# Communication Design for Electronic Negotiations on the Basis of XML Schema

Michael Ströbel
IBM Research
Zurich Research Laboratory
8803 Rüschlikon, Switzerland
+41-1-724-8226
mis@zurich.ibm.com

## ABSTRACT

Representation of negotiations in electronic markets and their support are important issues in today's e-commerce research. Whereas most activities are focused on automation aspects, only few efforts address the design of electronic negotiations – e.g. the sequence of actions, or obligations and responsibilities of the negotiating parties. However, an explicit negotiation design can also address what is commonly referred to as the ontology problem of electronic negotiations: how can one ensure that the negotiating parties have the same understanding regarding the issues that are subject to the negotiation?

The solution this paper proposes is to perform a communication design for electronic negotiations that explicitly specifies the common syntax and semantics of the negotiating parties, the logical space of the electronic negotiation. Furthermore, XML Schema is suggested as the mechanism for the runtime representation of the logical space and the validation of actual negotiations from a syntactical and semantical perspective. On the basis of this approach, organisations creating an electronic market or sellers who intend to offer their buyers the ability to bargain can design and generate support mechanisms for electronic negotiations in a flexible and efficient way. The communication design action- and meta-model presented are part of SILKROAD, a design and application framework for electronic negotiations.

## Categories and Subject Descriptors

D.2.10 [Software Engineering]: Design
D.2.2 [Software Engineering]: Design Tools and Techniques – *computer-aided software engineering, state diagrams.*
H.2.11 [Information Systems]: Logical Design

## General Terms

Design

## Keywords

Application Framework, Electronic Negotiation, Ontology, XML

## 1. INTRODUCTION

Let us assume that a new electronic market for multiple sellers and buyers is being created. Due to the nature of the goods traded, price-focused coordination mechanisms such as auctions are not applicable because an agreement between a seller and a buyer has to consider multiple attributes of the good or item (e.g. price and quality) as well as the terms and conditions of the transaction (e.g. delivery time and return policy).

A critical factor for the efficiency of the future negotiation processes on this market and the success of the potential settlements is an a-priori agreement among the negotiating parties about how the issues of a negotiation (item attributes, transaction terms and conditions) are represented as abstract objects in the negotiation and what this representation means to each of the negotiating parties. If, for instance, party X offers a delivery date of '12/10/2000' for a workstation to party Y, one potential conflict arises if this syntax is misinterpreted by Y as 'October 12' whereas X intended to offer 'December 10'. A semantical problem could occur if the meaning of this date to X is the point in time where the product will leave the premises of X, whereas Y assumes this is the date the workstation will arrive on the premises of Y. This problem is referred to as the ontology problem of electronic negotiations [1].

Like any other information system, the creation of an electronic market can be structured along the system development phases of analysis, design and implementation. For an electronic market intended to support electronic negotiations, the design activity has to comprise the agreement scenario, which defines how potential trading partners reach an agreement if conflicts arise regarding the transaction or item configuration. Choice and further specification of this scenario will vary depending on the market requirements identified in the analysis phase. In the implementation phase, the agreement scenario is mapped to a technical architecture and application system.

However, if the agreement scenario is supposed to include some kind of negotiations between buyers and sellers, there are no common means by which the market creator and its stakeholders can reason about the potential form of these negotiations. In 1991, Holsapple et al. [10] have identified this need for general models of negotiations, which could be used to characterise the nature and process of the negotiation as well as to formalise its aspects, and which have the flexibility to describe a wide range of possible structures and interactions. But modelling aspects have been largely neglected in related research, with the undesirable consequence that it is difficult to discuss agreement scenarios on a conceptual level, and that design efforts cannot be reused and refined in the implementation phase in a formal way.

This lack of support for the design of agreement scenarios is the underlying motivation for SILKROAD – a design and implemen-


*WWW10*, May 1-5, 2001, Hong Kong.

tation framework for electronic negotiations. The SILKROAD framework can be used, for instance, by organisations creating electronic markets, for the design and implementation of electronic negotiation support. Two deliverables of this project, the design action- and meta-model for the specification of the common object syntax and semantics in an electronic negotiation, are presented in this paper.

After referring to theoretical foundations of this work in the next section, the approach chosen for SILKROAD will be illustrated in more detail in Section 3. Details of the communication design approach are presented in Section 4. Following the communication design in SILKROAD, the integrated design of agreement scenarios is outlined in Section 5. The consecutive generation of XML schemata for the runtime representation of logical spaces is then demonstrated in Section 6. Lastly, Section 7 discusses conclusions, as well as related and future work.

## 2. NEGOTIATION MEDIA

In SILKROAD, the notion of media and the media reference model [19] are used to conceptualise electronic negotiations. Media are platforms where the exchange of tangible or intangible items by means of transactions is coordinated through agent interaction. These platforms can be described in terms of three main components:

- Channels:
  Agents access a medium via channels that can transport the items to be exchanged.
- Logical space:
  The syntax and semantics defined for representing the items, which the agents exchange.
- Organisation:
  Roles describing the types of agents and protocols specifying their interactions.

An electronic medium in particular is a medium with electronic (digital) channels that transport data. The agents, however, might still be humans or organisational units and do not necessarily have to be software agents.

The media reference model identifies several phases of agent interaction (see Figure 1). Offers, expressions of will concerning the configuration of a transaction or its associated item(s) communicated to other agents, are one possible means of representing this interaction. Depending on the actual phase transition, offers may assume different states of formality:

- Advertisement
  In the knowledge phase, agents gather information concerning the items offered or the profiles of other agents. An offer in the form of an advertisement can be issued in the knowledge phase. This advertisement might relate to a general class of items (e.g. the types of products or services offered by this agent) and is typically not related to another offer from a different agent, but targeted at a group of potential trading partners. An advertised offer is also persistent in the sense that it is valid for a certain period of time.

- Bid
  In the intention phase, demand and supply are specified. An offer in the form of a bid can be a response to an advertisement in the intention phase of an electronic transaction, and is therefore specific to the transaction and item configuration proposed in the advertisement. Bids might also result from an advertisement, which spawns bids specific to received requests. This is, for instance, often the case if the item is configurable or has certain options. In such an example, an interested agent might bid to buy an advertised item with certain options and the advertisement 'generates' a complementary bid with a total price for this choice of item options. The validity of a bid is limited by the validity of the associated advertisement or complementary bid, but is usually even constrained further (e.g. 'please respond to this bid by…').
- Contract
  As a result of a successful agreement phase, a final offer in the form of a contract can seal mutually accepted bids with legally binding signatures of the agents. A contract marks the transition to the settlement phase where the agreed-upon transaction is executed and is therefore persistent beyond the duration of the agreement phase.

A negotiation takes place in the agreement phase when, based on the offers (bids) made in the intention phase, an agreement (a contract) cannot be reached, or the initial agreement has a potential for optimisation and the agents are willing to discuss their offer positions. From the perspective of one agent, negotiating is characterised by the modification of its own bid or the efforts to change another agent's bid.

An electronic medium supporting negotiation processes in the agreement phase, is denoted an Electronic NegotIation MEdiuM (ENIMEM). An ENIMEM provides electronic negotiation support, meaning the assistance or automation of certain tasks (e.g. decisions) within the negotiation process. If a negotiation process is conducted using an ENIMEM and no other media (e.g. letters), an electronic negotiation takes place. Depending on the level of support provided by the medium, electronic negotiations can be completely, or partly automated – the latter case requires human intervention in the negotiation process.

