www.ebay.com/
³See: http://www.itunes.com/
e-commerce are usually not at the same physical location. *Electronic contracts* support the conclusion of contracts between dislocated parties. For the purposes of this thesis, an electronic contract is defined as follows:

A *digital/electronic contract* is an agreement of two or more parties, on the exchange of rights to (digital) goods or services under certain terms and conditions. The memorandum of an electronic contract is digital and can be transmitted via an electronic network.

In the European Union the declaration of intent in electronic contracts is legally binding if it has been stated in the form of an electronic signature [Eur99]. Due to their digital format, electronic contracts have the potential to be electronically processable. The memorandum of an electronic contract in a well-structured, standardised format increases its processability. A way to structure and standardise electronic contracts is the formulation in a rights expression language (REL) (see Chapter 3).

The remainder of this chapter is structured as follows: Section 4.1 introduces the contract life cycle and its four basic states *offer placement*, *contract conclusion*, *execution of contracts*, and *contract archiving*. Section 4.2 addresses the states and state transitions for electronic contracts. Section 4.3 gives a detailed insight into the execution of rights that result from contracts and their processing. At modelling level, an electronic contract can be seen as a composition of different contract objects with various attributes. In other words, an electronic contract aggregates a number of interrelated objects. The core objects, additional application-specific contract objects, and their attributes are addressed in Section 4.4 of this chapter.

The Section 4.5 is concerned with the application-specific generation of electronic contracts. First, various usage scenarios for the application of electronic contracts (e.g. access control, accounting) are identified (see Section 4.4.2) and shortly described. Then, in an example, the required contract objects and their attributes are derived for the usage scenario access control. In the last subsection, Section 4.5.3, a basic process for the tailoring of electronic contracts is proposed.

Section 4.6 addresses the pragmatics of electronic contracts, i.e. their processing in software services. For this purpose the generic Contract Schema (CoSa) is introduced that is an abstraction layer of rights expressions. Within one application or domain all rights expressions are mapped to one contract schema. The application programming interface of CoSa
then allows a uniform querying of the contract information within that application respectively domain no matter what underlying representation the contract has.

In Section 4.7 the enforceability of electronic contracts is addressed. Section 4.8 covers management issues, such as contract validity, digital signatures, identification of contract content, when dealing with electronic contracts. The chapter closes with various related projects and approaches in the field of electronic contracts.

4.1 Contract Life Cycle

The contract life cycle defines the different phases that are undergone by electronic contracts. The contract phases directly affect the required contract characteristics, such as the contract content, and also influence management issues with respect to electronic contracts. The contract phases can be considered from a technical and a legal viewpoint. The legal phases are outlined in the contract life cycle and the technical phases are addressed in the contract states.

Figure 4.1 shows a simple contract life cycle from a non-technical standpoint. The phases were derived from considering the German Civil Code. The contract life cycle has four different phases: the offer placement, the offer confirmation, the contract fulfillment, and the contract archiving. The following paragraphs describe these four phases in more detail.

![Figure 4.1: A simple contract life cycle with four phases](image-url)
1. **Offer Placement.** A contract results from two (or more) declarations of intent about the exchange of rights to goods or services. A signed electronic offer is legally considered as a declaration of intent. Content owners (or rights holders) offer their goods to the consumers on certain terms and conditions. These terms and conditions describe the permissions and duties of the contracting parties (i.e. the consumer and the content owner).

2. **Offer Confirmation / Conclusion of Contract.** At this stage, the consumer confirms the electronic offer made by the content provider. The consumer does so by signing the offer (second declaration of intent) and thereby accepting the terms and conditions; this results in an electronic contract. Note that the contract conclusion is usually preceded by negotiations between content owners and consumers. In other words, a contract can only be concluded if the contracting parties have reached an agreement on the relevant terms and conditions. Sometimes, several new offers are placed until an agreement is reached. Every new offer may include significant modifications to the contract’s original terms and conditions. In order to be legally valid, a contract has to be signed by all contracting parties.

3. **Fulfillment of Contract/Execution of Rights.** In this phase, the contract “fulfillment” takes place, i.e. the contract parties exercise their rights and fulfill their duties under the corresponding conditions. The chronological sequence of these actions can be specified in the contract (e.g. payment in advance). Once all rights have been exercised and all duties have been fulfilled, the contract is completed.

4. **Archiving of Contract.** After completion, each contract is saved in a permanent archive. However, the statutory period for which a particular contract has to remain archived depends on the type of contract (e.g. contract of sale, last will, etc.) and on local law.

### 4.2 Contract States

The various contract states and state transitions describe the process that electronic contracts undergo to move from one phase in the contract life cycle to the next. Figure 4.2 depicts a state chart diagram with the basic states and state transitions of electronic contracts. State transitions are specified by the required event, the /condition/ and the respective /action.
Once a consumer has expressed interest for buying digital goods or services, the owner of the corresponding content places an *electronic offer* and thereby begins the negotiations. In general, the consumer may reject the offer, demand modifications to the offer, or accept the offer. If the consumer rejects the offer, because of unfavorable terms and conditions, and s/he does not wish to continue negotiation, then the negotiations have failed and are consequently aborted. If the consumer rejects the offer, but is willing to negotiate with the content owner, then the terms and conditions can be modified. Subsequently the consumer checks terms and conditions again. [GSSG00] presents the negotiation process with additional cases (e.g. an offer is not answered) that lead to the states in Figure 4.2. If an agreement is reached, both parties (consumer and content owner) will sign the contract. After the conclusion of contract the contract in turn becomes *valid*. Subsequently, both parties can execute the rights and/or have to fulfill the duties specified in the contract. Once all rights have been consumed and all duties have been carried out, the contract is fulfilled and then moved to a permanent archive. These basic states and state transitions can be extended and customised for software services that support this process.

![Diagram](Figure 4.2: Basic states and state transitions of electronic contracts)
4.3 Execution of Rights

This section deals with options for the execution of rights derived from electronic contracts. Rights are exercised in the ‘fulfillment of contract’-phase of the contract life cycle (see Section 4.1). The execution of rights is of particular interest for this thesis as it addresses the process of extracting rights information from contracts and forward it to other software services, such as access control. A detailed technical consideration of this process can be found in Section 7.2.

For the purpose of explaining the ‘execution of rights’-process in detail, in the first subsection the term electronic tickets will be introduced and distinguished from electronic contracts. Tickets can be derived from electronic contracts and redeemed as detached rights. To fully understand electronic tickets, Subsection 4.3.2 introduces tickets characteristics and some research that has been done in this field.

The term contract right in connection with the execution of rights has to be further explained: note that in general the contract duties of the content provider are the contract rights of the consumer, and vice versa. Therefore, instead of defining contract rights and contract duties separately, they can be expressed as contract rights only. For example, a contract has been concluded which states that the consumer has the duty to pay a certain amount in order to receive the right to visit a concert. This contract finally results in two rights, 1. the right of a customer to attend a concert and 2. the right of the concert promoter to collect the corresponding entrance fee from this particular customer. Consequently, a contract right is a triple that comprises an operation (e.g. collect, or access) which has been granted to one of the contract parties (e.g. consumer, provider, beneficiary) and which may be performed on certain objects (e.g. money, respectively digital goods or services) under certain terms and conditions. In contrast to this, a permission is defined as a pair that comprises simply an operation that may be performed on a certain object. An example for a permission is the pair (play, music track) (see Chapter 3). Figure 4.3 is an illustration of the two terms contract right and permission.
4.3.1 Electronic Contracts, Electronic Tickets, and Licenses

In this section, a definition of electronic tickets will be provided and the similarities and differences between electronic contracts and electronic tickets will be identified. In the definition of electronic contracts at the beginning of this chapter, these contracts comprise information about rights of contracting parties to goods or services (the object of the agreement), and their terms and conditions. Permissions are permitted operations to certain objects. Electronic tickets are defined as follows:

A digital/electronic ticket is the option to consume a permission under certain terms and conditions.

Electronic tickets are sometimes called voucher [Nok01]. Electronic contracts and tickets both contain rights expressions pertaining to digital goods or services. However, owning a ticket is different from being party of an electronic contract. Among other things, a contract specifies several rights exchanged between the contracting parties, while an electronic ticket describes an excerpt from a contract, namely one (or more) contract right(s) which can be executed. At the beginning of this section, we mentioned that both contract rights and contract duties can be expressed in the form of rights. Consequently, each contract right specified in a contract can be extracted and formulated as an electronic ticket. In one frequently encountered situation, two parties conclude an electronic contract in which one party receives the right to consume specific digital goods, while the other party receives the right to collect money for these goods. When fulfilling a contract, its rights can thus be transformed into tickets which may be executed independently of each other.

For example, a rock concert promoter sells two tickets for admission to a person who is planning to see the concert with a friend. The right to
attend the concert with a friend (the right of the consumer) as well as the right to receive the concert entrance fee (the right of the concert promoter) can be extracted from the contract and formulated as stand-alone digital tickets. The concert promoter thus owns a ticket that allows him/her to collect money from the consumer, and in return the consumer receives two admission tickets to the concert (see Figure 4.4).

A *license* is a specific type of contract right, respectively a specific type of electronic ticket. A license grants usage rights to intellectual property, technical know-how or technical inventions [Sch03]. Consequently, a license is a ticket for the usage of intellectual property, technical know-how, or technical inventions. Licenses are traded in *license agreements*.

The chronological sequence in which the tickets are executed (or redeemed) can be specified in the electronic contract. Concert tickets, for example, usually have to be paid for in advance. Once the concert promoter has received the entrance fees, he issues the concert tickets to the consumer, thereby granting the right(s) to attend a particular concert.

There are a number of other approaches to facilitate the consumption of rights granted in an electronic contract. One interesting approach is described in the works of Fujimura et al. [FKT+99], Stefik [Ste97], and Rivest [Riv97] who define the term “digital ticket.”

- Fujimura defines a digital ticket as "...a digital medium that guarantees certain rights of the owner and it includes software licenses, resource access tickets, event tickets, and plane tickets.” This defini-
tion is closest to our approach but not considering the ticket as option and does not state what the rights are referred to.

