Inter-organizational Reference Modeling - A Position Statement

Birgit Hofreiter[1], Christian Huemer[2], Gerti Kappel[2], Dieter Mayrhofer[2], and Jan vom Brocke[1]

[1] Institute of Information Systems
University of Liechtenstein, Vaduz, Liechtenstein
{birgit.hofreiter, jan.vom.brocke}@hochschule.li
[2] Institute of Software Technology and Interactive Systems
Vienna University of Technology, Austria
{huemer, gerti, mayrhofer}@big.tuwien.ac.at

Abstract. In our project BSopt we have developed a model-driven approach towards inter-organizational systems. During its evaluation phase we recognized that different partner networks operate similar, but still slightly differ in some details. However, our current approach is limited with respect to re-use of models or parts thereof for different partner networks. Thus, we plan to incorporate well established design techniques from reference modeling into our approach leading to inter-organizational reference models. We believe that inter-organizational reference models will speed up the development and will improve the quality of inter-organizational systems. In this position paper we outline our plans of integrating the design techniques of reference modeling into inter-organizational system development even if we do not provide a solution yet.

1 Introduction and Motivation

Today, discussions on inter-organizational systems are often limited to their implementations, e.g., based on Web Services. However, an approach to inter-organizational cooperation must also address the business issues in organizing and utilizing a distributed solution for a business partnership. Companies do business in order to reach their economic goals. Thus, the management focuses on the value perspective in order to maximize a company's profit. In addition to revenue estimates, a resulting business model - on the first layer of Figure 1 - captures the rational as well as the economic resources being exchanged with business partners.

A business model must be supported by a set of business process specifications. The process perspective manifested in the business process models - on the second layer of Figure 1 - specifies a flow of business activities and their dependencies specially designed to reach the business goals. It is important to differentiate two kinds of business process models: one kind describes the inter-organizational choreography between business partners and the other one specifies the orchestration internal to a business partner. The former serves as kind of

contract between business partners capturing the agreements and commitments on the exchanges between business partners. The latter describes processes to produce and consume the resource/message exchanges - it is a black box to other business partners.

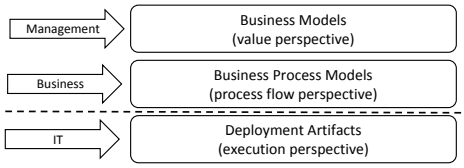


Fig. 1. Layers in Inter-organizational Systems

The business models and the resulting business processes must be supported by IT systems. The IT layer implements the business processes by means of tools, frameworks, API's, Web Services, etc. Instead of hard-coding the business processes into the IT systems, it is desired to configure a business service interface according to a process description. This means a workflow engine is fed with a machine-readable representation of a business process. This machine-readable representation corresponds to the deployment artifact at the third layer of Figure 1. The deployment artifact is consumed by the workflow engine of the business service interface which controls/executes the business process.

In the project *Business Semantics on top of Process Technology* (BSopt) we have developed a modeling approach towards inter-organizational systems that spans over all three layers described above [1]. Instead of starting on each layer from scratch we based our approach on well accepted approaches for each layer and focused on their integration. These approaches - e3-value, Resource-Event-Agent (REA), UN/CEFACT's Modeling Methodology (UMM), Core Components, Business Process Execution Language (BPEL), and Web Services Definition Language (WSDL) - are explained in more detail in Subsection 2.2 and Section 3.

When evaluating our BSopt approach, we recognized that although BSopt allows to develop consistent models across the three layers, it has some limitations with respect to re-use. When developing a new model for a similar case with slight variations, we were not able to properly reuse existing models and had to start from scratch again - except for copying & pasting parts of existing models. However, when a business partner wants to interact with a different set of partners, he wants to build up on existing models. Also partners that have not done the exercise before expect to profit from best practices from other networks (even if they are from other industries).

Accordingly, we claim that deriving a model from another existing model will speed up the development and improve the quality of inter-organizational models. This kind of model re-use is similar to the idea of reference modeling, which has predominantly been driven by the German speaking IS community and which

has mainly focused on business processes [2–4]. Thus, we plan to adapt existing design techniques from reference modeling which have been primarily applied to internal business processes to the domain of inter-organizational systems.

The remainder of this paper is structured as follows: In Section 2 we elaborate on related work. Subsection 2.1 introduces the design techniques known from reference modeling which we plan to adopt, whereas Subsection 2.2 gives an introduction to approaches for inter-organizational systems. The languages that are used in BSopt for modeling inter-organizational systems are briefly outlined in Section 3. Our plans for incorporating reference modeling into inter-organizational systems are detailed in Section 4. The conclusion in Section 5 highlights the advantages we expect by following a reference modeling approach for inter-organizational systems.

2 Related Work

2.1 Reference Modeling

The concepts of reuse has been studied in software engineering for some decades. Early works on reuse-oriented software engineering have been carried out on structuring programs by means of modules [5]. In particular, the concept of generic packages [6] provides inspiration for reference modeling. Generic packages allow reusing a unique data structure for various data types by means of deriving instances of the package for concrete data types. The idea of reuse is essentially incorporated in the object-oriented paradigm [7, 8]. The idea of composing information systems out of rather independent fragments is further developed in the concept of component-based software engineering [9, 10]. Pattern-based software engineering aims at providing solutions for recurring problems that can be reused by customizing them to the application context at hand [11–14].