A magnitude of technologies can be used to build electronic negotiation media. These technologies are core elements of development efforts that have historically come to be known as negotiation support systems (NSS, [11]). The notion of electronic negotiation media comprises NSS as services on the transaction layer of the media reference model (see Figure 1). In addition to this service level, the goal of an ENIMEM is to support negotiations in the agreement phase of electronic transactions also from a community, process, and infrastructure point of view.

The ENIMEM definition used in this proposal does not refer to negotiation media, which support agreements in electronic markets, but do not specifically provide assistance for negotiation processes. A medium might, for instance, support agents in legally accepting fixed offers with only one mechanism – signature validation. Hence, contractual obligation can be created and the agreement phase can be completed without any actual negotiation taking place. An ENIMEM might offer the same signature validation, but also has to include support for some form of negotiation mechanisms, e.g. auctions.

Finally, negotiation support is not restricted to electronic media. If a human mediator joins the negotiation process, for instance, to suggest an agreement in a dispute, this constitutes as well a negotiation support activity, but not a form of electronic negotiation support.

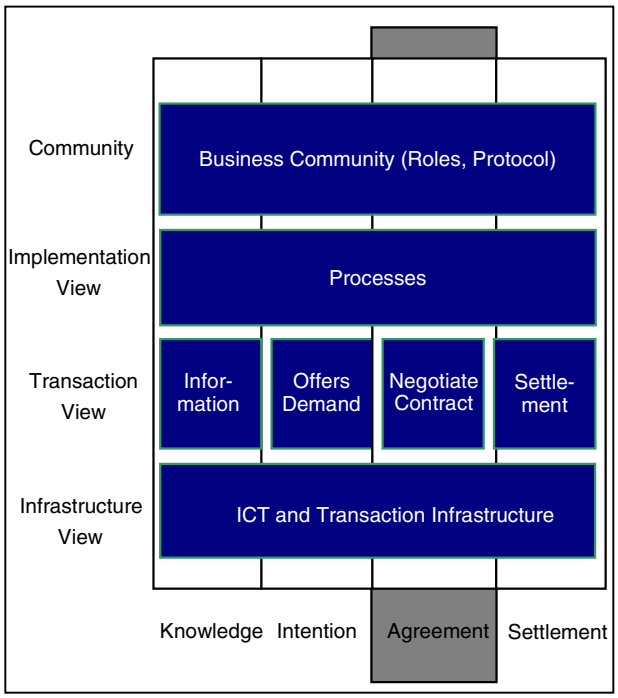


**Figure 1: Agreement phase in the media reference model [19].**

## 3. SILKROAD APPROACH

The primary goal for the SILKROAD framework is to facilitate the design and implementation of electronic negotiation media according to the definition discussed in this section.

The two core elements of SILKROAD are the ROADMAP and the SKELETON. The SKELETON provides several modular and configurable negotiation service components and can be classified as an application framework [8] – the skeleton of an ENIMEM. Hence, an ENIMEM is an instantiation of the SKELETON framework, which supports one or multiple agreement scenarios. Following the reuse and 'inversion of control' paradigm of frameworks, SILKROAD developers can subclass framework components to implement specific application logic. But the most common usage of the framework will be the customisation and deployment of ENIMEM instances of the SKELETON. The customisation affects the runtime behaviour of the ENIMEM and is based on specifications generated in the ENIMEM design.

Following the concept of media, the design of an ENIMEM has to encompass three dimensions [20]:

- The communication design provides the structures of the logical spaces of the Enimem – syntactical and semantical representations of the agents, attributes of the items, and the terms and conditions of the transactions.
- The organisational design describes the roles (structure) and protocols (behaviour) of agreement scenarios that will be supported by the Enimem.
- The IT design addresses the architecture of the Enimem, its technical channels and interfaces.

SILKROAD assists all of the introduced design dimensions with the ROADMAP design action-model, which prescribes how the design of an agreement scenario is performed on the basis of the SILKROAD design meta-model (SDMM). Hence, in the case where one ENIMEM should support various agreement scenarios, the ROADMAP design action model has to be applied several times, each time complementing the design of one agreement scenario.

In SILKROAD the complexity of the final IT design and implementation of electronic negotiation media is reduced to a generation of executable agreement scenario representations, which customise the behaviour of the SKELETON negotiation service component instances at runtime.

Before the design action-model can be applied, it is essential to perform an analysis of the preconditions of the agreement phase of the electronic transaction. For the organisation design, characteristics such as the transaction value (high, low, perishable etc.), the risk for the agents involved in this transaction, or the customisability of the item of the transaction have to be investigated in order to select an appropriate design for the electronic negotiation (see, for example, [3]). In addition to the characteristics of the transaction, this analysis also has to cover aspects of the agents' roles (their beliefs, desires, intentions…) as well as the relationships between the agents (dependency, distribution of market power, level of confidentiality, intensity of information exchange, etc.). For the communication design (see below), this analysis needs to identify typical and necessary elements of the logical space, such as standard terms for transactions (delivery time, return policy, etc.) or common representation formats for the transaction items in a certain domain (e.g. computers are always specified on the basis of CPU speed, RAM etc.).

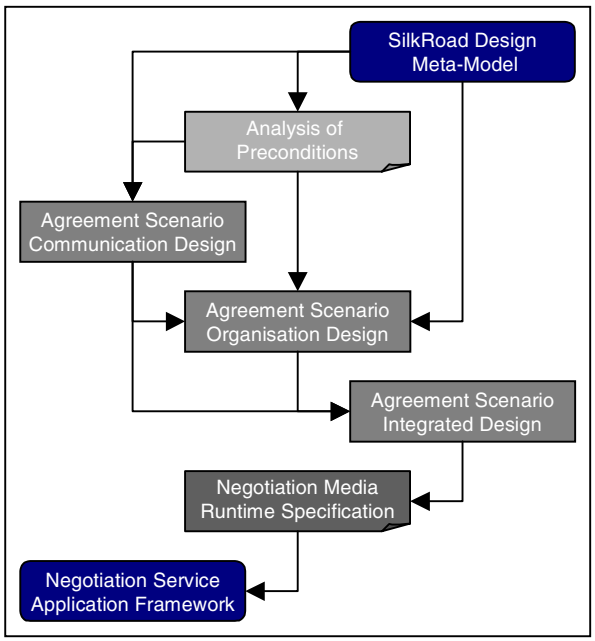


**Figure 2: SILKROAD ROADMAP.**

Figure 2 illustrates the sequence of actions in the design action model and the input/output relations between these actions. The first action to be performed in the ROADMAP is the agreement scenario communication design, which is based on the findings of the analysis of preconditions and the SDMM. Then the organisation design is performed, using the results of the communication design, the precondition analysis and the constructs provided by the SDMM for the organisation design. Finally, in the integrated design activity, the results from the organisation and communica-

tion design are refined, merged, and verified – resulting in one complete and consistent agreement scenario model, which can be used to generate runtime specifications for the deployed SKELETON instance.

Referring back to the layers of the media reference model, SILKROAD specifically addresses the community, implementation and transaction view. The roles and protocols of the community layer are modelled within the SILKROAD design phase. Actual processes on the implementation layer are then executed on the basis of the generated runtime specifications and the negotiation service component instances in the transaction layer.

The basis for all design activities in the ROADMAP is the SILKROAD design meta-model (SDMM, see in Figure 3), which introduces the principal entity types and the relations between these types for the organisation design as well as the communication design.