- Stefik compares digital tickets with "... coupons found in a local paper that give discounts on the purchase of grocery products. Issued by a publisher, they correspond to prepayment or discounts for using works by the publisher."

- Rivest et al. have referred to digital tickets as a "... means of payment".

However, Stefik's and Rivest's definition do not correspond our definition given above. According to my definition in this thesis, tickets are part of the contract life cycle and occur in phase 3, "fulfillment of contracts / execution of rights" (see Section 4.1). As mentioned above, some rights specified in a contract might result in a ticket allowing one of the contracting parties to collect money from another contracting party. In this particular case, the digital ticket can be seen as "a means of payment." However, other types of contract rights, such as "attending a concert", results in a ticket issued by the concert promoter; in such cases the ticket serves as "a means of gaining admission." In my view, their definition of tickets as a "a means of payment" or "coupon for discount" is only one special case in which digital tickets can be applied.

4.3.2 Ticket-Driven Rights Execution

This section introduces different types of electronic tickets and their application fields. A digital ticket can be personalised (e.g. bound to a certain individual) or anonymous:

- **Anonymous.** A digital ticket is considered anonymous if only the issuer of the ticket can be identified, while the beneficiary remains anonymous. Since contracting parties can be identified by their digital signatures, an anonymous ticket has to be signed only by the issuer. In other words, a digital ticket must at least include the signature of the ticket issuer. The signature then serves as means for verifying the integrity of the digital ticket and authenticating the ticket issuer.

- **Personalised.** A digital ticket is personalised if it allows the identification of the consumer and the issuer. Anonymous tickets are used e.g. for concerts or bus fares, to name but two examples. In such a case, the identity of the person who executes a ticket is generally
One possible use of personalised tickets is in airline ticketing, as airline companies are required to verify the identity of all passengers on a flight.

In many cases, all ticket information is available in the contract. It is reasonable ask: "What are it sensible applications of electronic tickets"? Examples might include the following:

- **Privacy.** If, for example, the contracting parties want to consume their rights anonymously, a ticket is a means of addressing this issue.

- **Efficiency.** As tickets are excerpts from contracts, they often comprise a smaller amount of data. The actual size of a digital document can be relevant for storage-restricted applications, for example transmitting electronic tickets to chip cards or SIM (subscriber identity module) cards.

- **Specific Ticket Information.** In some cases, consumption-relevant information (for example, the current download location of a digital resource) is not specified in the contract but has to be added to the ticket when issued.

In cases where a contract specifies that goods or services may be accessed a certain number of times, the ticket issuers can use two basic mechanisms to formulate a ticket. They can either issue a certain number of equal tickets, or one ticket that expires once all rights have been exercised. Depending on the intended use of the ticket and the technology used, both mechanisms can be appropriate. Issuing one ticket for each use may result in a large number of electronic tickets to be stored and managed. On the other hand, changing the number of “remaining uses” in the ticket after each use requires greater administration and security effort at runtime. Questions, such as "Who may edit the ticket?" arise in this case.

One proposal for a formal ticket language is XML Ticket [FNS99]. Every ticket formulated in XML Ticket can theoretically be expressed in a digital rights language, but not vice versa. XML Ticket only provides the syntax and semantics to specify a right which a ticket issuer grants to a (subsequent) ticket owner. The XML Ticket language is restricted to these two roles (issuer, owner) and provides no means of expressing other relevant information, such as payments methods, etc. Furthermore, the XML Ticket language is not subject to any current further development. Therefore, for the time being, I propose the formal expression of tickets in a digital rights language as well.
4.3.3 Hybrid Rights Execution

However, the fulfillment of a contract does not have to be regulated exclusively by tickets. For instance, it is not sensible to issue tickets for a service that can be consumed without specific limitations. If, for example, a consumer enters into a contract for an online newspaper subscription without a specific time limit, it is not sensible to issue an admission ticket for each time the newspaper is accessed. In my view, the most sensible use of tickets is to issue them for a single or limited number of access rights, such as downloading a specific resource or streaming a certain video.

As regards the execution of contract rights, a DRM system has the ability to combine electronic tickets with other mechanisms. Figure 4.5 depicts the case in which a customer purchases a subscription for an online newspaper, where electronic tickets are combined with direct processing of access rights. As described in previous examples, upon contract conclusion an electronic ticket is derived from the contract for the right to collect money. This ticket is then forwarded by the processing DRM system to the operator of the online newspaper. In contrast to the contract right of the seller, the contract rights of the customer will not be issued as tickets, but directly processed by the the DRM platform. In this scenario the DRM platform is most likely a secured web server that converts the contract rights to access right on the web server. After the access rights are processed the consumer has the ability to access the online newspaper according to the terms and conditions of the contract (e.g. for one month).

![Diagram](image-url)

Figure 4.5: Combination of tickets and direct rights processing
Note that Figure 4.5 differs from Figure 4.4 in Section 4.3.1, as it addresses not only the derivation of contract right to tickets, but partly describes their processing. The contract rights in tickets as well as the contract rights that are directly processed on the DRM platform contain rights expressions written in a rights expression language. Section 7.2 in this thesis addresses the implementation of directly processing electronic contracts. In Section 2.3 an application and the processing of electronic tickets has been described. In the sample DRM system the booking results in an electronic contract. After the payment is settled, the Content Preparation module adds a license (i.e. an electronic ticket) to the secure container. The license is later processed by the secure viewer on the consumer PC.

4.4 Contract Objects and Contract Use

In general, depending on the content of documents, different general document types can be identified. For example, a recipe is a document type and usually comprises the ingredients for a certain product and the work instructions for the production process. Likewise, a handwritten as well as an electronic contract is a document type that is characterised by containing one or more parties that exchange rights or products under certain terms and conditions. This section introduces a contract data model, that includes typical contract objects and their interrelations. Here the term contract objects refers to instances of classes that occur in contracts.

Contract objects can be subdivided into core objects, and additional scenario-specific objects. In Section 4.4.1 the interrelated core objects of electronic contracts and their attributes are introduced. Electronic contracts can be applied in various usage scenarios (see Section 4.4.2). Each usage scenario requires a distinctive agreement category. Each agreement category may require extra information in the electronic contracts. Scenario-specific objects, and their are addressed in Section 4.4.3.

4.4.1 Core Contract Objects

This section introduces three abstract contract objects which can be seen as core objects of electronic contracts. These three core objects of electronic contracts are: Party, Resource and Permission. The definition of these abstract core objects was influenced by earlier information models [Ian01], my experience with current rights expression languages which often apply
similar approaches [DWW03, Ian02b], and the investigation of projects in which electronic contracts are used (e.g. the COLIS project4).

![Diagram of Abstract Core Contract Objects]

**Figure 4.6: The abstract core objects of electronic contracts**

The core contract objects are interrelated as follows: Specific parties ("rights holders") possess intellectual property rights for one or more specific resources, such as books, software, music files, or digital videos. A party in possession of such property rights is authorised to grant usage permissions to other persons (customers, or beneficiaries). As a result, permissions are assigned to parties. Each permission refers to one particular or one specific type of resource, and one or more permissions may exist for each resource.

Each of the core contract objects shown in Figure 4.6 comprises a number of attributes. The required object attributes and their relevance are explained in the following:

- **Party** is a mandatory contract object that appears at least twice in each contract. A Party instance represents contracting parties, e.g. consumer and seller, and other contract-related persons. With respect to digital contracts, different party types can be distinguished: A rightsholder is a party that holds rights on the respective contract resource and may grant those rights to another contract party. A

---

4See: The Collaborative Online Learning and Information Services (COLIS) Project, [http://www.colis.mq.edu.au/](http://www.colis.mq.edu.au/)
consumer is the party that receives rights from the rightsholder. A beneficiary can be a third party that may actually execute the rights of the contract on behalf of the consumer. Parties must always be identified by a unique id. In general, different types of values can be used to identify a contract party, e.g.:

- A (globally or at least locally) unique identifier that identifies a certain individual, e.g. an X.500 distinguished name [IT93a] or a Kerberos [SNS88] established identity.
- A unique identifier that identifies a certain party type (e.g. as defined in MARC 21 role code list [MAR03]). A party type represents a number of individuals sharing one or more common characteristics. For example, a party type “faculty member” could represent each faculty member at the Vienna University of Economics and BA.

Note that additional scenario–specific attributes can be assigned to parties, such as name, role, position, age, credit standing, profession, etc.

- Resource is a mandatory contract object which denotes the objects of the agreement that are the actual digital goods or services. Resources are likewise identified by a mandatory unique resource id, as well as by optional metadata. Possible identifiers for digital goods are a uniform resource identifier (URI) [BL94] or a digital object identifier (DOI) [Nat00]. The scenario–specific metadata attributes may supply details on the resource, for example book title, author, isbn number, description, size, file format, author’s remarks, etc.

- Permission is a mandatory contract object which appears at least once in a contract. A Permission represents the concrete usage rights granted/assigned to the consumer as a result of the conclusion of the contract. Permissions express usage rights (e.g. play music file, print document, etc.) and may also comprise attributes describing (informal) copyright information or (informal) derivative work rights. A permission always represents at least an ⟨operation, object⟩ pair describing an operation that can be invoked on a specific resource (or object), i.e. the two mandatory object attributes of a Permission software object are operation and object. A simple example of a permission is: ⟨print, researchpaper⟩. Typical operation values include
terms such as print, play, copy, modify. Permissions stated in a valid contract can be enforced either legally or electronically (by a software service). Currently, there is no ongoing standardisation initiative for operation terms. Therefore, operations can not be uniquely identified, which may lead to ambiguous interpretation (see Section 4.6).

Generally, ids have to be uniquely identified for each type of application, even if no globally unique ids, such as x509, or DOI, are used or available. For example, operation terms that are used in an application can be defined in a rights data dictionary (see Section 3.3.2). Rights expression languages (RELs) (see Chapter 3) that are used for contract representation facilitate the expression of the core contract objects mentioned above (e.g. [Ian02b], [DWW03]).