In reuse-oriented software engineering, the concepts described above have also been covered in the early phases of analysis and design [15–17] and, thus, are incorporated into modeling languages like the UML. The implementation of these principles, however, takes place "within models". For reference modeling, in contrast, these principles are to be applied "between models". Reference models have to be differentiated from meta-models [18, 19] that are commonly used for the definition of modeling languages. Whereas meta-models define languages and their rules for conducting modeling, reference models offer content to be reused. However, it should be noted that meta models may have to undergo a change in order to incorporate concepts of reference modeling.

The idea of reference modeling has been intensively studied by the German IS community. This work has been motivated by the increasing demand of models addressing similar design problems to a certain extent. The essential idea is to provide information models as a kind of "reference" in order to increase both, the efficiency and effectiveness of modeling processes [20–22]. Practical applications of reference models are well-spread in the domain of ERP-Systems [23–26]. In this domain, reference models set the basis for general business solutions that

can be adapted to the individual needs of customers. In order to support this kind of customizing process, reference models are built in a configurative way [20, 27, 28].

Apart from configuration as a parametric approach, a great variety of principles are discussed in software engineering that are originally referred to as adaptive principles [29, 30]. In previous works, we have evaluated these principles for their use in reference modeling [31, 4]. As a result, we have identified a set of reference modeling techniques in addition to the techniques of configuration, namely instantiation, aggregation, specialization, and analogy. A definition of these techniques is given in Figure 2. For each technique, we have defined rules describing the way in which the content of one model is reused in constructing another model. The rules describe ways of taking over contents as well as adapting and extending them in the resulting model.

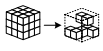
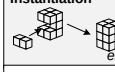
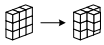
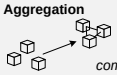
Pattern	Definition	Usage Rule
Configuration  *by selection*	*The technique of configuration is characterised by deriving a configured model c out of a configurative model C by means of making choices from a greater variety of alternatives offered in C.*	*The application domain can be described fully in design time including all relevant adaptations that have to be considered in various applications.*
Instantiation  *by embedding*	*The creation of a resulting model "I" by integrating one or multiple original models "e" into generic place holders of the original model "G". The model "I" incorporates the integrated construction results of "e" in "G".*	*The application domain can be covered by a general framework; this framework however, has to be adapted in regard to selected aspects that can not fully be described while building the reference model.*
Specialization  *by revising*	*Derivation of a resulting model "S" from a general model "G". That way, all statements in "G" are taken over in "S" and can either be changed or extended (but generally not deleted).*	*The application domain can be covered by a core solution; but this solution has to be extended and modified (without deleting) in an indefinite manner for various applications.*
Aggregation  *by combination*	*The combination of one or more original models "p" that build "a" resulting model "T", with the models "p" forming complete parts of "T".*	*The application domain can be described partly; each part can fully be specified whereas their combination for replenishing the entire coverage of an application cannot be foreseen when building the reference model.*
Analogy  *by creativity*	*An original model "A" serves as a means of orientation for the construction of a resulting model "a". The relation between the models is based on a perceived similarity of both models regarding a certain aspect.*	*The application domain can be described by certain patterns recurring in each application; the entire solution, however, has to be replenished in an indefinite manner.*

Fig. 2. Design Techniques of Reference Modeling

In particular, the principles of instantiation, specialization, aggregation, and analogy offer useful means for reference modeling in addition to the principle of configuration. According to instantiation, general aspects of a domain are designed as a framework providing generic placeholders for plugging in models considering special requirements of an application. Specialization enables the takeover of the entire contents of a general model into a specific model allowing individual modifications and extensions. Aggregation enables the takeover of contents delivered by a number of reference models that are composed in and extended according to special requirements of the new model. Analogy, finally, employs seemingly similar solutions in a creative way to tackle new problems.

The design principles are specified on a general methodological level independent of special modeling languages. The design principles have to be transferred to specific modeling languages. So far, in previous work [4] we have stud-

ied the transformation of the design patterns to Entity-Relationship-Diagrams [32] and Event-driven Process Chains (EPCs) [33, 26] since these languages are wide spread in reference modeling. However, we are convinced that the design techniques will also improve effectiveness and efficiency of modeling languages for inter-organizational systems as described in the following subsection. As a first step in this direction, we have already studied the UN/CEFACT modeling methodology (UMM) to detect an appropriate example of each of the design techniques [34]. These preliminary results are promising, but a comprehensive analysis to find all elements of UMM (and of the other inter-organizational modeling languages) that may be subject to reference modeling is still missing.

2.2 Inter-organizational Models

In the introduction we have outlined that an approach to inter-organizational systems has to cover three levels as defined in Figure 1: the value perspective, the process flow perspective, and the execution perspective. On each layer, different means of abstractions are used to capture/implement what an enterprise does or wants to do. Accordingly, each layer has its own specific methods, which are elaborated in more detail in the following.