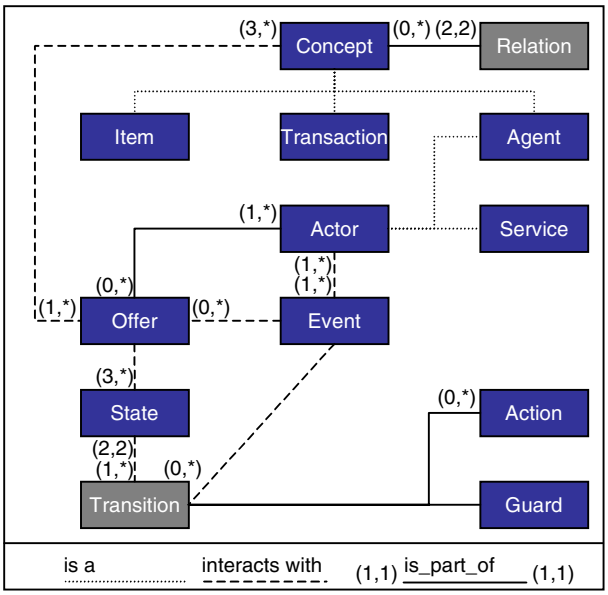


**Figure 3: SILKROAD design meta-model.**

All entity types in the SDMM have associated properties except the *relation* and *transition* types marked in lighter grey, which are used to formalise relations between other entity types.

The SDMM captures both structural and behavioural aspects of agreement scenarios. The semantics of the entity types can be summarised as follows: An *actor* is a *service* or an *agent*. *Items*, *transactions* and *agents* are represented as concepts in an offer. An *offer* has three or more associated *states*. *Actors* create, delete or modify offers and cause *events*, which can stimulate *transitions* between the states of an offer. One *event* can be caused by multiple actors and might be associated with a set of offers. A transition always transfers an offer from one state to another, and will only occur if the *guard* condition is true. The 'firing' of a transition might also invoke an *action*.

The concept of state charts is the underlying modelling paradigm (for both the organisation and communication design). The advantage of state charts is that they are commonly used in information systems design and also part of UML [18]. Therefore it can be assumed that most designers are familiar with this approach.

For the remainder of this paper, the focus is set on the communication design aspects of SILKROAD. Organisation design issues are only referenced if they are coupled to constructs in the communication design. For details regarding the organisation design, see [22].

# 4. COMMUNICATION DESIGN

The goal of the communication design is to structure the logical space of an electronic negotiation medium for a particular agreement scenario. The central objects of the communication design are the offers exchanged in a negotiation. Offer instances are the primary means of communication in the agreement phase (see, for example, [13]) and in the SILKROAD framework are the only supported type of structured interaction.

The SDMM distinguishes between two types of offers that can be issued by agents: offers-to-buy (O2B) and offers-to-sell (O2S). Depending on the agreement scenario chosen, a final contract might require that two compatible offer instances be found that are both signed by the originator with respect to the complementary offer (one-sided contracting), or that one offer instance of one type is signed by both agents (double-sided contracting).

In the ROADMAP the design of offer types is separated into the definition of offer ontologies for the semantical aspects, and the specification of offer states for the syntactical aspects of offer communication in a negotiation.

## 4.1 Offer Ontology Design

The goal of commercial negotiations is to conduct one or more transactions between the agents involved in the negotiation. A transaction transfers one or more items (e.g. a product, money etc.) from one agent to another and vice versa. The transaction, the item, and the agent(s) involved can be described with sets of attributes such as the *delivery date* of the transaction, the *colour* of the item or the *location* of the agent. An attribute has a value or value domain such as '12-12-00', 'green', or 'Switzerland'.

Ontologies are formally specified models of knowledge, which can be used to share semantics among a set of agents. An ontology defines the concepts describing a certain domain and the relationships that hold between them [5]. It can be represented as a hierarchy of concepts. For electronic negotiations in SILKROAD domains have to be specified for the representation of and reasoning about the transaction, its related items, and the agents involved.

Figure 4 illustrates an (incomplete) example of a hierarchy of concepts in the domain of computer hardware items. A *notebook*, for instance, is a sub-concept of a *computer* and accordingly inherits the properties of *computer*, which are, in this example, the CPU clock speed, the type of the media drive etc. Notebooks are also sub-concepts of monitors, thus inheriting another set of properties (e.g. the display resolution). Properties in the ontology have a certain type and can be constrained, thus allowing only certain property values (in the example the CPU clock-speed is constrained to the range between 300 and 1200 MHz). Relations between concepts complement the ontology. An example of such a relation is that the CPU of notebooks has to have power management functions. It is possible to infer new knowledge on the basis of given facts. An agent could derive, for instance, that if a certain CPU is offered with notebooks, it must have power management functions.

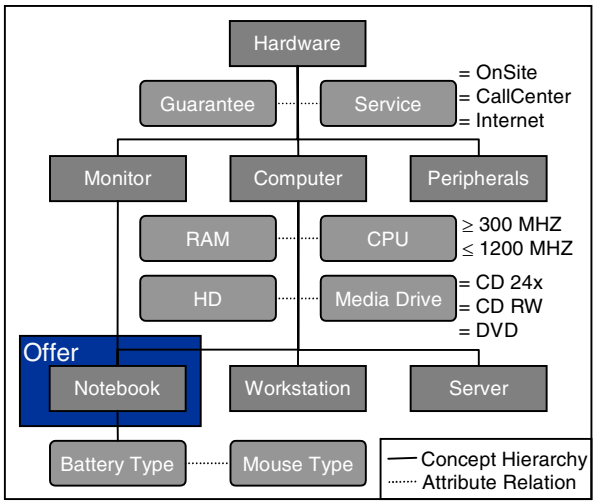


**Figure 4: Ontology example.**

For a complete offer ontology design, this item domain has to be complemented with a domain ontology for the transaction, which defines possible attributes and attribute values for terms and conditions and an agent ontology. Domain ontologies can be reused for multiple agreement scenarios. Hence, the transaction and agent domain ontology in the example could also be used for scenarios designed for other items such as computer software or IT services.

In an offer instance, concepts from the item, transaction and agent domain can or must be used as offer properties to describe the proposed deal completely. The representation of concepts in an offer follows the notation *domain.property* (e.g. *transaction.delivery_date*, *notebook.CPU*, or *seller.location*).

The effort to design and establish an ontology for an electronic negotiation medium can be significant, as agents have to agree (in a social process) on this common terminology (see, for example, [2]). In other words, before ontologies can be used in the agreement phase, the agents have to negotiate on a meta-level the structure, meaning, and content of these domains – their common language. This meta-level negotiation is manifested in the ontologies developed or chosen for the latter electronic negotiation.

## 4.2 Offer State Design

In the offer state design, the dynamic structures of the offer-to-buy and offer-to-sell types for a specific agreement scenario are modelled. From a behavioural perspective, any offer instance in SILKROAD has a certain type and, at least and initially, three different states of formality during the negotiation process: *advertised*, *bid* and *contracted*. In the SDMM, an offer is associated with these three or more states, with one or more actors, and might be related to certain events. To associate a state with an offer, the notation *offer.state* is used.

The basis for the offer state design is a generic offer syntax specification developed for SILKROAD. This syntax defines the notation for structural offer elements such as property domains (e.g. *price < $1000*) or evaluation criteria (e.g. *utility[price,$800] = 0.4*). On the basis of the defined notation, offer instances are created and edited. The notation for property value domains, for example, is the syntax used to represent item, transaction, or agent ontology concepts in an offer instance. In general, the defined notation is not specific to one particular domain ontology but applies to all concepts represented in an offer.