4.4.2 Sample Usage Scenarios for Electronic Contracts

This section presents an overview of potential usage scenarios for electronic contracts. The selection of usage scenarios is based on experiences in the field of electronic contracts, an additional analysis of the literature on rights expression languages [DWW03, Ian02b], research papers [MSMO1, GSSS00] as well as projects dealing with the management of electronic contracts (e.g. the COLIS project\(^5\)).

Access Control

Contracts contain information on permissions concerning digital goods or services. This information is suitable to serve as basis for access control. Theoretically each contract right can be transformed into an access control statement. Primarily, access rights to digital resources that are stated in electronic contract are suitable to be processed in an access control mechanism. The information that is processed in this usage scenario depends on the access control model (e.g. Role-Based Access Control or Discretionary Access Control) that shall be applied.

Accounting

Electronic contracts are the documentation of an agreement between two parties over assets or services. Usually, in an agreement duties respectively payments are stated that have to be settled by the contracting parties.

Therefore, electronic contracts can be an information source for accounting services. For example, every time a contract is concluded, the monetary duties are transferred to open positions in the accounting service.

**Intellectual Property Rights (IPRs) Protection**

From a legal perspective, content owners market their IPRs to customers. Electronic contracts provide a means for the content owner to specify the extent to which the content may be used. As the IPRs specified in electronic contracts can be (semi-)automatically enforced, the IPRs of content owners are protected. Therefore, on a technical level, this can be seen as a special case of the usage scenario access control.

**Customer Relationship Management (CRM)**

The overall body of contracts concluded can represent a valuable data pool for marketing activities. Information on purchasing habits of customers, that is which goods or what type of services customers usually demand, can form an information basis for marketing activities such as one-to-one marketing or personalised marketing within the framework of CRM. For example, personalised goods or services can be offered to the respective customers on the basis of their recorded contract history.

**Workflow Management**

To a certain degree, electronic contracts can be used to specify workflow process information. This process information can be used to control certain task sequences in an information system. For example, let us assume that a contract states that a right is granted after a certain amount has been paid to the content provider. This information reveals a sequencing of tasks that have to be executed and can be used in the workflow process, as follows: an incoming payment event related to an electronic contract initiates the assignment of rights to the respective consumer(s) or contract party.

This section has spanned a relatively broad range of possible uses for electronic contracts. However, it does not claim to be complete and could be extended at reasonable expense. This thesis especially focuses upon the usage of electronic contract in access control services.
4.4.3 Scenario-Specific Contract Objects

The information to be included in electronic contracts varies depending on the scenarios they need to satisfy. For some scenarios not all contract information can be sufficiently represented by the core contract objects or by an extension of their attributes. For example, when using electronic contracts for accounting, the payment conditions and banking details have to be included. Consequently, scenario-specific objects and attributes have to be identified and added to the data model. The additional objects have to be set in relation to the core contract objects. The number of objects and attributes that satisfy one usage scenario is called agreement category. Naturally, the agreement category comprises the core contract objects and scenario-specific contract objects. If in one specific application an electronic contracts has to satisfy several usage scenarios, e.g. access control, accounting, and CRM, the number of contract objects and attributes potentially grows with each additional scenario. More precisely, for each additional usage scenario objects have to be added that are not yet covered by earlier agreement categories. Contracts objects that conform to a specific application (that possibly includes several usage scenarios) are denoted as application-specific contract objects.

Each application then accesses specific contract information, i.e. contract objects and their attributes, when the contract is processed. Figure 4.7 illustrates various agreement categories (AC1-5) and their overlapping as well as how different software services access electronic contracts in order to fulfill specific usage scenarios. The contract objects include the core objects (white rectangle) and additional, interrelated, scenario-specific objects (colored rectangles). The resulting data model of an electronic contract is called application-specific data model. An application-specific data model is described in more detail in Section 4.6. It could be argued that it is sufficient to simply state the ids of resource, party, permissions, constraints, etc. in an electronic contract and query the scenario-specific attributes from a database. However, this proceeding would contradict the original goals that have been defined for this thesis, e.g. to support contract transparency for contracting parties in e-commerce.

The following section describes how additional objects and attributes can be identified for a number of (additional) usage scenarios.
4.5 Contract Modelling and Creation

This section deals with the modelling of electronic contracts and their formulation in rights expression languages tailored to specific usage scenarios. One main reason why the formulation of electronic contracts in rights expression languages and their interpretation make sense is that two or more DRM systems respectively DRM system components use the electronic contracts to exchange rights expressions. Contracts formulated in rights expression languages are suitable to serve as an interface between DRM systems and DRM system components, if their contents meet the specifications of the respective rights expression language (see Chapter 3) that the two systems...
have agreed on. Nevertheless, for most applications it is not sufficient to simply agree on a REL. If the electronic contract shall be reliably processed in a sensible application, such as access control, the DRM systems additionally need to agree on application policies (see Section 3.3.2) and the contract content. The contract content is defined by contract objects as described in the previous section. This section addresses how application-specific contract objects are identified.

A concrete implementation that processes electronic contracts in a specific usage scenario is referred to as software service. Software services are, for example, accounting software, access control mechanism, etc. Section 4.5.1 identifies information which is required to satisfy two different usage scenarios. Section 4.5.2 derives a contract data model from the required information by identifying contract objects, their attributes and relations. In Section 4.5.3, a process for the tailored composition of electronic contracts is introduced. This process supports the composition of contracts, i.e. assembling contract objects and their attributes, tailored to their intended use (i.e. the usage scenarios a contract is to be applied to).

4.5.1 Required Information for Specific Software Services

As mentioned above, all information in electronic contracts and their respective uses should be clearly defined in advance in order to facilitate the automatic processing of contract information. Before identifying the required information of electronic contracts, the intended usage scenarios have to be defined. Each scenario requires a certain number and type of contract objects and attributes in order to process contract data properly, more precisely, software methods that accomplish the respective usage scenario in a well-defined sequence require certain attributes. Therefore, the first step towards tailored contracts is to identify the information that the software methods of each scenario require. In the following requirement analysis, it is assumed that the respective contract shall be processed in an access control mechanism and in an accounting software. For each required attribute it has to be defined whether it is a mandatory or an optional attribute for this application, how often the attributes will occur in the contract, and to which other attributes it is related.
Required Information for Role–Base Access Control

Access control mechanisms aim at regulating the access of users (subjects) to resources. When using the discretionary access control (DAC) approach, access permissions are directly assigned to the users. For example, the permission `read book #"The future of ideas"` is directly assigned to subject `sguth`. The role–based access control (RBAC) approach assigns usage permissions to roles rather than to subjects [FSG+01, SCFY96]. The roles are then assigned to subjects. Thus, users receive permissions transitively via their assigned roles (see Figure 4.8). For example, the permission `read book #"The future of ideas"` is assigned to the role `researcher`, which in return is assigned to user `sguth`. Roles can be arranged in role hierarchies, in which more powerful roles (senior roles) inherit permissions (and constraints) from subordinate roles (junior roles). A role hierarchy is a directed acyclic graph. Roles are a convenient means to assign and manage permissions.

![Figure 4.8: Assigning permissions in RBAC](image)

Permissions can also be associated with constraints [NS03b]. One specific type of constraint is a precondition, i.e. a premise which has to be fulfilled before a permission can be granted/assigned to a specific subject. Other types of constraints might restrict permissions, for example to a certain time interval or to a specific user or device. For instance, prepayment might be necessary in order to receive the permission to play an audio file (assignment constraint), and once the fee is paid, this particular permission might be exercised a limited number of times (authorisation constraint).

Thus, to execute the usage scenario "role–based access control" the following methods are used:

- `createSubject(SubjectID)`: Creates a new subject, i.e. the party that receives access rights from the contract.
- `createPermission(Operation ResourceID)`: Creates a new permission from an operation–object pair (see Section 3.3.1).
- `createRole(RoleID)`: Creates a new role.
- `createConstraint(Name Operator Value)`: Creates a new constraint.

- `rolePermAssign(RoleID PermissionID)`: Assigns the created permission to the role.

- `subjectRoleAssign(SubjectID RoleID)`: Assigns the created role to the subject.

- `relatedConstraintToPerm(Constraint Permission)`: Assigns the created constraint to the permission.

These methods have to be provided with contract information to implement the access rights from an electronic contract to the role-based access control mechanism. In other words, the parameters of the methods are the required information for the usage scenario role-based access control. In the following the RBAC-specific contract information is described in detail:

- **Subject Type.** To identify which of the contract parties receives access rights, each contract party must be further specified by a type, e.g. customer, seller.

- **Subject-ID.** For all contract parties that receive access rights to resources via the contract, a user id is required (e.g. sguth, mstrem). A consumer subject id occurs at least once in a contract and is mandatory data that uniquely identifies the users in the system to which the access control assigned shall be assigned.

- **Resource-ID.** For all resources that access shall be granted to, a resource id is required (e.g. music-file#12345). Resource id occurs at least once in a contract and is mandatory data that uniquely identifies the resource in the system where the resource is stored and secured by the respective access control mechanism. The resource id can also identify a set of resources, such as a folder or a certain type of resource.

- **Operation.** The operation is mandatory data for RBAC that has to occur at least once in a contract. Remember that in the access control community a permission is a pair consisting of a certain operation (e.g. play) and an object (or resource, e.g. music-file#12345). If more than one permission is defined and the contract comprises several contract parties, the relations between parties and permissions have to be stated unambiguously.
• **Role.** The role is mandatory data in RBAC, unless the access control mechanism can also handle discretionary access control (see Section 4.4.2). A role that is "known" by the respective RBAC service has to be assigned to each party that shall receive access rights. A role is a named collection of users and permissions, and possibly other roles [San96]. Role names sometimes resemble user group names, such as *student, employee,* etc.

• **Constraint.** A constraint specifies that certain context attributes must meet certain conditions in order to grant a specific permission. For example, a constraint may specify a date until which the permission is valid. A constraint can be assigned to no, one, or several permissions and has at least three attributes, two operands and one operator, for example, *name, operator,* and *value,* e.g. \((date, <, 12/31/2004)\). Constraints are optional data.