The top level layer describes how to formalize business models, which we consider as being a layer on top of business process models. There exist several methods to capture and model the economics behind the business process. Approaches to model enterprise networks are Tapscott's business webs [35] and the e-Business Model Schematics [36]. Another approach to formalize business models is the Business Model Ontology (BMO) [37]. However, it does not describe the network constellation from a global point of view. For this reason we opted for following two methods in the BSopt project: e3-Value and REA. The e3-Value ontology [38, 39] is the most advanced approach. It considers a business model as a value constellation, i.e., a network of enterprises that jointly creates and distributes objects of economic value. An approach that goes beyond strategic business models - but which we still consider to be on the first layer of Figure 1 - is the Resource-Event-Agent Ontology (REA). It was originally proposed in 1982 by McCarthy as a generalized accounting model [40]. Today, it captures the declarative semantics of the collaborative space between enterprises from an economic viewpoint [41]. It describes the involved actors, their value exchange, and holds the triggers for economic exchanges by the means of economic events. On the policy level, REA is able to model agreements and commitments on certain economic events between the actors.

On the business process modeling layer a lot of approaches have been developed based on different notations. The Business Process Modeling Notation (BPMN) [42], standardized by the Object Management Group (OMG), incorporates aspects of already advanced modeling notations (e.g. UML activity diagrams, IDEF [43], ebXML BPSS [44], RosettaNet [45], etc.). Another very popular notation are the Event-driven Process Chains (EPC) focusing on control flow dependencies of activities in a business process. EPCs are utilized in the ARchitecture of Integrated Information Systems (ARIS) [46]. In addition

to special modeling notations, UML can be customized for modeling business processes. In recent years a lot of research has been done by extending the UML meta model (e.g. UML activity diagrams). Most of these approaches focus on describing business processes internal to an organization in order to fulfill customer needs (e.g., the UML Profile for Business Process Modeling [47]). In [48] a comparison of different approaches is given.

Comparatively less work is spent on inter-organizational business process modeling. A theoretical framework for the communication between organizations may be seen in the Language-Action-Perspective (LAP) [49]. An approach to model business processes based on LAP is delivered by DEMO [50]. With respect to the notation, we used in BSopt UN/CEFACT's Modeling Methodology (UMM) [51, 52] for modeling an inter-organizational business process from an observer's perspective. UMM is a UML profile which we have progressed at TU Vienna and which we accompany in its standardization process at UN/CEFACT (United Nations Centre for Trade Facilitation and Electronic Business) - a standards body known for its work in the field of UN/EDIFACT and ebXML [53, 54].

Another important artifact at the business process modeling layer are the business documents which are exchanged during the inter-organizational business process [55, 56]. In the BSopt project, we employed the core components technology (CCTS) [57] as proposed by UN/CEFACT. Core components are reusable building blocks for assembling business documents. In order to allow a simple integration into a UML modeling tool of choice we have developed and contributed the UML profile for Core Components (UPCC) [58, 59] to UN/CEFACT.

The third layer describes the deployment artifacts capturing the process specifications in order to make business process descriptions machine-interpretable. Process or workflow engines are fed with business process specifications (mostly based on an XML-based syntax) in order to adapt their behavior as required by the business process. A survey of different XML-based business process languages is provided in [60]. In BSopt we opted for a Web Services implementation based on the Business Processes Execution Language (BPEL) [61] and, alternatively, for a Windows Workflow Implementation [62].

3 A Tour on BSopt

The BSopt project provides a set of domain specific languages (DSLs) to create models for the management layer (e3-value, REA), the business layer (UMM and Core Components) and their transformation to deployment artifacts (c.f. Figure 1. This section gives a brief overview of the supported DSLs by means of a simple example.

Starting with e3-value, Figure 3 shows a value exchange between an actor *Buyer* and an actor *Seller* (A). They are perceived by its environment as independent economic entities engaged in a value exchange. A diagram may consist of a multitude of actors. By exchanging value objects (B), they aim for either

profitability (in case of an enterprise) or economic utility (in case of an end-consumer). Through the black outgoing value port (D) *Money* (B) is transfered through the value transfer (C) from the *Buyer* to the white ingoing value port of the *Seller*. In return the *Buyer* receives the *Good* from the *Seller*. Value ports are grouped by value interfaces (E) which bundle the value objects an actor is willing to exchange in return for other value objects. The scenario path (F) starting at the start stimulus (G) and ending at the end stimulus (H) indicates through which value interfaces objects are exchanged. *AND* and *OR* forks/joins can be used to split/join paths.