In the meta-model the following abstractions of common offer notation elements with associated sets of notation options are available to represent an offer state:

- Agents (one, n, unbounded)
- Signatures (none, single, all)
- Timestamps (none, start, end, both)
- Domains (properties, values, ranges, dynamic)
- Constraints (basic, negotiable, weighted)
- Counters (none, n, unbounded)
- Criteria (none, importance, utility, functions)

Details regarding the semantics of these notation elements can be found in [23]. The notation options are ordered in the sense that a 'higher' option allows a richer notation. To give an example, the value *dynamic* for the property *Domains* explicitly allows an agent to define the range of values for any property in an offer-to-sell, only if the agent knows more about the agent interested to buy. A typical example can be found in the insurance industry, where quotes are usually dependent on age, medical record, driving experience etc. A more restricted notation would disallow the *dynamic* option and limit offer specifications to a definition of domain *ranges*. Another example is the *negotiable* value for the *Constraints* element. It allows an agent to express the intention to concede this offer property if he/she is compensated with another property, thus enabling tradeoffs between buyer and seller (see [21] for further details).

The specification of the offer state notation is performed on two levels: *required* and *optional* offer notation elements. Generic offer templates for the three introduced states are provided by SILKROAD. The *offer.advertised* state, for instance, is characterised by the offer state notation in Table 1.

**Table 1: State offer.advertised template.**

| Notation element | Level | Option | Modifiable |
|---|---|---|---|
| Agents | Required | One | + |
| | Optional | One | + |
| Signatures | Required | None | + |
| | Optional | Single | - |
| Timestamps | Required | Start | No |
| | Optional | Both | - |
| Domains | Required | Attributes | + |
| | Optional | Dynamic | - |
| Constraints | Required | Basic | No |
| | Optional | Negotiable | - |
| Counters | Required | None | No |
| | Optional | None | No |
| Criteria | Required | None | + |
| | Optional | Functions | - |

These initial offer-state templates are the starting point of the communication design. Depending on the analysis of preconditions, further refinements and adaptations of the notation can be applied. Some scenarios might, for example, require property domain specifications with explicit values or ranges, whereas other scenarios may disallow dynamic property domains. To ensure compliance with the framework, templates cannot be changed arbitrarily; modifiable offer structure properties are explicitly marked (see Table 1 where '+' indicates that a richer

notation might be used and '-' indicates that a more restricted notation is possible).

Additional states might be necessary to model the agreement scenario. These states are added in the organisation and integrated design (see Section 5). For each additional offer state the respective level of formality is also represented by enabling or disabling notation elements for the construction of offer instances.

The final step of the communication design is to assign the offer type with its related state design to domains specified in the offer ontology design. An offer type needs to be associated with at least one item domain, one transaction domain, and one agent domain. Multiple agent domains, for instance, might make sense if certain typical agent types such as distributors or outsourcers participate in a market and their properties might be referenced in an offer. If a concept (e.g. in the item domain a *computer*) has sub-concepts (*workstation*, *notebook*, etc.), the offer can be issued for any of the sub-concepts as well (this functionality is especially useful for advertisements where often general classes of products or services are offered, see Section 2).

This ontology association guarantees that the content of offer instances can be validated not only syntactically, on the basis of the offer state design, but also semantically against the domain specifications in the ontology. Hence, only properties related to the concepts and the concept relations defined can be used in the offer description. If an offer were assigned to the *notebook* concept in Figure 4, it is only possible for the construction of an offer instance to use constraints for item properties related to *notebook*, such as *display resolution* or *CPU clock-speed*.

# 5. INTEGRATED DESIGN

In the integrated design of an agreement scenario, the deliverables from the organisation and communication design are integrated, refined, and verified – resulting in one complete and consistent agreement scenario model. On the basis of this agreement scenario model, runtime specifications are generated, which are used to customise the behaviour of an ENIMEM and to validate actual negotiation processes executed through the ENIMEM.

## 5.1 Integration and Refinement

The basis for the integrated design is the set of offer states defined for an agreement scenario in the precedent design activities. These offer states are the mandatory (and optionally customised) template states (*advertised*, *bid*, and *contracted*) specified in the communication design, complemented by additional states discovered within the organisation design.

The task of the organisation design is to model all necessary states of offer types within an agreement scenario and thereby to discover the associated actor roles, events, transitions, guards, and actions. One agreement scenario completed in the organisation design represents all necessary roles and the protocol for the complete agreement phase specification of a transaction in an ENIMEM. *Roles* are defined as the total of all possible events an agent can or must raise. The *protocol* constitutes all the rules in one scenario, represented by valid states and transitions, which define how agents come to an agreement.

Figure 5 illustrates an example of an organisation design. The graphical notation follows the UML conventions for state-chart diagrams. States are represented by rounded rectangles. The offer type related to a state is indicated with capital letters preceding the state identifier. Transitions are arrows connecting states. Events ('E:'), guards ('G:'), actions ('A:'), and properties ('P:') are specified as textual information complementing the transition arrows.

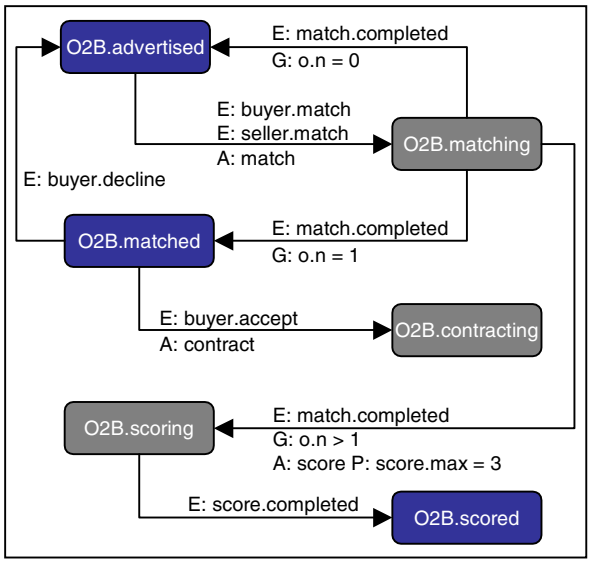


**Figure 5: Organisation design example.**

For the organisation design additional state templates, so-called service-states, are pre-defined (shown in a lighter grey). One of the state templates from the communication design (*O2B.advertised*) is also represented in Figure 5.

The task of the integrated design is to add syntactical structure to the additional states stemming from the organisation design, and potentially to identify supplementary states necessary to represent the organisation design. Depending on the organisation design of the agreement scenario, agents can or can not, for instance, counter the offer of another agent by deriving a new bid that disputes some of the constraints of the original advertisement or bid. In the integrated design this additional offer state has to be reflected with a corresponding offer state representation where the notation element *counters* is set to the *bound* or *unbound* notation option.

The integrated design may result in additional offer states in order to reflect necessary changes to the offer structure. These changes might also require additional agent interaction. In the example in Figure 5, the *score* service can be invoked after an offer instance was matched. This requires the initiating offer to feature evaluation criteria such as utility functions. Therefore, an additional state *O2B.updated* is necessary if an offer in *O2B.matched* does not necessarily contain evaluation criteria. The event activating a transition from *O2B.matched* to *O2B.updated* is *buyer.submitted*. The guard for this transition specifies a successful validation of the modified offer according to the offer structure properties defined for the state *O2B.updated*.

The result of this design activity is an agreement scenario model with offer state specifications, which is complete from a communication and organisation design perspective, thus comprising the logical space (syntactical and semantical representation of items, transactions and agents) as well as the roles and protocols of the agreement scenario.

## 5.2 Consistency Checking

Merging the organisation and communication design in the integrated design phase enables one to check the resulting agreement scenario model for consistency and accuracy from a structural and behavioural point of view.