**Required Information for Accounting Services**

From Section 4.4.2 it can be learned that, for example, every time a contract is concluded, the monetary duties can be transferred to an open position in the accounting system. To implement the use of electronic contracts in accounting services the following method has to be called:

• `createOpenPosition(PartyID ResourceID Duty Conditions)`: Creates a new open position in the accounting system.

This means that the attributes Party-ID, Resource-ID, Duty, and Term and Condition are required:

• **Party ID.** For all parties that are involved in monetary transactions an id has to be specified (e.g. sguth, mstrem). Party id is mandatory data that uniquely identifies the user in the accounting system and occurs at least once. In accounting software users are related to payment obligations (duties) and to the resource they have purchased.

• **Party Type.** Each contract party must be further specified by a type with respect to payment relations, e.g. customer or seller.

• **Resource ID.** Resource id is optional data that, if available, uniquely identifies the traded resource in the accounting system (e.g. music-file#12345). Although a duty can be booked without the related resource or service id, it is reasonable to specify this id, to further
specify the business transaction. The resource is related to one or more permissions.

- **Duty.** The duties specify the monetary or non-monetary liabilities between the contract parties or third persons that result from a contract. Duties are mandatory data for the accounting service and for defining the type (e.g. amount of money), value (e.g. 1000,00), and attribute (e.g. €) of the duty. The goods might also be bartered or paid for with artificial credits, which are alternative occurrences of a monetary duty. However, one or more duties can be specified in a contract. Duties are related to a certain contract party and can be related to constraints.

- **Terms and Conditions.** A duty is often afflicted with terms and conditions, such as "the payment has to be settled until 31st December 2003." Such terms and conditions are optional data and can also be expressed as triples, like the constraints in the role-based access control example, e.g. \( \langle \text{settlement datetime}, <, 01/01/2004 \rangle \). None, one, or many conditions can be assigned to a duty and one condition can be assigned to several different duties.

### 4.5.2 Modelling Scenario-Specific Contracts

In Section 4.4 the abstract core contract components *Abstract Party, Abstract Permission,* and *Abstract Resource* have been introduced. Section 4.4.3 explains why for each agreement category it is necessary to extend the core objects with scenario-specific objects to ensure the sophisticated processing of electronic contracts in these scenarios. In this section these scenario-specific objects will be identified on the basis of the attribute analysis in Section 4.5.1.

The required contract objects for the agreement category "role-based access control", are *Party, Resource, Permission, Constraint,* and *Role.* The attributes resource-id, subject-id, subject-type and permission can be represented with the contract objects *Resource, Party,* and *Permission.* Thus, for access control purposes the core objects have to be extended by the objects *Role* and *Constraint:

- **Role** is a mandatory access control-specific contract object. Its single attribute *name* is storing the role name. Roles are related to *Party* objects.
• *Constraint* is an optional, scenario-specific contract object. This object type provides the three attributes *type, operator, and value*. Constraints are related to *Permission* objects.

With these three new contract objects the agreement category for role-based access control service can be represented. The same procedure has to be accomplished for the scenario-specific attributes of the accounting software. The attributes *party id, party type, and resource id* can be represented by the *Party* respectively the *Resource* object. As the object attribute *terms and conditions* can be expressed with the same attributes as the RBAC constraints, no additional objects or attributes are required for this attribute. Finally, to satisfy the accounting service a new contract object of the type *Duty* has to be added.

• *Duty*. The *Duty* object is comprising the attributes *name, value, and attribute*. Duties are related to *Party* objects, and can be associated with *Constraint* objects.

![Application-specific data model](image)

**Figure 4.9: Application-specific data model**

In a contract that has additional scenario-specific contract objects, such as *Role, Constraint, Duty, and Contract*, the relations between the contract objects change, respectively new relations have to be added. A contract
that is processed by an access control mechanism or and accounting software
and automatically results in access rights to protected resources respectively
open positions must be secured, i.e. it is necessary to check if the contract
is valid, e.g. with an electronic signature. Such external characteristics of
the contract do not belong directly to the respective usage scenario and
are implementation-dependent. The additional contract object Contract is
comprising such extra attributes for eventual validity checking, i.e. it can
store digital signature, physical location of the contract conclusion, date,
etc. The Contract object also aggregates the remaining contract objects and
therewith provides a means to assign the contract objects to one specific
contract. Figure 4.9 depicts the application-specific data model, i.e. the
resulting contract objects and their relations (also called contract schema)
that are required for the usage scenarios RBAC and accounting.

In the example, the Constraint object is related to the Permission object
as well as to the new objects Role and Duty. Constraints are capable of
narrowing Permissions, Duties and Roles. A Constraint can be assigned
to an infinite number of Permissions, Duties, and Roles and vice versa.
The Role objects are now assigned to Party objects. The "is-assigned–
to"–relation, depicted in Figure 4.6 between Permission and Party is no
longer required for the current usage. Permissions are now assigned to Role
objects. A Duty can be related to one or more Parties, and one Party
can be related to one or more Duties. A new relation between Party and
Permission indicates which Permissions have been purchased by Parties.

With these contract objects the agreement categories for both usage sce-
narios can be represented. Figure 4.10 shows instances of the object types,
their attributes and actual values. The values have been taken from the
examples of each attribute from the previous section. Furthermore, Fig-
ure 4.10 shows the mapping of the respective instances and their attributes
to the corresponding software services. Each software service may require
several attributes from different instances; contract objects (and attributes)
might be used in one or more software services. For example, role is solely
processed in the RBAC service (or the seller information in the accounting
software), and some attributes are processed in both, e.g. the customer id
and the permission. Due to clarity reasons the aggregation function of the
contract object is not illustrated in Figure 4.10. Please note, that the illus-
trated case is an example, and other access control and accounting software
might use different attributes.

With a growing number of software services, the required objects and at-
tributes for the contract will increase as well. Figure 4.7 in Section 4.4.3 can
be seen as an extended example where various usage scenarios are mapped
to a number of agreement categories respectively contract objects. The modelling of electronic contracts is important for communication and implementation purposes (see Section 4.6). The next section introduces a process that aims at ensuring that electronic contracts include all required objects identified in Section 4.5.1.

### 4.5.3 Scenario–Specific Contract Composition

This section describes a conceptual framework which enables the composition of electronic contracts formulated in any rights expression language and tailored to the requirements of specific usage scenarios. In order to include application–specific objects and attributes in as contract document, it is necessary to know the usage scenarios for this particular contract in advance, i.e. in order to ensure that electronic contracts contain sufficient information to satisfy the requirements of specific usage scenarios, the contracts need to be tailored with regard to their intended use(s). Subsequently, a simple process for the tailored composition of electronic contracts is introduced that adds contract attributes to a contract document based on its usage scenarios respectively its agreement categories. Figure 4.11 depicts such a process; the respective activities are described below.
• **Identify relevant usage scenarios:** In this activity, the list of usage scenarios (see Section 4.4.2) is identified to specify the intended use of the contract under consideration. In other words, the estimated use of this particular contract is defined, for example access control and accounting.

• **Identify software methods:** Here, the software methods of each scenario that finally process the contract data are identified. These software methods, executed in a well-defined sequence accomplish a certain usage scenario.

• **Identify required attributes:** The parameters of the identified software methods are the required attributes of the contract. Therefore, the parameters of each identified software method have to be determined for each usage scenario.

• **Develop agreement category:** For each usage scenario the required contract objects, their attributes and interrelations have to be identified. The resulting data model is the agreement category of the respective usage scenario.

• **Identify application-specific objects:** As shown in Figure 4.9, the agreement categories of the various usage scenarios are overlapping. By combining the various agreement categories, the application-specific contract objects and their attributes are identified. The usage of attributes in software various services is illustrated in Figure 4.10.

• **Append attributes to contract template:** Based on the application-specific objects and attributes a contract template is generated. For this step a tool called *rights expression generator* is used (see Sections 5.1.2 and 6.2). In the analysis shown in Section 4.5.1 the mandatory and optional attributes and their occurrences have been identified. The characteristic whether an attribute is mandatory or optional has to be taken over for the creation of the contract template. The contract template is formulated in the preferred rights expression language. When creating the template, it must be considered that the contract information can be unambiguously mapped to the contract data model, as defined earlier. This is a prerequisite for the reliable processing of the contract (see Section 4.6).

• **Fill in contract:** In this activity the different attributes with actual values are filled in, i.e. the party ids, the party types, the resource
Identify relevant usage scenarios

* [for each usage scenario]

Identify required attributes

* [for each method]

Develop agreement category

* [for each usage scenario]

Append object attributes to contract template

Use rights expression generator

Identify application specific objects

Fill in contract information

Sign contract

* [for each contract party]

Figure 4.11: Composing tailored electronic contracts

ids, the roles, the permissions and duties, and the terms and conditions are included in the contract. Note that the contract parties should have the ability to add additional contract information that has not been identified as mandatory. The contract template ensures the availability of, but shall not restrict the contract to the mandatory attributes. At this point the contract is still in the negotiation or offer phase (see Section 4.1). All contracting parties have to agree on the actual attribute and their values before the contract can be signed. If necessary, the contract is modified until an agreement is reached (see also Figure 4.2).

- **Sign contract**: In the final step, the contract has to be signed by each contracting party in order to be considered as valid. At this point, the contract reaches the 'Conclusion' stage in its life cycle. Without the signatures of the contract parties the contract is not valid (see Section 4.8) and will not be processed.

A framework that supports the above mentioned process, i.e. which generates the contract template and transforms the filled out contract to a REL instance, can be built on top of the rights expression exchange framework designed in Chapter 5.
The tailoring process increases a contract's enforceability [GK02]. For example, a contract consisting of the contract objects shown in Figure 4.9 has the potential to be fully electronically enforceable in terms of access control services and accounting. However, if a contract is not designed as proposed above, it can still be a valuable source of information for specific applications, such as sales statistics and customer relationship management. In such cases, electronic contracts can be processed in 'unstructured' ways, such as in data mining procedures. Then, usage scenarios will be less likely be provided with all required information. Accordingly, the enforceability of such contracts is low.