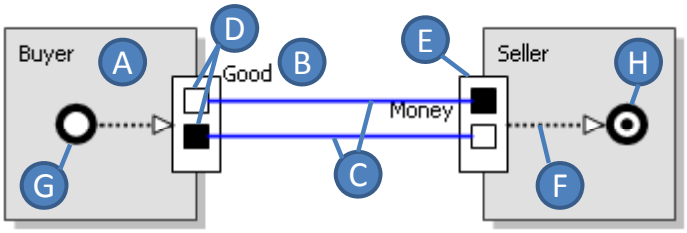


Fig. 3. e3-value example

e3-value has shown the value web between multiple actors. REA defines the bilateral agreements and commitments between exactly two agents. Thus, we derive one or more REA models from an e3-value model, where the resulting agents correspond to the actors in e3-value. REA stands for the 3 core concepts Resource, Event, and Agent. It specifies the economic drivers of the information system from a more IT driven perspective. Figure 4 shows a simple REA example derived from the e3-value example in Figure 3. Two agents *Buyer* and *Seller* (A) exchange a resource (C) triggered through an economic event (B). The arrow (D) indicates from which agent the resource is coming from (fromParticipant) and to which agent the resource is going to (toParticipant). In the example the agent *Buyer* is transfering the resource *Money* to the agent *Seller* through the event *Payment*. If a party receives a resource, he has to give up a compensating resource. In this case the *Seller* returns a *Good* to the *Buyer* through the event *GoodPurchase*. The duality (E) indicates the order of occurrence of the events. The event *Payment* initiates the resource transfer and the event *GoodPurchase* will terminate it.

When moving from the management layer to the business layer, we semi-automatically derive UMM business process models from REA and extend them in further steps. Since we base our approach on a DSL and not on UML, we only keep the UMM concepts, but optimize the presentation by getting rid of UML meta model overheads. Figure 5 depicts an order process from an obeserver's point of view. The *Participants* container (A) holds all participating business partners, in this case the *Buyer* and the *Seller*. These partners can now participate in a transaction (B) as a requesting or responding role (C). The thin arrows

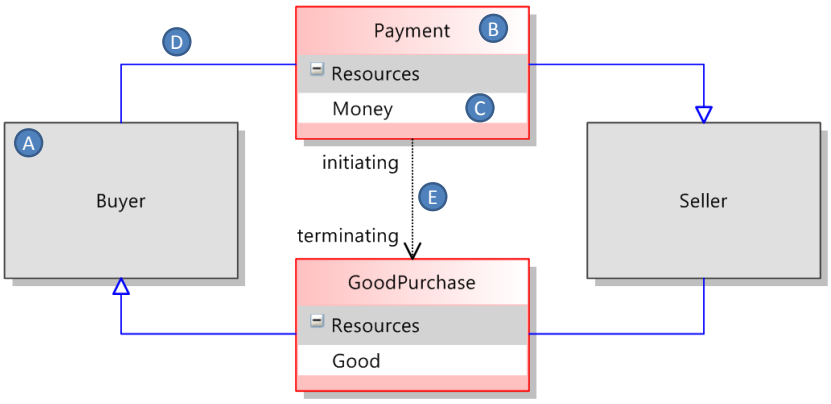


Fig. 4. REA example

(D) indicate whether the transaction is a one-way or a two-way transaction. In the transaction *Request for Quote* the *Buyer* sends a requesting document (E) *QuoteRequestEnvelope* to the *Seller*, who then answers with a responding document (F) *QuoteEnvelope*. Depending on the data in the *QuoteEnvelope* the resulting entity state (G) will be set to either *Quote [provided]* or *Quote [refused]*. The *Quote [refused]* state will lead to a *Business Failure* (I). If the state is *Quote [provided]* the flow (H) will go on to the next transaction *Place Order*. Note, the arrows marking the transition always start from the corresponding entity state. The second transaction *Place Order* is based on the same concepts as the first one and either results in an overall business success or business failure.

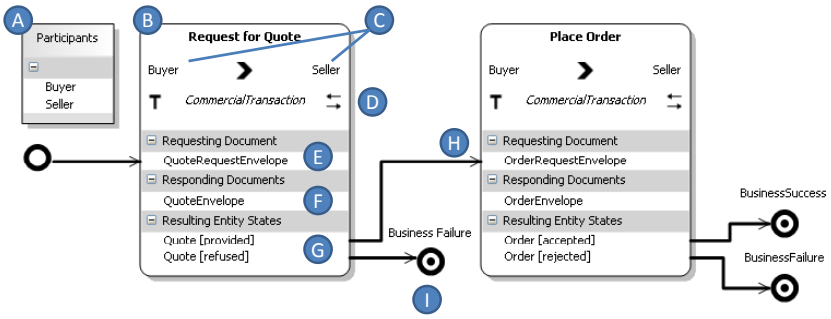


Fig. 5. UMM example

As a last step, the document structures for the messages being exchanged in the business transaction have to be defined. Therefore the Core Components DSL is utilized. In Figure 6 you can see a cutout of the *OrderEnvelope* message. An Aggregated Core Component (ACC) represents an Object Type (A), in this case

a *Person*. The *Person* ACC contains a number of properties (B), so called Basic Core Components (BCC). These properties like *DateofBirth* and *FirstName* are of a special type called Core Data Type separated by a colon. Complex Properties (C) are so called Associated Core Components (ASCC) and can reference other ACCs. Referring to the example, the *Person* ACC contains one *Private Address* ASCC and one *Public Address* ASCC. This structure now defines a part of the message being exchanged in the business processes.