To be a valid agreement scenario model, the model has to comply with the following types of conditions:

- Offer template states are modified only within the defined restrictions.
- Events with actions activate only transitions to service-states.
- Only service-states and actions available in the application framework are used.
- Guard conditions evaluate only those offer properties and notation elements that are available at the preceding offer state(s).
- Offer notation options required for subsequent service executions are specified.
- Negotiation service component interrelationships are reflected (e.g. match is a necessary predecessor of mediate).

If the agreement scenario model passes the consistency check, the next step in SILKROAD is the generation of executable representations for this design[1].

# 6. Generation of XML Schemata

This section describes how the communication design is transferred to XML schemata, which are used for the runtime validation of offer instances.

On the basis of the completed agreement scenario model, runtime representations for the ENIMEM can be generated. These runtime representations are persisted in communication and organisation design repositories as *agreement scenario policies* (see Figure 6). One ENIMEM can support multiple agreement scenarios, depending on the policies available in its repositories.

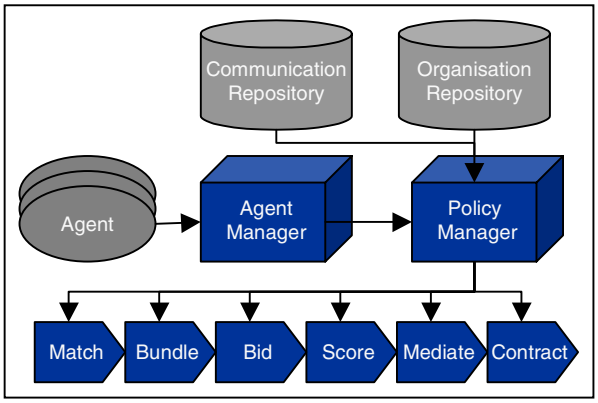


**Figure 6: Runtime architecture overview.**

Electronic negotiation media are instances of the SKELETON. The facility in the ENIMEM responsible for controlling the execution of actual agreement scenarios is the *policy manager*. It checks, depending on the current state of the agreement scenario, offer instances for correctness as well as events and actions of agents for compatibility with the protocol and role specification in the organisation design. Depending on the underlying agreement scenario model the policy manager will also invoke services, if, for instance, a transition fires with an associated action for a negotiation service component. The current set of negotiation service components available within the SKELETON is outlined as well in Figure 6.

## 6.1 XML Schema

XML Schema is a W3C working draft, which was published in April 2000 for review by the public and by the members of the World Wide Web Consortium [7]. In November 2000 it was considered to be stable and promoted to a candidate recommendation.

Schemata are used to specify classes of XML instance documents by describing the document structure in a much richer way than is possible on the basis of document type definitions (DTD) [6]. With the basic vocabulary and predefined structuring mechanisms of XML Schema, fine-grained constraints on XML documents can be defined, thus enabling rich automated validation. The primary advantages of using XML schemata compared to DTDs are that it is possible to express hierarchies of data types, and that schemata themselves are XML documents. Hierarchies of types are critical for the schema generation process in SILKROAD as model-specific types are derived from a set of generic types. Owing to their XML nature, schemata can be created in the same way (with the same tools) as traditional XML documents. Accordingly, it is not necessary to build an automated schema generation process from scratch.

In SILKROAD, schemata represent the logical space design of agreement scenarios at runtime. For each offer state definition in the integrated design a customised schema is generated. If different offer ontology assignments are used for the same offer states, additional schemata have to be generated. At runtime, agents use these schemata to construct or modify offers for the various offer states.

## 6.2 SILKROAD Base Schema

The foundation for the customisation and generation process is the basic SILKROAD syntax. A snippet of the syntax representation in XML Schema, the base schema, is illustrated in Figure 7.

The base schema defines fundamental constraints such as 'an offer needs to have one or more specified item domains'. Overall, the base-schema defines all possible offer notations supported from a structural point of view by the underlying framework. All types in the base schema are declared to be abstract (using the attribute setting *abstract="true"* in the type declaration). Abstract types cannot be used in conforming XML document instances. Hence, all generic types need to be re-defined in the subsequently customised scenario-specific schemata.

[1] Once graphical tools are available to support the design process in SILKROAD, the consistency check can already be performed at design-time, when new states or transitions are added.

```
. . .
<element name="CONTAINER" type="xsr:CONTAINER">
<complexType name="CONTAINER" abstract="true" mixed="false">
  <sequence>
      <element name="AGENT" type="xsr:AGENT"
       maxOccurs="unbounded"/>
      <element name="OFFER" type="xsr:OFFER"/>
      <element ref="xsr:ITEM_DOMAIN" maxOccurs="unbounded"/>
      <element ref="xsr:TRANSACTION_DOMAIN" minOccurs="0"
       maxOccurs="unbounded"/>
      <element ref="xsr:AGENT_DOMAIN" minOccurs="0"
       maxOccurs="unbounded"/>
  </sequence>
</complexType>
<element name="ITEM_DOMAIN" type="xsr:CONTEXT"/>
<element name="TRANSACTION_DOMAIN" type="xsr:CONTEXT"/>
<element name="AGENT_DOMAIN" type="xsr:CONTEXT"/>
<complexType name="CONTEXT" abstract="true" mixed="false">
  <element name="NAME" type="string"/>
  <sequence>
      <element ref="xsr:OFFER_CONSTRAINT" maxOccurs="unbounded"/>
      <element ref="xsr:COUNTER_CONSTRAINT" minOccurs="0"
       maxOccurs="unbounded"/>
  </sequence>
  <attribute name="NUMBER" type="integer" use="required"/>
</complexType>
<element name="OFFER_CONSTRAINT" type="xsr:CONSTRAINT"/>
<element name="COUNTER_CONSTRAINT" type="xsr:CONSTRAINT"/>
<complexType name="CONSTRAINT" abstract="true" mixed="false">
  <choice>
    <element ref="xsr:ATTRIBUTE_DOMAIN"/>
    <sequence>
          <element ref="xsr:ATTRIBUTE_DOMAIN"/>
          <element ref="xsr:ATTRIBUTE_DOMAIN"/>
    </sequence>
    <sequence>
          <element ref="xsr:ATTRIBUTE_DOMAIN"/>
          <element name="CONSTRAINT_OPERATOR"
            type="xsr:OPERATOR"/>
          <element ref="xsr:ATTRIBUTE_DOMAIN"/>
    </sequence>
    </choice>
    <attribute name="NEGOTIABLE" type="boolean" use="optional"
value="false"/>
</complexType>
<element name="ATTRIBUTE_DOMAIN" type="xsr:ATTRIBUTE_DOMAIN"/>
<complexType name="ATTRIBUTE_DOMAIN" abstract="true"
mixed="false">
  <sequence>
      <element name="PROPERTY" type="string"/>
      <element ref="xsr:OPERATOR" minOccurs="0"/>
  </sequence>
</complexType>
. . .
```

**Figure 7: Base schema.**

To generate a state- and ontology-dependent schema, additional constraints are derived from the design specification, which lead to restrictions of the base schema. To restrict a schema, the following generic XML Schema mechanisms are used in the generation process:

- Redefining types.
- Deriving types by extension or restriction.
- Changing attribute *use* from *optional* to *required*.
- Forbidding the use of attributes with *prohibited.*
- Assigning *fixed* values to attributes or elements.
- Setting elements to be required (*minOccurs = 1*).
- Limiting the number of elements (*maxOccurs = x*).
- Deleting enumeration elements in simple types.

In the next sections, the subsequent scenario-specific derivation and customisation mechanism, which underlies the automated schema generation process in SILKROAD, is outlined.

## 6.3 State-dependent Customisation

Whereas the base schema defines a generic namespace *www.silkroads.ch*, a new unique namespace is created for each agreement scenario. Hence, the first step in the derivation and customisation mechanism is to define this agreement scenario namespace.