Note that the question of who decides on the contract usage and consequently on the contract content is a serious privacy matter. This privacy topic also requires a discussion of organisational, management, and privacy issues in contract composition, because the contract content has to be agreed upon by all contracting parties. For example, on the one hand consumers will demand that their personal information is handled confidentially, while on the other hand the marketing department will be interested in personal information for CRM purposes. This issue occurs in every step of the supply chain, in which electronic contracts are applied.

4.6 The Generic Contract Schema

This section deals with the processing of electronic contracts. It introduces a further development of contract models from the previous section and introduces a representation of contracts in a generic data model.

4.6.1 Definition of Terms

In the previous section the three core elements have been introduced. The interrelated objects Party, Permission and Resource are the minimum constituents of every concluded contract, i.e. they build the core data model of electronic contracts. To these core objects, additional objects can be added to represent application-specific information required by the usage scenarios of the contract.

The generic contract schema (generic CoSa) is the theoretical idea of a contract schema that is capable of representing the to-
tal of (electronic) contracts. The entirety of usage scenarios, and thus all scenario–specific extensions of the core elements underly such a generic contract schema. Consequently any (electronic) contract can be mapped to the generic CoSa. The generic CoSa is thus an abstraction layer of various representations of electronic contracts, such as ODRL, XrML, other rights languages, or even contract runtime models. The generic contract schema shall provide for a higher level of standardisation and openness in DRM systems that process rights expressions. To access contract data, the generic CoSa can be queried via the CoSa interface. The CoSa interface is a generic application programming interface (API) which is independent of usage scenarios, allows to query all contract data, and thus facilitates a standardised processing of contract data in software services.

The implementation of the generic contract schema is a considerable challenge and probably technically impossible. It would require knowledge about all today’s and tomorrow’s contract usage scenarios as well as the accordingly needed contract objects and their relations. Yet, the generic CoSa interface is practicable and will be presented later in this Section. However, with some restrictions the generic contract schema can be implemented and helps facilitating and standardising the processing of contract data:

- **Domain–specific CoSa.** The domain–specific CoSa covers a great number of deal of popular contract objects and their relations in a specific domain (e.g. education, music industry). The domain–specific contract schema would permanently undergo further development (such as the the Learning Object Metadata (LOM) standard [IEE02]) and an independent organisation (such as the Learning Technology Standards Committee (LTSC)) would watch and control the development of the contract schema. The domain–specific CoSa would also provide guidelines for the contract object attributes and their allowed values. For example, the permitted attributes of Resource objects are all defined in the LOM standard, or the permitted attributes for Party are attributes defined in the vCard [HF98] standard. If not already defined in the metadata standard, the permitted values of attributes need to be stated as well, e.g. the attribute Identifier may comprise ISSN [ISO98], ISBN [ISO92], and DOI [Nat00] compliant identifiers. The extension of the contract schema by new objects

---

6See: http://ltsc.ieee.org/
<table>
<thead>
<tr>
<th>Covered Scenarios</th>
<th>Domain-specific</th>
<th>Application-specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevalent ones</td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>&lt; 100%</td>
<td>100%</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
<td>Poor</td>
</tr>
</tbody>
</table>

Table 4.1: Characteristics of application-specific and domain-specific CoSa

or attributes has to be requested and publicly discussed. After this process, new objects can be added to the schema by relating them to existing objects. Technically, this means that it can not be guaranteed that contract data which is not covered by the domain-specific CoSa can be reliably processed respectively enforced. Still, this approach offers high flexibility for actors in the educational domain, as electronic contracts can be easily processed in additional usage scenarios (see Table 4.1). Also by implementing the domain-specific CoSa, actors in that domain can ad hoc use and provide services, such as trading, booking, and rendering of electronic goods.

- **Application-specific CoSa.** The generic contract schema, as defined above, also holds for a specific, closed application, i.e. an application specific data model (such as shown in Figure 4.9). One application can comprise several usage scenarios, e.g. access control and accounting. The application-specific CoSa is fixed in advance; apart from the contract objects this also includes the object attributes and their permitted values. As in the domain-specific CoSa already existing description standards, such as LOM, Dublin Core or vCard, can be reused for this purpose. All contract data can be reliably processed in the designated usage scenarios. The disadvantage of this approach is caused by its prerequisites (predefined and fixed data model) that lead to a poor flexibility. The processing of electronic contracts in additional usage scenarios or extended object attributes requires changes in the application-specific contract schema which, in return, requires software modification of the contract interpreter.

The characteristics of the domain-specific and the application-specific CoSa are opposed in Table 4.1.

### 4.6.2 Application-Specific CoSa Example

In the following a simple example of an application-specific CoSa is presented, in order to address further technical details of the CoSa approach.
In this sample application it is assumed that the processing in intellectual property rights (IPR) management services is the only usage scenario of electronic contracts. This usage scenario requires additional information about the contract itself, i.e. it is necessary to store the physical location where the contract is concluded to determine the legal venue. Therefore, the core objects have to be extended by the object Contract. The Contract object aggregates all other objects and therefore has to be related to all core objects (Party, Permission, and Resource). Figure 4.13 illustrates the objects and their relations of the agreement category for IPR. Because the application only includes one usage scenario, the agreement category for IPR is concurrent with the application–specific CoSa.

Figure 4.12: Class diagram of an application–specific Contract Schema

The resulting class diagram of the application–specific contract schema is depicted in Figure 4.12. This figure also reflects the functionality of the Contract object to aggregate all other objects of one particular contract. The class CoSaObject comprises common instance variables and methods of all contract elements (here, the attribute relations).

Figure 4.14 shows instances from the application–specific CoSa classes. In fact, it depicts all instances that are part of a small sample contract (see ODRL serialisation in the listing below) and that have been mapped to the application–specific CoSa.
Core Contract Objects

Party

Permission

Resource

contract

Permission comprises

grants

is assigned to

Resource comprises

refers to

Contract comprises

possesses control permission for

agreement category "Intellectual Property Rights Management"

Figure 4.13: Application-specific contract schema

<rights>
  <agreement>
    <context>
      <pLocation> Germany </pLocation>
    </context>
    <party>
      <context>
        <uid> #sgraf </uid>
        <name> Steffi Graf </name>
      </context>
      <rightsholder/>
    </party>
    <party>
      <context>
        <uid> #bbecker </uid>
        <name> Boris Becker </name>
      </context>
    </party>
  </agreement>
</rights>
The contract has two parties, the consumer Boris Becker with the (system wide) unique id #bbecker and a seller Steffi Graf with the unique id #sgraf. Steffi is selling a print permission for an Ebook on international Tennis rules to Boris. The Ebook has the unique id ebook#123. The contract itself was concluded in Germany. Classes from the application-specific CoSa have two different kinds of attributes: contract attributes and intrinsic attributes:

- **Contract attributes.** In contract attributes the factual contract information is stored. Here, FN, UID, ROLE are contract attributes for the object Party and Identifier and Title etc. are typical contract attributes for the object Resource. The application-specific contract schema is reusing existent, standardised vocabulary for the contract attributes. The Resource attributes are expressed by the Dublin Core [Dub01] vocabulary, and Party attributes are taken from the vCard

![Figure 4.14: Instance of an application–specific Contract Schema](image)
standard [HF98]. We are not aware of initiatives that define usage and access rights or general contract attributes customary in a line of trade. The contract attributes are extensible by additional vocabulary sets.

- **Intrinsic attributes.** Intrinsic attributes are necessary to express the relations between contract, objects as defined in the contract data model. The contract schema defines the attribute relations to store all relations that the actual objects has to other objects in the contract schema. This attribute comprises a list of (type, CoSa-object)-pairs. Each pair represents one relation, for example, the relation \( \text{refers to} \) of obj03 expresses that the permission obj03 is granted to the resource that is represented by object obj04. All possible relations for this specific application can be derived from the data model of the contract schema (see Figure 4.13). One relation in the data model results in two relation roles (names for both ends of the relation), the role names have to be unique. For example, the relation with the label \text{refers to} between the CoSa objects Resource and Permission in the data model results in the two roles \text{refers to} and \text{rel_perm} (related permission). A list of all roles from the current small example can be found in Table 4.2.

### 4.6.3 The CoSa API

The generic CoSa aims at supporting the standardised processing of electronic contracts. Therefore, in the previous section an application-specific contract schema has been developed and described. To also access the contract schema in a standardised way, an application programming interface...
(API) is required that allows to query every information from the contract schema and process it in subsequent software services. In the following, such application programming interface, the CoSa API, is introduced, i.e. the CoSa API. The methods only are presented with a short description, for a complete reference of the CoSa API please refer to Appendix B (Chapter 10.1). The method prefix cosa represents an instance of a class that is implementing the CoSa API. The cosa instance is initiated with an electronic contract and holds all contract information in runtime mapped to the application-specific CoSa. The question marks tag parameters that are optional. The method calls are coded in XOTcl [NZ00b].

- **cosa getObjects object-type**
  This method returns instances of the class object-type that exist in the current runtime CoSa. For example, the following command would return the instances obj05 and obj02 as a list.

  ```
  cosa getObjects CoSaParty
  ```

- **cosa getRelatedObjects cosaObject ?relation? ?object-type?**
  This method returns all instances related to cosaObject. The related instances can be optionally filtered by either the relation, or by the object-type of the related instance, or both. For example, the following command would return the instance obj03.

  ```
  cosa getRelatedObjects obj02 hasPerm
  ```

- **cosa getRelObjectTypes cosaObject ?relation?**
  This method returns the object-types (classes) of all CoSa objects related to cosaObject optionally filtered for a given relation. For example, the following command would return all the object-type CoSaPermission of obj03 which is related to obj02 in the has_perm manner).

  ```
  cosa getRelObjectTypes obj02 has_perm
  ```

- **cosa getRelations cosaObject ?relation?**
  This method returns the relations attribute content of cosaObject.
The content is a list of \((relation - typecosaObject)\)-pairs. If the argument \(relation\) is specified, all pairs are returned of which \(relation\) equals a certain relation type. For example, the command below would return a list with one pair, namely \((agg\_parent \ obj01)\).