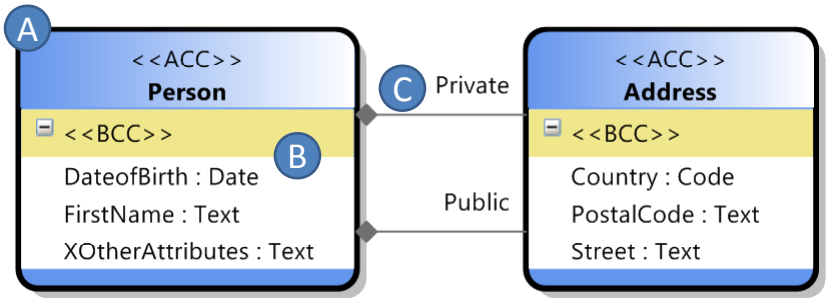


Fig. 6. Core Components example

4 Road towards Inter-organizational Reference Models

4.1 Approach

As outlined in the introduction, the evaluation phase of our BSopt project, which delivered a method spanning over the three layers identified Figure 1, has revealed the need to build models on the basis of other models. For this purpose we plan to follow a Design Science approach [63], which is commonly used in the discipline of information systems (IS). As the area of IS research is at the "intersection" of people, organizations, and technology, Design Science is not only about designing, but also observing and validating "implemented" changes. Design Science identifies design processes (i.e., build and evaluate) and design artifacts. The latter are constructs (e.g., specific vocabularies and symbols), models, methods, and instantiations (e.g., prototype systems). These artifacts enable researchers to understand and to address the problems in developing, implementing, and validating information systems. In addition, this approach names several guidelines such as that the artifacts must be relevant for the problem domain; the evaluation of these artifacts is crucial; research contributions must be clearly identified; and, the research must rely on rigorous methods, both when building and evaluating the constructs.

Given our experience of the BSopt evaluation, we consider inter-organizational reference models as relevant artifacts for our problem domain. When designing

an approach towards inter-organizational reference models we build upon rigorous design techniques known from reference modeling as described in Section 2.1. Accordingly, we research the use of the techniques configuration, instantiation, specialization, aggregation, and analogy for inter-organizational systems. Thereby, we address the management, the business, and the IT layer of inter-organizational systems. Again we base our approach on well established languages for describing inter-organizational systems, which have been introduced in Section 3. In other words, we study the advantages of reference modeling for e3-value and REA describing business models, UMM and Core Components describing inter-organizational business processes and their documents, as well as WSDL and BPEL as relevant deployment artifacts of the Web Services stack. The constructive part as described above has to be complemented by an empirical part to evaluate the research results. Both parts are described in the following subsections.

4.2 Research Contributions

In this subsection we detail the identified set of research contributions to the subject of inter-organizational reference models. In a first step, we have to analyze each of the considered modeling languages - e3 value, REA, UMM, Core Components, BPEL/WSDL - according to their potential to be used in a reference modeling approach. This analysis has to address both, the model and the meta model layer. On the model layer, we start from sets of example models. Each set includes variants of models of the same (or at least a similar) business case. By comparing these models we will be able to identify those elements or set of elements that are commonly subject to variations. In other words, the result are those meta model elements of each language that are candidates to be supported by a reference modeling approach.

In a next step, we systematically analyze the potential of applying the design techniques - configuration, instantiation, specialization, aggregation, and analogy - when transforming a reference model to a new model by customizing the model element instances of a well-defined set of meta model elements. For different parts of a model (i.e., group of related model elements) different design techniques may be applicable, or in certain cases even different design techniques may be useful for the same part of a model. The result of this step are change patterns (based on the design techniques) that are used in the transformation process of reference models. These change patterns have to be specified on a meaningful level of abstraction covering a set of changes on low level of abstraction manipulating single nodes and edges (cf. the change pattern approach of Weber et al. [64] which is specific to workflows only). In summary, our goal is specifying language specific design techniques for each of the different modeling languages.

A resulting model always has to follow the corresponding meta model. However, a reference model may leave some parts open to further specifications in a derived model. These "*under specified*" parts may result in violations against the

meta models. Thus, the meta models of the different languages must be somewhat relaxed to support the reference modeling approach. As a result, we have to deliver meta model extensions/adaptations to support the specification of reference models. Evidently, these extensions/adaptations have to be identified according to the language-specific design techniques.

So far, all of the considered modeling languages have been addressed in isolation of each other. However, a reference modeling approach has to consider cross-layer effects as well. In other words, if a reference model spans over multiple layers and a model on a certain layer is derived from this reference model, the changes may be propagated to the model on the next layer. Such propagations may be derived automatically, but more likely semi-automatically. Even more likely, not all of the changes may be propagated. As a result a well defined set of propagation patterns between the different modeling languages has to be specified. In cases where a classical propagation is not possible, one may further consider a self-learning system to suggest models (or parts thereof) on the next level. This means if a certain characteristic of a model on a certain layer is changed, the system recommends changes on the next layer based on the (manual) changes made by others before.