For all states defined in the agreement scenario model for an offer type, a corresponding state-dependent schema has to be generated that adds the state-specific offer notation to the agreement scenario namespace. This is done by importing all types defined in the generic SILKROAD namespace, and redefining state-specific types according to the notation elements assigned to this state in the offer state design. The process can be illustrated using the example of the state *offer.advertised* as defined in the template (see Section 4.2), which results in the following snippet of an *offer.advertised* schema for a sample namespace www.silkroads.ch/example:

```
<schema
  targetNamespace="http://www.silkroads.ch/example"
  xmlns=http://www.w3.org/2000/10/XMLSchema
  xmlns:xsr=http://www.silkroads.ch
  xmlns:example="http://www.silkroads.ch/example"
  elementFormDefault="unqualified">
  <import namespace="http://www.silkroads.ch"
   schemaLocation="silkroad.xsd"/>
  <complexType name="OFFER" mixed="false">
    <complexContent>
          <restriction base="xsr:OFFER">
            <attribute name="START" type="string" use="required"/>
          </restriction>
    </complexContent>
  </complexType>
. . .
```

**Figure 8: State-dependent schema example.**

In the example in Figure 8, the use of the *START* attribute of the *OFFER* type is *required*, corresponding to the notation element definition in the *offer.advertised* template. Deriving by extension or restriction in XML Schema is comparable to the inheritance mechanism in object-oriented programming languages in the sense that elements and attributes can be added or omitted, and specifications of the super-type can be overwritten. The state-dependent schema redefines only those types, where the offer design for this state manifests specific notation elements. For all other types, the definition in the base schema remains valid.

As outlined in Section 4.2, modifications to the templates can be performed within certain restrictions. If a specific agreement scenario requires, for example, an agent to define an expiration date for the advertisement, the *OFFER* type definition in Figure 8 would also set the use of *END* to *required*. To restrict, for instance, the domain structure to allow no value ranges, all elements of the *OPERATOR* enumeration for a domain ('>', '<' etc.) except the '=' operator are deleted.

The result of this first customisation step is the generation of a set of schemata, one for each offer-state, defining an agreement scenario namespace and constraining XML instance documents from a syntactical perspective. In the next step, semantical constraints are added.

## 6.4 Ontology-dependent Customisation

In this step, the ontology domain assignment for the offer type, performed in the conceptual communication design, is manifested in all generated state-dependent offer schemata. Ontology-dependent offer schemata are constructed using the syntactical notation from a state-dependent schema and the semantical concept specification from the ontology.

For each state-dependent offer schema, this ontology-dependent customisation has to be performed. The state-dependent offer schema is included (using the *include schemaLocation* directive in XML Schema) in a new ontology-dependent schema specification (which shares the namespace with the state-dependent

schema). The base schema is also imported. A designer has two options for the ontology-dependent customisation:

- Domain typing
  This option defines for each property in the chosen ontology domain a new type as extension to the ATTRIBUTE_DOMAIN type.
- Context typing

  This second option adds more semantics through additional extensions of the *CONTEXT* and the *CONSTRAINT* type and the definition of corresponding element substitution groups.

The trade-off between these two options is that domain typing does not guarantee structural integrity – it cannot be validated, for instance, whether an agent used all necessary properties in the specification of an offer for a certain item domain represented as a *CONTEXT*. Context typing, on the other hand, does provide context structure, but the elements used in the specification are not standardised, thus making parsing much more complicated, because every property is represented with a specific domain and constraint element[2].

The first example shown in Figure 9 demonstrates domain typing for the *WORKSTATION.CPU* property, which is restricted to values between 300 and 1200 GHz (see the definition of the *Computer* concept in Section 4.1).

```
<schema
  targetNamespace="http://www.silkroads.ch/example"
  xmlns:example="http://www.silkroads.ch/example"
  xmlns:xsr="http://www.silkroads.ch"
  xmlns=http://www.w3.org/2000/10/XMLSchema
  elementFormDefault="unqualified">
  <import namespace="http://www.silkroads.ch"
   schemaLocation="silkroad.xsd"/>
  <include schemaLocation="silkroad_advertisement_example.xsd"/>
  <complexType name="WORKSTATION.CPU" mixed="false"
   final="restriction">
    <complexContent mixed="false">
      <extension base="xsr:ATTRIBUTE_DOMAIN">
      <sequence>
          <element name="VALUE">
            <simpleType>
              <restriction base="integer">
                    <minInclusive value="300"/>
                    <maxInclusive value="1200"/>
              </restriction>
            </simpleType>
          </element>
          <element name="UNIT">
            <simpleType>
              <restriction base="string">
                    <enumeration value="MHz"/>
              </restriction>
            </simpleType>
          </element>
      </sequence>
    </extension>
    </complexContent>
  </complexType>
. . .
```

**Figure 9: Ontology-dependent schema with domain typing.**

Similarly all other concepts and (inherited or native) attributes from the chosen domain ontology are defined as extensions to *ATTRIBUTE_DOMAIN* types in the state-dependent schema. As the *ATTRIBUTE_DOMAIN* type is declared to be *abstract* in state schemata, only these new semantic domain types can be used for the actual offer specification.

In addition, the ontology-dependent schema declares new types with *final="restriction"*, which prevents further restrictions of this type in new schemata, whereas extensions are still possible (e.g. if an agent needs to extend the MHz range or add *GHz* as another unit enumeration).

```
. . .
<complexType name="WORKSTATION" final="restriction" mixed="false">
    <complexContent mixed="false">
      <extension base="example:CONTEXT">
          <sequence>
            <element ref="example:CPU_CONSTRAINT"/>
            <element ref="example:HD_CONSTRAINT"/>
            . . .
          </sequence>
      </extension>
    </complexContent>
  </complexType>
  <element  name="CPU_CONSTRAINT"
    type="example:CPU_CONSTRAINT"
     substitutionGroup="xsr:OFFER_CONSTRAINT"/>
  <complexType name="CPU_CONSTRAINT" mixed="false">
    <complexContent mixed="false">
      <restriction base="xsr:CONSTRAINT">
          <sequence>
            <element ref="example:WORKSTATION.CPU"/>
          </sequence>
          <attribute name="NEGOTIABLE" type="boolean"
           use="optional" value="false"/>
      </restriction>
    </complexContent>
  </complexType>
  <element name="WORKSTATION.CPU"
    type="example:WORKSTATION.CPU"
    substitutionGroup="xsr:ATTRIBUTE_DOMAIN"/>
  </complexType>
. . .
```

**Figure 10: Ontology-dependent schema with context typing.**

The second example shown in Figure 10 illustrates context typing, where additionally the *CONTEXT* and *CONSTRAINT* type are extended and complemented with corresponding element definitions.

Figure 10 demonstrates a customisation example for *CONTEXT* with the type *WORKSTATION*, and for *CONSTRAINT* with the type *CPU_CONSTRAINT*. The semantics of this example is as follows: the *WORKSTATION* type requires that a mandatory constraint be defined for the *CPU* property of a workstation. This *CPU_CONSTRAINT* can substitute any valid occurrence of an offer constraint in an offer instance document. Furthermore, the *WORKSTATION.CPU* domain has to be used in this constraint. The example also demonstrates how types from the base schema (denoted with the *xsr:* namespace reference) and types from the state schema (such as *example:Context*) are combined to construct the ontology-dependent schema.

## 6.5 XML Instance Document Examples

With the completion of the final customisation step in the previous section, the set of ontology- and state-dependent schemata for an offer type is complete and can be used to construct and validate XML offers at runtime. To demonstrate the result of the generation process, Figure 11 features an XML instance document compliant with the ontology schema in Figure 9.