\texttt{cosa getRelations obj04 agg\_parent}

- \textbf{cosa hasRelation cosaObject relation}
  This method determines whether \(cosaObject\) has a relation to a specific other instance or not. For example, the following command would return \texttt{true}.

\texttt{cosa hasRelation obj04 agg\_parent}

- \textbf{cosa getAllAttributes cosaObject}
  This method returns all attributes of \(cosaObject\). For example, the following command would return the list \{\texttt{relations UID FN ROLE}\}.

\texttt{cosa getAllAttributes obj02}

- \textbf{cosa getAttributeValue cosaObject attribute}
  This method returns the value(s) of the attribute of \(cosaObject\). For example, the following command would return the value \texttt{consumer}.

\texttt{cosa getAttributeValue obj02 ROLE}

- \textbf{cosa selectObjects list attribute ?value?}
  With this method a list of \(cosa\) instances can be filtered with respect to a certain attribute, respectively its value. For example, the following command would return the instance \texttt{obj05}.

\texttt{cosa selectObjects \{obj02 obj05\} ROLE seller}

This API is short but generic. When new contract objects or relations are added to the contract schema, the API does not have to be changed. You may have noted that the API allows querying contracts but does not offer methods to create or modify electronic contracts. This is due to the
characteristics of contracts. Once contracts are concluded, they should not be changed. The concept of the contract schema therefore is not addressing the offer creation, negotiation or contract conclusion phase, but is intended to support the fulfillment respectively the automated processing of electronic contracts.

4.6.4 CoSa Serialisation

The contract schema is the heart of a contract processing framework. As mentioned above, it determines all information of electronic contracts that can be processed in software services. Let us assume that new partners or platforms desire to participate at a domain or application that has defined a CoSa, for example accounting software providers. They would need to know what the contract schema looks like and if it meets the requirements for processing electronic contracts in their accounting software. Additionally, programmers need the names of CoSa objects and their attributes in order to call the CoSa API methods. In short, a generally accepted representation of the CoSa is required to communicate its shape. The resource description framework (RDF) [LS99] is suited for this purpose. The following listing illustrates the RDF serialisation in XML of the application-specific CoSa exemplified in Section 4.6.2:

```xml
<?xml version="1.0" ?>
<RDF xmlns:RDF = "http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
     xmlns = "http://www.guth.it/CoSa#" >
   <RDF:Description about = "Contract Schema for IPR Applications">
     <CoSaObjects name='simple contract type'>
       <CoSaParty>
         <oid/>
         <relations>
           <has_perm/>
           <grants/>
           <control_perm/>
           <agg_parent/>
         </relations>
         <FN/>
         <UID/>
         <ROLE/>
       </CoSaParty>

       <CoSaResource>
         <oid/>
         <relations>
           <rel_perm/>
       </CoSaResource>
   </RDF:Description>
</RDF>
```
The RDF instance describes the objects and their attributes of the application-specific CoSa. The description contains all information that has been determined in the earlier data model of the specific application (see Figure 4.13). Both the data model and the RDF description serve as communication basis for engineers and developers of applications that process electronic contracts. The following numeration summarises the characteristics and advantages of the generic contract schema concept. The generic CoSa ...

... is a concept that enhances openness and interoperability in DRM systems.

... aims at standardising the processing of electronic contracts.

... serves as abstraction layer for various representations (e.g. rights expression languages) of contract information. The abstraction is achieved
by mapping the contract attributes from proprietary representations to generic objects and their attributes\(^7\).

\[\ldots \text{ defines a generic data model for contracts.}\]

\[\ldots \text{ has to be restricted either by domain or application, to be applicable.}\]

\[\ldots \text{ and its API provide standardised access to contract information that has been transformed into the application- or domain-specific contract schema.}\]

Another application-specific contract schema, namely for the usage scenarios access control and accounting, has been developed in Section 4.5.2. An implementation of this contract schema and its processing is shown in Chapter 6.

### 4.7 Enforceability of Electronic Contracts

Electronic rights enforcement aims at verifying specified usage rights in digital contracts and ensuring their observation both by electronic means. Thus, electronic rights enforcement addresses the enforcement of access rights to electronic goods but not to physical goods. However, the rights to physical goods can still be enforced by legal action. Not all usage rights that can be expressed in digital rights languages can be electronically enforced. For business partners and, above all, for rights holders it is important to know to what extent electronic contracts are enforceable. Identifying and ensuring enforceability of electronic contracts should generate trust as a basis for successful electronic commerce. Therefore, I suggest to divide usage rights into two categories, based on their enforceability in electronic contracts:

- **Non-enforceable rights**: These parts of electronic contracts that specify usage rights for resources, which cannot be observed by computer technology.

- **Enforceable rights**: These parts of electronic contracts that specify usage rights for resources, which can be enforced by computer technology. This category must be further specified into:

\[7\text{For example, a resource in ODRL is expressed by an asset-tag. XrML provides a tag called digitalWork. To implement the generic CoSa the two proprietary terms have to be mapped to a generic term, here resource}\]
— *Potentially enforceable rights*: these parts of electronic contracts that specify usage rights for resources that are currently not enforceable but have a high potential to be enforced under certain circumstances.

— *Reliably enforceable rights*: these parts of electronic contracts that specify usage rights for resources, which can be reliably enforced as intended by contract parties with existing computer technology.

In order to identify the different levels of enforceability of a given set of rights, clear criteria are needed. Accordingly, the three criteria for electronic contracts to be enforceable are: *availability of required information, availability of appropriate technology, and implementation of a trusted environment.*

**Availability of required information in the system**

The first criterion for the enforceability of an electronic contract is that all required information can be recorded, and/or is available to the system.

**Example:** As the extensibility of digital rights languages and therefore the range of expressions is limitless, there are no boundaries to information in electronic contracts. The following right expression might not be a common one, but it represents a possible clause in an electronic contract. "The consumer may have access to my entire resources, after s/he has invited my department for a discussion round." The precondition "after s/he has invited my department for a discussion round" is not enforceable, since the necessary information (receive invitation) to check the precondition cannot be recorded by the system.

**Control over the usage of resources**

A second criterion is the availability of an appropriate technology that permits controlling the specified usage rights for the resource format concerned.

**Example:** A specified usage right for a resource could be (in words): "The consumer may show the digital video to a class once per semester." We now have all relevant information required to enforce this rights expression. However, in order to prevent the video from being shown more than once per semester, a reliable enforcement technology is needed that is capable of monitoring the usage rights of video formats.
The music industry is currently promoting the development of such technology.

**Availability of a trusted environment**

The third criterion is the availability of a trusted, i.e. tamper-resistant environment. The term "trusted" here refers to the point of view of the rightsholder who anticipates a license conforming access to resources of the consumer. Rights enforcement in a "trusted" environment is expected to be reliable, i.e. in an non-trusted environment electronic tickets or contracts can be modified, copied, or forged and are then longer enforced as intended.

<table>
<thead>
<tr>
<th>Enforceability</th>
<th>Enforceable</th>
<th>Non-Enforceable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information = available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology = available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment = trusted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information = available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology = not available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment = trusted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All other cases</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.15: The enforceability matrix**

**Example:** Rights enforcement is easier to implement if the resources are not delivered to the consumer but remain with the provider. The consumer then receives "access rights" to the resource but physically the goods remain stored on the providers' server. For example, when consumers purchase access to an online newspaper, every time they desire to access the latest news, they have to authenticate themselves to the online newspaper. In this case, the system administrator of the platform retains responsibility for access control of the resources. Conversely, rights enforcement is hard to implement if the execution of usage rights is managed by software on the client PC, because then the management of access rights is not the responsibility of the delivery system. We classify the environment of the provider as rather
"trusted", and that of the consumer as rather "non-trusted", because no 100-per cent "trusted environment" exists [Fed02]. Due to the characteristics of certain goods it is difficult to provide a trusted, tamper resistant system. For instance, as far as the distributing of music, video, etc. is concerned there is a great amount of recording technology that facilitates making unprotected copies of the resource.

The relation between these three criteria and the three levels of enforceability are represented in matrix form (see Figure 4.15). If all three criteria are fulfilled the electronic contract is enforceable. The process for the composition of tailored electronic contracts (see Section 4.5) respectively the application-specific contract schema in Section 4.6 are means to ensure the availability of all required contract information and facilitate electronic rights enforcement. The DRM sample system (see Section 2.3), for example, addresses the criteria available technology (e.g. secure viewers) as well as the trusted environment (e.g. secure containers).

4.8 Contract Management Issues

When aiming at processing electronic contracts, a number of technical challenges arise, such as representation of contracts in a machine readable format (addressed in Chapter 3) or defining software objects that are able to store contract information in a software program (addressed in Section 4.4.3). Besides the purely processing issues some managerial questions appear, such as "When is a contract valid?" or "How can I be sure that I bought the content that I intended to buy?" This section aims at drawing attention to these management issues and addressing some relevant ones of them in detail. For the overall focus of this work the identified issues are mainly derived from the access control environment. Figure 4.16 shows the occurrence of some management issues that typically occur along the four phases of the contract life cycle (see Section 4.1).\(^8\) All issues will be addressed in the subsequent paragraphes in detail.

Managing Offer Placement

In the offer placement phase important management issues are content administration, checking the offer validity, and checking the authorisation of the seller.

\(^8\)Please note that this sequence and allocation of issues is not universally valid.
Content Administration

Creating an offer implies that the sellers have access to a list of their content. This list should be sorted by sensible criteria, such as format, content type, offer-already-created, etc. For resources on this list, sellers are allowed to formulate offers. Offers should also be visible to the seller after they have been formulated. To enable these features, metadata that describes the content has to comprise the content criteria, and sellers have to be related to their content and to the offers they have created.

Checking Offer Validity

Checking whether an offer is valid can comprise several activities and varies depending on the concrete application. By all means the digital signature of the offer has to be checked.