Our reference modeling approach is going to be demonstrated by a prototype implementation. Thereby, we are able to start from the domain specific language (DSL) tool we have developed as part of the BSopt project. This tool already supports the different modeling languages under consideration, but must be extended to support reference modeling. First of all, the meta model extensions/adaptations have to be implemented. In addition, the identified change patterns must be realized. Evidently, the changes should not be made on a low level of abstraction, but the change patterns have to be implemented by wizards. Note, wizards are considered as one of the strengths of software factories compared to traditional modeling environments.

Finally, we aim at contributing a registry supporting the reference modeling approach for inter-organizational systems. This registry must be able to manage all the different models on each of the three layers of Figure 1. Furthermore, it has to support traceability, both, between models on different layers and between reference models and derived models. Evidently, the registry has to facilitate the search for models (which are treated as opaque, extrinsic objects) based on appropriate meta data, partially extracted from the models. Accordingly, we deliver an appropriate registry information model (extending a more general, existing one) and a set of well specified registry functions supporting the reference modeling approach. In addition, the registry is demonstrated by a prototype implementation, which has to talk to our DSL tool.

4.3 Evaluation

Implementations on a prototypical level will demonstrate the technical correctness of the conceptual models. Proof of correctness which goes beyond the technical one is mainly done in terms of two different kinds of evaluations:

(1) Experiments at academic level. In the first phase, we develop approaches to apply design techniques to the different kinds of inter-organizational models: e3-value, REA, UMM, Core Components, and Web Services (BPEL/WSDL). In order to evaluate the reference modeling approach for each type of model we conduct experiments with students from modeling and e-business related courses in Liechtenstein and in Vienna. We divide the students into two groups. Each student in each group has to solve 5 examples for a certain type of model. Students of the first group start each time from scratch, whereas the ones of the second group apply the reference modeling approach supported by the DSL tool. The evaluation considers the time for creating the models (i.e. the learning curve) and the quality of the models (i.e. the correctness by comparing the models to a given sample solution).

(2) Case Study with the industry. Once the overall approach integrating the different layers and their interdependencies is available, we plan to test it by means of a real world case study. This case study should take place in a company that has experience with inter-organizational systems. Our active participation within UN/CEFACT should guarantee to involve an appropriate company. The evaluation will be based on a questionnaire as well as on face-to-face interviews to consider the opinion of practitioners. This qualitative evaluation has to show whether or not practitioners feel that our approach is superior to what they have done before with respect to creation time and quality of the models.

5 Conclusion

The reference modeling techniques would offer tremendous potentials for the e-business community, both in terms of efficiency and effectiveness. Besides others, this is due to the current approach that is followed by e-business standardization activities. Not only that standards exist only for business documents and to a certain extent for inter-organizational business processes, the standards are specified in a generic way to cover a multitude of industry sectors and geopolitical regions. In order to fulfill this requirement, numerous optional elements are added to the standards. For example, a UN/CEFACT purchase order message consists of 1200 elements and the UBL purchase order includes more than 800.000 elements [65]. However, only less than 5% of the elements are used in a specific business partnership. Unfortunately, an effort to create a message implementation guideline for a specific industry and geopolitical region starts off from the generic standard. Hence, parallel customization activities that do not profit from each other as well as partly overlapping specifications that can not exactly be differentiated from each other are rather common. In this context, we particularly see three major gains by our approach:

Gains to Complexity Management: Today there exists no structured approach to customize standards to the specific needs in a business partner network. This makes the resulting customized specifications hard to compare and to reuse. It is almost impossible to track the usage of certain components in different specifications. In this regard our approach will significantly reduce

complexity by introducing a process of reusing specifications. This allows to compare artifacts of different specifications created by following the proposed design techniques.

Gains to Efficiency: Efficiency gains come into play, as customization initiatives can build on each other. Having the design techniques in place a new customized specification needs not to be designed from scratch. Models developed for a similar purpose in a related industry and/or geopolitical region may serve as a starting point for adapting and extending them to a new industry and/or geopolitical context according to well defined procedures. In addition to referring to fully contextualized models developed by others, one may use reference models that are refined when narrowing down their context. Taking advantage of synergies in developing models for similar business contexts, will result in both, time and cost savings. Since the development of inter-organizational solutions usually takes several months and is a few hundred thousand Euro business, such savings are important for e-business standardization especially in times of more and more frequently changing business requirements and scare resources.

Gains to Effectiveness: In addition, also the quality of inter-organizational solutions may be increased leading to gains of effectiveness. This effect is caused by the continuous evaluation of reference models stored in the registry by multiple stakeholders. The continuous improvement of the models in various revision cycles certainly results in a better quality. Hence, a maximum reuse of inter-organizational models does also raise the quality of those models that are reused. By quality, we understand meeting the actual requirements of business partners by appropriate elements in the inter-organizational models. In fact, generic and, consequently, overloaded standard models provide a lot of space for misinterpretations and misuse of certain (optional) elements. Thus, continuous evaluation of models is a crucial part for e-business standardization guaranteeing harmonization of models towards the intended use of standard elements.