In this example, the property types are specified with the *xsi:type* assignment for the *ATTRIBUTE_DOMAIN* element. It can be seen that only standardised elements such as *ITEM_DOMAIN* or *OFFER_CONSTRAINT* are used, thus simplifying the parsing of instance documents. However, there is no constraint that the *WORKSTATION.CPU* type is required in the offer.

[2] The distinction between domain and context typing is already reflected in the state-dependent customisation. For context typing, additional types from the base schema such as *CONTEXT* and *ATTRIBUTE_DOMAIN* are restricted. The <NAME> and <PROPERTY> elements in these types are not needed, because specific named types such as WORKSTATION.CPU are created in the process of context typing (see below).

```
<CONTAINER
  xmlns=http://www.silkroads.ch/example
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:schemaLocation="http://www.silkroads.ch/example
   silkroad_advertisement_ontology.xsd">
  <OFFER ID="OF_007" TYPE="O2B" SCENARIO="SC_001"
   STATE="ADVERTISED" START="10.01.2001"/>
  <ITEM_DOMAIN NAME="WORKSTATION" NUMBER="1">
    <OFFER_CONSTRAINT>
          <ATTRIBUTE_DOMAIN xsi:type="WORKSTATION.CPU">
            <PROPERTY>"WORKSTATION.CPU"</PROPERTY>
            <OPERATOR>GREATER_THAN</OPERATOR>
            <VALUE>700</VALUE> <UNIT>MHZ</UNIT>
          </ATTRIBUTE_DOMAIN>
    </OFFER_CONSTRAINT>
. . .
```

**Figure 11: XML instance document with domain typing.**

This additional validation can be achieved with context typing and is illustrated in the instance document example in Figure 12.

```
<CONTAINER
  xmlns=http://www.silkroads.ch/example
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:schemaLocation="http://www.silkroads.ch/example
   silkroad_advertisement_ontology.xsd">
  <OFFER ID="OF_007" TYPE="O2B" SCENARIO="SC_001"
   STATE="ADVERTISED" START="10.01.2001"/>
  <ITEM_DOMAIN NUMBER="1" xsi:type="WORKSTATION">
    <CPU_CONSTRAINT>
          <WORKSTATION.CPU>
            <OPERATOR>GREATER_THAN</OPERATOR>
            <VALUE>700</VALUE> <UNIT>MHZ</UNIT>
          </WORKSTATION.CPU>
    </CPU_CONSTRAINT>
    <OFFER_CONSTRAINT>
          <WORKSTATION.HD>
. . .
```

**Figure 12: XML instance document with context typing.**

In the example in Figure 12 the type specification is used to assign the restricted *WORKSTATION* type for the *ITEM_DOMAIN* element, which requires that *CPU_CONSTRAINT* be used. The disadvantage of this added semantic is that an item specification may contain different constraint types: customised constraints (e.g. *CPU_CONSTRAINT*) and generic *OFFER_CONSTRAINT* elements, which are used, for instance, to define additional binary constraints.

# 7. CONCLUSIONS

This paper demonstrates how the communication design of electronic negotiations is performed within the SILKROAD framework. The goal of the communication design is to define agreement scenario models for the logical space of an electronic negotiation medium. This logical space comprises the syntax and semantics of offer representations shared by agents, which negotiate item attributes and/or terms and conditions of an electronic transaction. The proposed solution is intended to avoid misunderstandings during the negotiation process, before an agreement is made and the settlement is enacted.

In this final section, the proposed solution is evaluated and compared with related research efforts.

## 7.1 Evaluation

Referring back to the initial claims, an evaluation of the presented communication design approach has to discuss two interrelated questions:

- Can the ontology problem of electronic negotiations be addressed by the proposed solution?
- Are XML Schema mechanisms useful for expressing and validating the communication design at runtime?

The result of the explicit communication design of electronic negotiation media within the SILKROAD framework is an ontology for the item, transaction, and agent domain, and state specifications for offer instances associated to these domains. To achieve a common understanding of the issues that are subject to the negotiation, these design deliverables can be specified in a joint process with all agents involved in the later usage of the ENIMEM. The constructs introduced in the SDMM (ontology definitions and state diagrams) support this meta-level agreement process, as they can be used for communication and discussion on a conceptual level. The resulting formal agreement about the semantics of offer representations is a necessary prerequisite for the latter negotiation support implementation.

Once the communication design has been mutually accepted, it can be transferred to a runtime representation, thus enabling the checking of a negotiation process for semantical and syntactical correctness towards the original design. Hence, assuming that both the communication design and the generation of the runtime representation are complete and correct, the ontology problem cannot occur during actual negotiation processes as violations of the agreed-upon logical space are detected. This is at least true for the agents originally involved in the design process. Accordingly, the admission of new agents to participate in the ENIMEM requires an acknowledgement of the logical space defined.

Whether the runtime representation is complete and correct depends largely on the mechanisms provided by XML Schema in association with the defined generation process. Various suggestions (see, for example, [5]) have been made to move from specific ontology formalisms (KL-ONE, KIF, frame logic…) towards more standardised and widely used representation mechanisms such as UML or XML document type definitions (DTD). The latter approach was chosen by Erdmann and Studer [6]. They point out that transforming ontologies into XML Schema appears to be more appropriate than into DTDs, mainly because of the ability to define type hierarchies. In [15], a process for the stepwise translation of an ontology to XML Schema is proposed. SILKROAD uses a similar abstraction-based approach, but in comparison, the communication models do not represent a complete ontology in a schema but only the selected set of concepts.

Regarding completeness, the status of the current representation is still insufficient. On the basis of domain typing, the relation of properties to concepts is lost if multiple concepts are represented in one ontology-state schema. This might be the case if an agent intends to issue a combinatorial offer for several types of goods (e.g. notebooks and servers). Related to this problem is also the fact that multiple inheritance cannot be represented in XML Schema. This is one of the shortcomings of the current framework, which has to be tackled in future work.

Beyond the completeness and correctness necessary to address the ontology problem, the usage of XML Schema provides additional advantages. As a forthcoming W3C standard, a number of powerful and widely accepted tools such as the Xerces parser [26] can be used to create or validate XML documents adhering to this standard. Hence, agents can easily interface with an ENIMEM by submitting XML documents. These documents can be edited, administered, and validated decentrally according to the internal processes of the agent's organisation. Though this creates a distributed and decentralised system of negotiating agents, common integrity constraints are defined centrally using schemata.

XML Schema by means of the control options for the derivation process also offers the ability to extend the ontology in a decen-

tralised way. Let us assume that a seller agent can offer computers with new features not reflected in the current ontology in Figure 4, e.g. a DVD writer. The domain schema specification could then be extended by the agent with a derived *media drive* type, which also includes an enumeration for the *DVD write* option. Using this extension functionality enables the ontology to be maintained in a distributed way. To guarantee the integrity of the overall ontology, the other agents certainly would have to approve such extensions.

Finally, from a technical perspective, the light-weight XML access interface to the ENIMEM, which allows for decentralised schema validation and extension, can be further extended across all functionalities (raising events to execute services etc.) if, for instance, SOAP (Simple Object Access Protocol, see [26]) is used as a general means of service invocation. This option is currently being investigated.

## 7.2 Related Work

This approach relates to work in the areas of negotiation support and semi-structured data models. From a negotiation support view, this work is an effort situated in the area of generalised models of negotiations, which is undertaken from an information systems perspective. Most approaches to modelling negotiations to date stem from an artificial intelligence (e.g. [16]) or decision science (e.g. [14]) background. In addition, the media concept with its explicit distinction in communication and organisation design aspects adds a different perspective on negotiation support. This distinction provides an additional level of abstraction and reduces the complexity of negotiation design significantly.