- Digital signatures provide several services, e.g. unique identification of the offer signer. Generally, digital signatures are a technical means to provide authentication, verification, non-repudiation and integrity [Sch79]. The offer is only valid if the signature on the offer is verifiable and unambiguously identifies the signer. Along with the signature, sufficient information about the signature such as hash algorithm or key length algorithm needs to be provided to technically facilitate the signature verification.
• Besides verifying the signature it should also be checked whether the contract content is sensible, e.g. if the offered goods are in stock, if the terms and conditions are reasonable, etc.

• After the offer has been signed, the signature unambiguously identifies the seller. Here it is an important management issue to check if the signer is a person that has the authorisation to create and sign an offer. For the offer to be valid, the signer needs to be a person who owns property rights of the offered resource. Often the content creator is such an authorised person, but sometimes retailers or second-hand vendors have such rights as well.

Managing Contract Conclusion

In the contract conclusion phase of the contract life cycle the three issues contract negotiation, contract signing and defining contract validity play an important role.

Negotiation

On the way to reaching an agreement between buyer and seller concerning terms and conditions of the contract, often a negotiation between contract parties takes place. For example, contract party A offers certain conditions to party B. B has the choice between agreeing and rejecting these conditions. If the conditions are rejected, party B may state new conditions and present them to party A. This process is called negotiation (depicted in Figure 4.2) and needs to be technically facilitated in the conclusion phase of electronic contracts.

Contract Signing

After the contract parties have agreed on the contract content, both parties have to sign the offer. At this stage, one party has already signed the electronic offer to make it legally valid. The subsequent signature of the accepting contract party then is called offer confirmation and automatically results in the conclusion of the contract. Like the offer signing, the contract signing requires the respective environment. To facilitate digital signatures and their verification in general, a public key infrastructure (PKI) has to be provided. PKI [KL89] offers means to issue key pairs, issue and revoke certificates, create and verify signatures, etc.
Defining Contract Validity

"When is a contract valid?" is a question of great importance when processing electronic contracts and brings up a lot of crucial issues: Is the contract valid because all contract parties have signed it? How long is a contract valid? Is the contract still valid after it has been processed by an access control service? How can double spending of electronic contracts be avoided? Guidelines for these issues should be well-defined in the contract conclusion phase. These guidelines for contract validity have two main dimensions: there are guidelines to make the contract legally binding, and others to make it technically valid.

- Legal requirements. If an electronic contract is designed to be legally binding, the signing certificates have to be qualified, and the electronic signatures have to meet the nationally required security level [Eur99]. Legally binding electronic signatures have to

  - be exclusively assigned to the signer,
  - unambiguously identify the signer,
  - be created with means that can be kept under the signer’s control,
  - ensure integrity of the signed data, and
  - be based on a qualified certificate that has been created by technical components and practices that meet the security requirements of the national law and its regulations.

A qualified certificate can only be issued by a qualified certification authority. Qualified certification authorities are nationally registered authorities that have undergone a quality and security check by the regulatory authority for electronic signatures.

- Technical requirements. First of all, some technical requirements are predetermined by the legal requirements. The system that processes electronic contracts has to provide technology that meets the legal requirements. Besides the implementation of legal requirements a DRM system usually has additional specifications for valid contracts, such as

  - the electronic contracts have to be written in a specific rights expression language (see Chapter 3),
- the contracts need to comprise certain information for specific usage scenarios (see Section 4.5),
- the objects (parties, resources, etc.) in electronic contracts have to be uniquely identifiable by the system,
- the resources stated in the contract have to be available in the system,
- the electronic contract must not have been revoked,
- the electronic contract must not be expired,
- if the electronic contract shall be processed by an access control service, it must not have been processed successfully before (see *check contract status* in Section 4.8).

**Managing Contract Fulfillment**

This section addresses management issues in the contract fulfillment phase of the contract life cycle, such as *Contract Provision, Checking Contract Validity, Authenticating Beneficiary* and *Error Management*, leaving out the technical issues, such as contract parsing, interpretation and processing.

**Contract Provision**

The contract fulfillment phase includes the execution of rights. First of all, the contract that shall be executed has to be provided to the executing platform. There are two basic possibilities to provide contracts to a platform:

1. *Contract remains with platform.* After the conclusion of a contract the platform stores and administers the contract. Beneficiaries arrive at executing their rights by authenticating themselves to the platform and then receive access to resources and services according to their contracts.

2. *Ticket/Contract remains with the consumer.* The consumers receive the contract after its conclusion and are required to provide the contract at the time they want to execute their rights.

In the second case, the following management issue arises: if several contracts are in the contract repository of the consumer, who chooses the right contract? Does the consumer need to know what the appropriate contract is, like it is the case with x509 certificates? Or does the platform have the
intelligence to pick the right and valid contract from the contract repository of the customer? This example names management issues that will arise with the dispersion and frequent use of electronic contracts.

**Checking Contract Validity**

An electronic contract is valid if it meets the legal and technical requirements defined in the contract conclusion phase. Technical requirements include the implementation of the legal requirements and the definition of system-specific respectively application-specific requirements. For example, checking contract validity may comprise the following steps: checking the contract status, checking the contract signatures, identifying contract objects, and checking resource availability.

- **Checking Contract Status.** As soon as the electronic contracts arrive at the platform with the request to execute certain rights, the contract processing platform has to check the contract status. This activity includes the inspection of specific contract revocation lists in order to identify expired and legally revoked (i.e. invalid) contracts and to prevent duplication, respectively "double spending" (see e.g. [MN93]), of digital contracts and the herein granted permissions. For example, the "double spending" prevention procedure is necessary if a contract defines a maximum number of uses for certain digital goods/services and the contract has already been executed on the same or on another platform.

To the best of my knowledge, there are no existing revocation lists for electronic contracts yet. Such revocation lists, would have similar functions as certificate revocation lists [KL89], such as identifying and publishing expired, legally invalid, manipulated and forged contracts, as well as publishing contracts that have already been fulfilled. Running contract revocation lists requires the unique identification of each electronic contract in a closed system.

- **Checking Contract Signatures.** Before a contract is ready for processing, this step verifies the digital signatures of the corresponding contract to ensure the integrity of the contract and the authenticity of the contract parties. This activity comprises all legally and technically required tasks as defined in the contract conclusion phase. In this context the sustainability of electronic contracts is an important issue.
As mentioned earlier electronic contracts signed with "qualified" certificates are legally binding. Digital certificates get a time stamp from the qualified certification authority at the time they are issued and have a certain validity period. After the time period has expired the digital certificate is invalid and no longer suitable for the verification of electronic signatures. To ensure the sustainability and verifiability of digital certificates and thus of electronic contracts, additional time stamp and naming services have to be applied. These services extend the basic functions of digital signatures. The time stamp service affirms that a particular contract has been presented to the time stamp issuer at a certain point in time. The naming service (of a certification authority) handles the binding of signature keys to persons and confirms whether a certificate has been valid at a certain point in time [BRB99]. To verify a signature from a certificate that has expired, the naming service has to be questioned about whether the signature on the respective contract was valid at the time of the time stamp. Thus the verifiability of digital signatures can be provided. This issue is also addressed in the work of Anagnostopoulos et al. [AGT01], which presents data structures that can support an infrastructure for persistent authenticated dictionaries. Applications include credential and certificate validation checking in the past (as in digital signatures for electronic contracts), digital receipts, and electronic tickets.

- Identifying Contract Objects. To execute a contract as it is intended by the contracting parties, all objects that occur in electronic contracts have to be uniquely identified. Contract objects are for example, resources, contract parties, etc. Here it is important to distinguish between system-wide and world-wide, i.e. globally unique identifiers. For some applications it might be satisfactory to uniquely identify the contract objects system-wide. For example, it is adequate to identify subscribers of an online newspaper only by their id, which is unique on the online newspaper platform. In cases where contracts state the royalties for authors and the contracts circulate on various different platforms, an id is required that is globally unique. System identifiers are generated and assigned to objects by the respective system, whereas global identifiers are issued and assigned to objects by international initiatives. Examples of globally unique identifiers are the digital object identifier (DOI) [Nat00] for the identification of digital goods respectively resources and the uniform resource name (URN).
that globally and uniquely identifies a resource or unit of information independent on its location. Also an IP address [Pos81a] could serve for the identification of a certain resource, or the X.500 distinguished name [IT93a] for the identification of a certain individual.

- *Checking Resource Availability.* Prior to finally execute the electronic contract, the system has to check on resource availability. Although the contract can be valid otherwise, it can not be executed if the resource, identified in the contract, is not available.

**Authenticating Beneficiary**

As referred to in the issue *contract provision* an authorisation of the beneficiary is required for usage scenarios, such as access control. If the electronic contract shall be used to grant access to electronic resources, the access control service first and inevitably demands for the authentication of parties. The system initiates the authentication process for the party that has triggered a particular access request. Possible authentication mechanism are e.g. a Kerberos-based service [SNS88], or an authentication infrastructure based on X.509-certificates [IT93b]. Authenticating the beneficiary is relevant for access control services but not for other purposes, such as accounting.

**Error Management**

From the previous paragraph one can conclude that electronic contracts are sometimes not valid or not qualified to be processed in certain software services. In cases where electronic contracts have been rejected to be processed, in certain software services an error management needs to be available. The error management defines how to proceed with rejected or unemployable electronic contracts. Like in programming environments, the error management should distinguish between *error severity*, *error types*, and support *error routines*.

- *Error Severity.* The error management should distinguish different grades of severity of errors. Some errors might be severe and cause the rejection of the electronic contract, while others might not result in the contract rejection but only throw a warning during processing. Errors might be assessed as variably severe depending on the software
service that processes the contract. For example, the accounting service rejects a contract if the contract does not include a monetary obligation. As the access control service is not dependent on this information, the missing attribute does not even cause a warning.

- **Error Types.** The error management should distinguish various error types, for example invalid-signature, contract-expired, contract-already-executed, object-identification-failed, contract-format-invalid, resources-not-available, etc. The identification of error types is important for the further handling of errors. Depending on what error type occurs the respective error routine is called.