References

1. Huemer, C., Liegl, P., Schuster, R., Werthner, H., Zapletal, M.: Inter-Organizational Systems: From Business Values over Business Processes to Deployment. In: Proceedings of the 2nd International IEEE Conference on Digital Ecosystems and Technologies, IEEE Computer Society (2008) 294–299
2. Fettke, P., Loos, P.: Perspectives on Reference Modeling. In Fettke, P., Loos, P., eds.: Reference Modeling for Business Systems Analysis. Idea (2007) 1–20
3. Thomas, O.: Understanding the Term Reference Model in Information System Research. In: Business Process Management (2005) Workshops, Revised Selected Papers, Springer LNCS (2005) 16–29
4. vom Brocke, J.: Design Principles for Reference Modeling - Reusing Information Models by Means of Aggregation, Specialisation, Instantiation, and Analogy. In Fettke, P., Loos, P., eds.: Reference Modeling for Business Systems Analysis. Idea (2007) 47–76
5. Jones, T.C.: Reusability in Programming: A Survey of the State of the Art. IEEE Trans. Software Eng. **10**(5) (1984) 488–494
6. Slater, P.: Output from generic packages. Ada Lett. **XV**(3) (1995) 76–79

7. Coad, P., Yourdon, E.: Object Oriented Analysis. 2nd edn. Prentice Hall PTR (1991)
8. Cox, B.J.: Planning the Software Industrial Revolution. IEEE Software **7**(6) (1990) 25–33
9. Heineman, G.T., Councill, W.T.: Component-Based Software Engineering: Putting the Pieces Together (ACM Press). Addison-Wesley Professional (June 2001)
10. Szyperski, C.: Component Software: Beyond Object-Oriented Programming (ACM Press). Addison-Wesley Professional (December 1997)
11. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series). Later printing edn. Oxford University Press (August 1977)
12. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional (1995)
13. Fowler, M.: Analysis Patterns: Reusable Object Models. Addison-Wesley, Menlo Park, CA (1996)
14. Hay, D.C.: Data Model Patterns: Conventions of Thought. Dorset House Publishing Co., Inc. (2000)
15. Coad, P., North, D., Mayfield, M.: Object Models: strategies, patterns and applications. Prentice Hall, Englewood Cliffs (1995)
16. Kruchten, P.: The Rational Unified Process: An Introduction. 3rd edn. Addison-Wesley Longman Publishing Co., Inc. (2003)
17. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design. Prentice–Hall, Englewood Cliffs, NJ (1991)
18. Object Management Group (OMG): Meta Object Facility (MOF) Core Specification. (January 2006) Version 2.0.
19. Karagiannis, D., Fill, H.G., Höfferer, P., Nemetz, M.: Metamodeling: Some application areas in information systems. In: Information Systems and e-Business Technologies, 2nd International United Information Systems Conference (UNISCON 2008), Springer LNBIP (2008) 175–188
20. Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuropka, D.: Configurative Process Modeling - Outlining an Approach to Increased Business Process Model Usability. In: 2004 Information Resources Management Association Conference. (2004) 615–619
21. Fettke, P., Loos, P.: Classification of Reference Models: A Methodology and its Application. Information Systems and E-Business Management **1**(1) (January 2003) 35–53
22. Scheer, A.W., Nüttgens, M.: ARIS Architecture and Reference Models for Business Process Management. In: Business Process Management (BPM) 2000, Springer LNCS (2000) 301–304
23. Becker, J., Schütte, R.: Retail Information Systems - Handelsinformationssysteme (in German). 2nd edn. Verlag Moderne Industrie (2004)
24. Huschens, J., Rumpold-Preining, M.: IBM Insurance Application Architecture (IAA). In Bernus, P., Mertins, K., Schmidt, G., eds.: Handbook on Architectures of Information Systems. Springer (2006) 669–692
25. Kittlaus, H.B., Krahl, D.: The SIZ Banking Data Model. In Bernus, P., Mertins, K., Schmidt, G., eds.: Handbook on Architectures of Information Systems. Springer (2006) 723–743
26. Scheer, A.W.: Business Process Engineering: Reference Models for Industrial Enterprises. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1994)