Approaches to the ontology problem of electronic negotiations that aim at a common understanding of the negotiating parties regarding the question 'what is to be negotiated', can be found in the ContractBot project and in the work of Kang and Lee.

For ContractBot, Reeves et al. [17] developed a declarative contract language that allows one to specify offers and eventually contracts with terms and conditions, constraints, dependencies, as rules and to represent them as XML documents. The expressive power of this contract language certainly exceeds the capabilities of the notation and XML Schema offer representation in SILKROAD: first, because rules have higher semantics than constraints, and second, because these contracts are executable logic programs. SILKROAD, however, does not only provide an offer language, but also a design framework with offer templates and means to model and represent the various states of an offer within a negotiation process. Furthermore, this design framework allows linking the question of 'what is to be negotiated' with the complementary organisation design question 'how is the negotiation executed' through the integrated design activity.

Kang and Lee developed a negotiation support system that relies on a shared ontology mechanism to structure negotiations. Based on the description in [12] buyers and sellers can edit the ontology – but the documentation does not disclose how this ontology is constructed and validated.

Regarding syntax formalisms, related work can be found in the area of XML-based trading protocols such as IOTP [4] or OBI [25]. The difference to SILKROAD is that these protocols are focussed on the settlement phase of electronic transactions (see Figure 1) by providing reference expressions for payment conditions etc. whereas the base-schema in SILKROAD defines generic syntactical structures for the agreement phase, abstracting from the actual message content.

## 7.3 Outlook

Regarding future work, an interesting opportunity arises once the design approach is actually in use and applied to a multitude of real agreement scenarios. Whereas the SDMM specifies 'how' to model electronic negotiation media, a reference model can specify 'what' to model. This reference model could evolve from a set of basic agreement scenarios, which, comparable to proved idioms in object-oriented software engineering [9], represent reusable best practices for electronic negotiations. A communication pattern might suggest, for example, that offers for the domain of *internet services* usually comprise certain mandatory attributes such as the definition of a *support contract* (e.g. 24x7) or the *pricing scheme* (fixed rate, traffic dependent etc.).

Another promising foundation for the definition of communication patterns could be the INCOTERMS and ETERMS repositories (see for example [24]). These collections of standard commercial terms aim at avoiding the friction resulting from the diversity of semantic and legal interpretation of terms in international commerce. For usage in SILKROAD these terms could be represented in generic transaction domain ontologies, defining, for instance, standard concept terms for packaging, delivery points, transits etc.

If this abstraction is feasible, SILKROAD could provide not only a design and implementation framework, but also a reference model for electronic negotiations.

# 8. ACKNOWLEDGEMENTS

The author thanks his colleagues at IBM's Zurich Research Laboratory for supporting this work – Heiko Ludwig and Markus Stolze for valuable feedback on the ideas presented in this paper, and Achille Fokoue Nkoutche for XML Schema troubleshooting.

# 9. REFERENCES

[1] Beam, C., Segev, A., Bichler, M., and Krishnan, R. On Negotiations and Deal Making in Electronic Markets. Information Systems Frontier, Vol. 1, No. 3, 1999, 241-258.

[2] Benjamins, R., Fensel, D., Decker, S., and Perez, A. Building Ontologies for the Internet: A Mid Term Report. International Journal of Human Computer Studies, Vol. 51, 1999, 687-712.

[3] Bichler, M. A Roadmap to Auction-based Negotiation Protocols for Electronic Commerce. Proceedings of the 33rd Annual Hawaii Int'l. Conference on Systems Science HICCS, Hawaii, 2000.

[4] Burdett, D. Internet Open Trading Protocol - IOTP Version 1.0. IETF TRADE Working Group, Internet Draft, 1999.

[5] Cranefield, S., and Purvis, M. UML as an Ontology Modelling Language. Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration, 1999.

[6] Erdmann, M., and Studer, R. Ontologies as Conceptual Models for XML Documents. Proceedings of the 12th Workshop for Knowledge Acquisition, Modeling and Management (KAW '99), Banff, Canada, October 1999.

[7] Fallside, D. XML Schema Part 0: Primer. W3C Working Draft, April 7 2000.

[8] Fayad. M., Schmidt, D., and Johnson, R. Building Application Frameworks - Object Oriented Foundations of Framework Design, Wiley Computer Publishing, New York, 1999.

[9] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns – Elements of Reusable Object Oriented Software. Addison-Wesley, Reading, England, 1995.

[10] Holsapple, C., Lai, H., and Whinston, A. Negotiation Support Systems: Roots, Progress and Needs. Journal of Information Systems, Vol. 1, 1991, 233-247.

[11] Jelassi, T., and Foroughi, A. Negotiation Support Systems: An Overview of Design Issues and Existing Software. Decision Support Systems, Vol. 5, 1989, 167-181.

[12] Kang, J., and Lee, E. A Negotiation Model In Electronic Commerce to Reflect Multiple Transaction Factors and Learning. Proceedings 12th International Conference on Information Networking, Tokyo, Japan, 1998.

[13] Kersten, G., and Noronha, S. Negotiations in Electronic Commerce: Methodological Misconceptions and a Resolution. InterNeg Research Report INR02/99, 1999.

[14] Kersten, G., and Szpakowicz, S. Modelling Business Negotiations for Electronic Commerce. InterNeg Research Report INR98/015, 1998.

[15] Klein, M., Fensel, D., Harmelen, F., and Horrocks, I. The Relation between Ontologies and Schema-Languages: Translating OIL-Specifications to XML-Schema. Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI'00, Berlin, Germany, August 20-25, 2000.

[16] Parsons, S., Sierra, C., and Jennings, N. Agents that Reason and Negotiate by Arguing. Journal of Logic Computation, Vol. 8, No. 3, 1998, 261-292.

[17] Reeves, D., Grosof, B., Wellman, M., and Chan, H. Automated Negotiations from Declarative Contract Descriptions. Proceedings AAAI-2000 Workshop on Knowledge-Based Electronic Markets, Austin, USA, July 2000.

[18] Rumbaugh, J., Jacobson, I., and Booch, G. The UML Reference Manual. Addison-Wesley, Reading, England, 1999.

[19] Schmid, B. Elektronische Märkte - Merkmale, Organisation und Potentiale, in: Sauter M. (ed.), Hermanns A. (ed.): Handbuch Electronic Commerce. Universität der Bundeswehr München, July 1998.

[20] Schmid, B. Was ist neu an der digitalen Ökonomie? in: Dienstleistungskompetenz und innovative Geschäftsmodelle, eds. Belz, Bieger, Thexis Verlag Universität St. Gallen, 2000, 178-196.

[21] Ströbel, M. A Framework for Electronic Negotiations Based on Adjusted-Winner Mediation. Proceedings of the DEXA Workshop on e-Negotiations, London, UK, 2000, 1020-1028.

[22] Ströbel, M. Design of Roles and Protocols for Electronic Negotiations, to appear: Electronic Commerce Research Journal, Special Issue on Market Design, 2001.

[23] Ströbel, M. Intention and Agreement Spaces – A Formalism. IBM Research Report No. 3279, 2000.

[24] Tan, Y., and Thoen, W. INCAS: A Legal Expert System for Contract Terms in Electronic Commerce. Decision Support Systems, Vol. 29, 2000, 389-411.

[25] www.openbuy.org – visited August 28, 2000.

[26] xml.apache.org – visited August 28, 2000.