- **Error Routines.** Error routines define activities that are executed on the occurrence of a certain error type. For example, if the system is checking an access request for a certain resource and the error type invalid-signature occurs, the system would reject the access request. The error type contract-already-executed would initiate a routine that rejects the access request and additionally adds the electronic contract to a public list of already "spent" contracts.

**Managing Contract Archiving**

The archiving of electronic contracts brings up the management issues Changing Contract Status and Determining Archiving Format. These issues deal with the consequences of contract completion and with proper contract storage.

**Changing Contract Status**

If a contract has been completed and all rights have been executed, the contract should officially get the status fulfilled. Ideally the electronic contract is registered on a public list for fulfilled contracts that other DRM systems and platform may access. Such public lists reduce the risk of contract double-spending (see e.g. [MN93]).

**Determining Archiving Format**

Electronic contracts that have been executed are often archived. However, the national laws on electronic contracts have different statements for the
format of electronic contracts and their archiving. In Germany, for example, solely electronic contracts that require to be stated in written form in the non-electronic world have to be archived [Deu01b]. The archiving periods for written and electronic contracts are the same and depend on the contract type, such as purchase, insurance contract, etc. Such electronic contracts have to be readable for all contract parties and also for third parties, such as lawyers. A detailed treatment of legal issues of electronic contracts would go beyond the scope of this thesis. To read more about these issues in Europe, and particularly in Germany, please refer to the following EU-directives: directive of electronic commerce [Eur00], directive on distant contracts [Eur97], and directive on electronic signatures [Eur99], respectively the implementation into national German law Gesetz zum Elektronischen Geschäftsverkehr (EGG) [Deu01a], Fernabsatzgesetz [Deu00a], and Novelle des Signaturgesetzes [Deu00b].

The above-mentioned management issues occur during the life cycle of electronic contracts. I do not claim that the list is complete, but it names important issues that have been discussed during the development of this work. In a concrete application some issues might become obsolete whereas others might have to be added. The management issues heavily depend on the respective usage scenario.

4.9 Related Work

This section discusses other approaches to modelling electronic contracts. Some approaches include an introduction of their underlying data model of electronic contracts [LDF+02, KGV99, GHM00], which most times refer to one special application rather than to a generic model for electronic contracts. In most of the works a (proprietary) XML-based language for the contract serialisation [NCL+03, SDN+00, ZSO1] is used or at least proposed [BJPW99].

Service Level Agreements

In the work of Keller et al. [KKL+02] a management architecture for specifying, deploying, monitoring, and enforcing service contracts is proposed to provide a basis for service level agreements. A service level agreement is a contract that defines the technical support or business parameters that an application service provider or other IT outsourcing firm offers its clients. The agreement spells out measures for performance (i.e. quality of service
(QoS)) and consequences for failure. Keller et al. define object classes that represent their contract model. Their contract model is tailored to the needs of service level agreements, and thus contains other contract objects than the contract model introduced in this work (see Section 4.6). However, their model does contain the core contract objects that have been defined in Section 4.4.1: provider and customer (i.e. parties), and service (specifying permission and resources), as well as objects that represent the guaranteed service parameters (i.e. constraints). In a later work Keller et al. [LDF+02] motivate the description of service level agreements in a standardised format, namely the web service level agreement (WSLA) language specification. Keller et al. finally demonstrate the application of this language in [KL02] for the dynamic e-Business. In this work and in the language specification Keller et al. explicitly state that the general structure of a service level agreement can be described with three basic object types: parties, service description, and obligations (see Figure 4.17).

![Figure 4.17: General structure of an service level agreement [LDF+02]](image)

Parties are, for example, service provider and service customer. Service descriptions include the guaranteed service (e.g. stock quote service) and the service parameters (e.g. availability and response time). The service parameters can be seen as rights to a respective service (i.e. permissions). Obligations define the appropriate actions (duties), e.g. payments to be taken if a violation of a guaranteed service level has been detected. One important difference from the licenses and contracts presented in this thesis is that service level agreements define not one but several different sets of permissions and duties between the contract parties. All these sets are bound to a condition. When executing the contract one of these conditions is met and the respective set of permissions is granted. For example, the service provider receives a basic fee for a service, e.g. a stock quote service. If the service provider meets all conditions (constraints) that have been defined in the service level agreement, the fee will be regularly paid. If the
service provider can not provide the promised conditions (e.g. availability and/or response time) of the service, the payable fee will be reduced (as specified in the agreement). The introduced rights expression languages are not designed to support service level agreements. A subject of future work is an analysis about whether the predominant RELs are or should be able to express service level agreements.

The work of Meredith and Bjorg suggests using electronic contracts to ensure quality of service in a technical environment where web services discover and interact with each other [MB03]. The work does not propose a specific rights language but discusses, what such contracts should cover.

**Rule XML**

The Babel project aims at providing interoperability between inhomogeneous applications [ZS01]. Babel supports the specification of related applications in terms of the functions they deliver and the data they expect as input and produce as output. Furthermore, it enables the specification of business rules for how these functions should be integrated, which can be seen as a kind of agreement. The business rules are formulated in the XML-based language Rule XML.

**Trading Partner Agreements**

Executable Trading Partner Agreements (TPA) [SDN+00] are contracts that trading partners in electronic commerce have agreed on. The agreements are formulated in the XML-based trading partner agreements Markup Language (tpaML) [Sac00]. The language specification has been submitted to OASIS⁹. TPAs additionally contain policy information for different layers in the protocol stack, whereas the contracts addressed in this work only contain information of the application layer. The TPAs contain an agreement on functionality and services that the trading partners offer to each other. Rather than agreeing on usage rights over digital goods the partners agree on predefined and implemented procedures, such as "reserve hotel" that may be called by the remote trading partner.

---
Contract Aware Components

Beugnard et al. [BJPW99] introduce a general model of software contracts that aims at increasing trust and reliability between software components. To conclude contracts between components, every component publishes a feature set to describe its services in a common language (e.g. CORBA IDL). Contracts are established between a client and server component in a negotiation phase in which the contract parties agree on certain services. The work provides a basic interface description for the negotiation phase. Beugnard et al. suggest an “XML-formatted description of the contracts” that is applied for negotiation (i.e. interoperability) purposes.

Electronic Contracts Used for Workflow Management

Crossflow\(^\text{10}\) is a European Community research project into support for cross-organisational workflow management in virtual enterprises. In the project electronic contracts are used to define transactions between automated systems [KGV00]. The automated systems on both sides are Workflow Management Systems (WFMSs), extended with contract handling facility that supports outsourcing interaction. On the one side there is a consumer WFMS that desires to outsource a process, on the other side there is a provider WFMS that is capable of executing this process for the consumer.

![Figure 4.18: Simplified model of contracts applied in a WFWM [KGV99]](image)

The contracts that are used in this project are designed to be deployed in a workflow management application. Therefore, the five basic contract

\(^{10}\text{See: http://www.crossflow.org/}\)
elements that are identified describe all aspects of automatic workflow (see Figure 4.18): the concept model defines and assigns values to all objects that can be parameterised in the contract; the process model describes the internal structure of the workflow process implementing the service; the enactment model, describing details of the enactment; the usage model, defining manners in which the contract can be used; the natural language description, being a piece of text that is not meant for electronic interpretation but describes the service in an easily understandable way and refers to the legal context of the transaction. These basic contract element differ greatly from the general core contract objects identified in Section 4.4.1, because they serve one specific application. Such specialised contracts are hardly applicable in other usage scenarios (see Section 4.4.2). Within the scope of the same project the states of such workflow contracts have been identified [KGV99]. The identified states are: contract template, contract advertising template, contract search template, contract instance, and partially filled contract. These states occur in the first two phases of the contract life cycle, defined in Section 4.1; contract execution and contract archiving are not addressed in detail.

Legal Aspects of Electronic Contracts

The work of Gisler et al. [GSSG00] considers the business phases and the legal phases of digital contracts. Considering the precedent actions between the later contract parties and not addressing the archiving of electronic contracts, their model results in four phases that differ from the approach presented in this thesis: Information Phase (Contract Conception), Intention Phase (Contract Preparation), Agreement Phase (Contract Negotiation), and Settlement Phase (Contract Fulfillment). The first two phases of their model do not represent contract states that are legally relevant, and the contract archiving is not considered. Therefore, I disagree with contract life cycle approach.

Business Contracts

In a very early work Milosevic and Bond make an initial attempt to address electronic contract issues [MB95]. They introduce a generic business contract framework comprising the issues Contract Domain, Contract Template, Contract Negotiation, Contract Validity, Contract Monitoring, and Contract Enforcement, and apply these measures to contracts on the Internet. As basic elements in contract templates they identify the roles of the parties, the period of the contract, the nature of consideration (resource),
the obligations associated with each role, and the domain of the contract. The generic business contract framework of Milosevic and Bond comprises legal and technical issues which they do not clearly distinguish. The basic contract elements are not described in detail and differ from the approach shown in this thesis (see Section 4.4.1). Roles are considered as one means for party identification, and the period is one of the several relevant contract attributes. In contrast to this approach I regard the contract domain as something that influences the contract content but is not part of the contract.

In a later work Milosevic et al. address Business Contracts for B2B [GHM00]. Here, they redefine the basic elements of business contracts, name typical contract phases in the B2B business, and present some implementation experiences. In this work a contract modelling approach is introduced that allows for the contract components: a preamble (involved parties), an approval section (enumerating those who have approved the contract), digital signatures, contract clauses, and policy specifications. This approach differs from my approach by formulating the entire basic contract information in the policy specification. The policy specification comprises the contract party information, contract permissions, duties, and constraints, and relates this information to programming logic (instead of a markup language). The formulation of the policy specifications has been influenced by the Event–Condition–Action paradigm from active databases and the ODP enterprise language. The latest work of Milosevic et al. introduces the requirements for a Business Contract Language [NCL+03] and addresses discretionary enforcement of electronic contracts [MJP02].