27. Meinhardt, S., Popp, K.: Configuring Business Application Systems. In Bernus, P., Mertins, K., Schmidt, G., eds.: Handbook on Architectures of Information Systems, Springer (2006) 705–721
28. Recker, J., Rosemann, M., van der Aalst, W.M.P., Mendling, J.: On the Syntax of Reference Model Configuration - Transforming the C-EPC into Lawful EPC Models. In: Business Process Management (BPM) Workshops 2005, Springer LNCS (2005) 497–511
29. Karhinen, A., Ran, A., Tallgren, T.: Configuring Designs for Reuse. In: Proceedings of the 1997 Symposium on Software Reusability (SSR '97) , ACM (1997) 199–208
30. Peterson, A.S.: Coming to Terms with Software Reuse Terminology: A Model-based Approach. SIGSOFT Softw. Eng. Notes **16**(2) (1991) 45–51
31. vom Brocke, J.: Reference Modeling, Towards Collaborative Arrangements of Design Processes (in German). Logos Berlin (2003)
32. Chen, P.P.S.: The entity-relationship model—toward a unified view of data. ACM Trans. Database Syst. **1**(1) (1976) 9–36
33. Keller, G., Nüttgens, M., Scheer, A.W.: Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK) . Technical Report 89, Universität des Saarlandes, Germany (January 1992)
34. Hofreiter, B., vom Brocke, J.: On the Contribution of Reference Modeling to e-Business Standardization - How to Apply Design Techniques of Reference Modeling to UN/CEFACTs Modeling Methodology. In: Business Process Management (BPM) Workshops 2009, Springer LNBIP (2009) 671–682
35. Tapscott, D., Ticoll, D., Lowy, A.: Digital Capital: Harnessing the Power of Business Webs. Harvard Business Press (2000)
36. Weill, P., Vitale, M.: Place to Space: Migrating to ebusiness Models. Mcgraw-Hill (2001)
37. Osterwalder, A., Pigneur, Y.: An e-Business Model Ontology for Modeling e-Business. In: Proceedings of the 15th Bled E-Commerce Conference - Constructing the eEconomy. (2002) 1–12
38. Gordijn, J., Akkermans, H.: Designing and evaluating e-business models. IEEE Intelligent Systems - Intelligent e-Business **16** (2001) 11–17
39. Gordijn, J.: E-business Value Modelling using the E3-Value Ontology. In Curry, W., ed.: Value creation form e-business models. Elsevier (2004) 98–127
40. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review **57**(3) (1982)
41. Geerts, G.L., McCarthy, W.E.: An Accounting Object Infrastructure for Knowledge-Based Enterprise Models. IEEE Intelligent Systems **14**(4) (1999) 89 – 94
42. Object Management Group (OMG): BPMN Specification. (2006) Version 2.0.
43. Mayer, R., Menzel, C., Painter, M., deWitte, P., Blinn, T., Perakath, B.: Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report. Technical report (September 1995)
44. UN/CEFACT: UN/CEFACT - ebXML Business Process Specification Schema. (November 2003) Version 1.11.
45. RosettaNet: RosettaNet Implementation Framework: Core Specification. (December 2002) Version 02.00.01.
46. Scheer, A.W.: ARIS - Business Process Modeling. Springer (2000)
47. List, B., Korherr, B.: A UML 2 Profile for Business Process Modelling. In: Perspectives in Conceptual Modeling - ER 2005 Workshops, Springer LNCS (2005) 24–28

48. Korherr, B., List, B.: An Evaluation of Conceptual Business Process Modelling Languages. In: 21st ACM Symposium on Applied Computing (SAC'06), ACM Press (2006) 1532–1539
49. Winograd, T.: A language/action perspective on the design of cooperative work. In: CSCW '86: Proceedings of the 1986 ACM conference on Computer-supported cooperative work, ACM (1986) 203–220
50. Dietz, J.L.: The deep structure of business processes. Commun. ACM **49**(5) (2006) 58–64
51. UN/CEFACT: UN/CEFACT's Modeling Methodology (UMM), UMM Meta Model - FoundationModule. (October 2009) Implementation Verification Draft, Version 2.0.
52. Huemer, C., Liegl, P., Schuster, R., Zapletal, M.: B2B Services: Worksheet-Driven Development of Modeling Artifacts and Code. The Computer Journal **52**(8) (2009) 1006–1026
53. OASIS, UN/CEFACT: ebXML - Technical Architecture Specification. (February 2001) Version 1.4.
54. Hofreiter, B., Huemer, C., Klas, W.: ebXML: Status, Research Issues, and Obstacles. In: 12th International Workshop on Research Issues in Data Engineering (RIDE'02), IEEE CS (2002) 7–16
55. Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M., Flett, A.: Product Data Integration in B2B E-Commerce. IEEE Intelligent Systems and Their Applications **16**(4) (2001) 54–59
56. Guo, J.: Inter-Enterprise Business Document Exchange. In: Proceedings of the 8th international conference on Electronic commerce (ICEC06), ACM Press (2006) 427–437
57. UN/CEFACT : Core Components Technical Specification. (November 2009) Version 3.0.
58. UN/CEFACT : UML Profile for Core Components Technical Specification 3.0. (January 2010) Draft for Version 3.0.
59. Huemer, C., Liegl, P.: A uml profile for core components and their transformation to xsd. In: Workshop Proceedings of the IEEE 23rd International Conference on Data Engineering (ICDE Data Engineering). (2007) 298–306
60. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: Proceedings of Business Process Management (BPM) 2003, Springer LNCS (2003) 1–12
61. OASIS : Web Services Business Process Execution Language Version. (January 2007) Version 2.0.
62. Andrew, P., Conard, J., Woodgate, S., Flanders, J., Hatoun, G., Hilerio, I., Indurkar, P., Pilarinos, D., Willis, J.: Presenting Windows Workflow Foundation. 1 edn. Sams (9 2005)
63. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly **28**(1) (2004) 75–105
64. Weber, B., Reichert, M., Rinderle-Ma, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-aware Information Systems. Data Knowl. Eng. **66**(3) (2008) 438–466
65. Holeman, K.: Contracts - Simplifying the next version of UBL. (January 2006) A mail to the UBL developers list, available at http://markmail.org/message/o3ra6fffiw6mu7jw.