

Co-designing Models for Enterprises and Information Systems – a Case for Language Integration

Peter Rittgen

University College of Borås, 501 90 Borås, Sweden, peter.rittgen@hb.se

Abstract. To achieve a close alignment of information systems and enterprises their designs have to be interwoven to a mutually supportive pattern. This requires compatible languages for expressing the designs. We suggest a framework for integrating two hitherto distinct languages specialized for the respective domain, UML and SIMM. With the help of a case study we demonstrate that this integration does indeed support the co-design of an enterprise model and an information systems model.

1. Introduction

The term co-design can be understood in different ways. In the context of our study it refers to an idea that was elaborated in (Forsgren 2005). It has its roots in “systems thinking” as established by (Churchman 1968). His principal idea was that we can design an unlimited number of views on reality. They may differ in their granularity (level of detail), their perspective, their level of abstraction, and so on. But from Churchman’s point of view this is not sufficient. We must also “calibrate” the viewing instrument (or measurement scale) to arrive at (or agree on) a view that is supposed to be “implemented”. This collective process of designing views and moving towards a consensus view is called co-design. It has shaped the way we look at social systems in general and information systems in particular

(Ackoff 1981; Checkland 1988; Mitroff and Mason 1981). But what are the implications of this idea when the objects of our design are models? We will return to this point later.

When we take a systematic approach to co-design we might ask what the co in co-design refers to. The most obvious answer is perhaps that a number of designers jointly develop a design. We speak in that case of the subjects **in** co-design. But the design process also has a number of customers, the ones we design for. We call them the subjects **of** co-design. If we further broaden our view we might also discover that design is about objects, and again we can identify two dimensions, the objects **in** and **of** co-design. The objects **of** co-design are the things that we design, the results of the design process. But in the process we also need tools or artefacts, the objects **in** co-design.

Our study is primarily about the latter category, i.e. the tools or artefacts. In modeling, the primary tools are languages. But languages have a problematic double role in being both objects **in** and **of** co-design. We use a language to describe, for example an enterprise or an information system. In that sense the language is a tool **in** design. But at the same time, when using the language we discover it new, reinvent it and put it to a different use by reinterpreting existing concepts of the language and creating new ones. Each use situation therefore leads to a change of the language. In that sense the language is also an object **of** co-design.

If languages are shaped by their use, then different contexts of use will also yield different languages. An enterprise modeling language will turn out to be different from a modeling language for information systems because they are used for describing different things. But although an enterprise and an information system are not the same thing, they do share an intimate relationship: The information system (of an enterprise) is a sub-system of that enterprise.

Let us summarize these important points. Languages are both design tools and design objects. As tools for modeling enterprises and information systems they are used on two objects **of** co-design where one is a part of the other. This situation makes the co-design of enterprises and their information systems an intricate business. The separation of enterprise modeling and information systems modeling into different areas of concern has led to the development of completely different languages for the respective areas (see section 2). But a co-design of enterprises and information systems is only possible when the modeling languages for these areas are also co-designed. Both the current situation and the future scenario of co-designed languages are depicted in fig. 1.

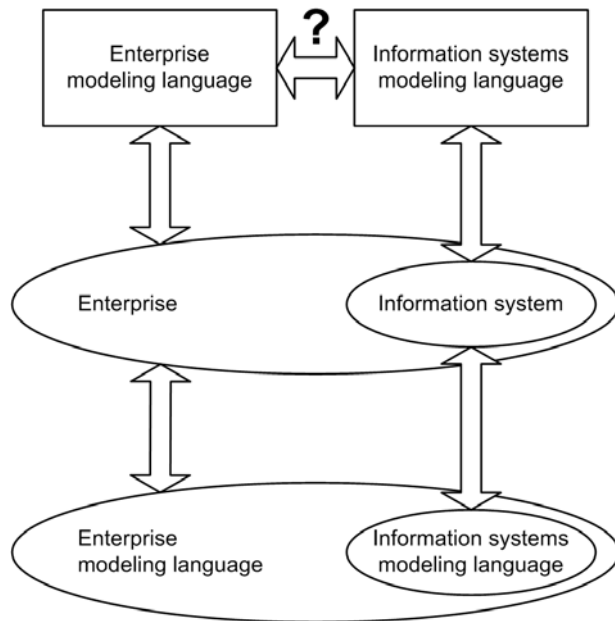


Fig. 1. Co-design of systems and languages

But how do we get started on such a process of co-designing languages? One approach would be to start at zero with “empty” languages, building both languages gradually from scratch in a co-design context, i.e. in projects that apply co-design thinking to develop an enterprise together with its information systems. But much of the time in this process would be spent on discovering concepts that already exist in current languages for enterprise modeling and information systems modeling. We therefore suggest another approach that avoids reinventing the wheel. We propose to investigate existing languages and to identify those concepts that could provide a “common ground” for language co-design. We call this process language integration. It consists of an analysis of the respective languages and their comparison (see section 2). Based on this we derive a framework for integration (see section 3) which we subsequently use for the co-design of business processes and information systems in the context of a case study (see section 4).

2. Language Analysis and Comparison

As candidate languages for our study we have chosen SIMM and UML. UML seemed a natural choice due to its high degree of standardization and its dominance in information systems modeling. Enterprise modeling, on the other hand, is less standardized and our choice of SIMM is hence less obvious. The justification for this particular method lies in a combination of a strong theoretical foundation and a significant empirical support. SIMM has its roots in Business Action Theory (Goldkuhl 1996, 1998; Goldkuhl and Lind 2004) and has undergone considerable empirical validation (Axelsson et al. 2000; Axelsson and Segerkvist 2001; Goldkuhl and Melin 2001; Haraldson and Lind 2005; Johansson and Axelsson 2004, 2005; Lind and Goldkuhl 1997; Lind et al. 2003; Melin and Axelsson 2004; Melin and Goldkuhl 1999).

The analysis of SIMM is based on the language specification (Röstlinger and Goldkuhl 2005). As SIMM is based on the ontology of Socio-Instrumental Pragmatism (SIP) (Goldkuhl 2002; Goldkuhl 2005) we have also made use of this information. Our analysis of UML is based on the UML 2.0 Superstructure Specification (OMG 2004) and the Infrastructure document (OMG 2006). In addition we make use of the ontological analyses of UML done in (Evermann and Wand 2001) and (Opdahl and Henderson-Sellers 2002). They employ the ontology by Bunge-Wand-Weber (BWW) which is an established tool for analyzing modelling languages. It is based on Mario Bunge's ontology (Bunge 1977, 1979) and was later adapted by Yair Wand and Ron Weber to the information systems field (Wand and Weber 1989, 1995; Weber 1997).

As an example of how language analysis and comparison are performed we look at three predominant concepts: Actors, actions and action objects.

2.1. Actors

In theories of social action an actor is always a human being. But SIP recognizes that there can be non-human agency. An artefact (e.g. a computing system) can perform actions and its actions can have a meaning in the social world. Artefacts can play the role of non-human agents, whereas actors are human agents. It is therefore that we speak of a "performer" in the context of SIMM, encompassing human and non-human performers.

In UML the concept of an actor is much broader and covers that of a performer in SIMM plus 'time' which is also an actor in UML (but not in SIMM). We might therefore map SIMM-performer to UML-actor. But in UML the concept of an actor is restricted to the particular context of use

cases. Otherwise actions are considered to be performed by objects. This is due to the fact that the UML is primarily a design language for (software) artefacts. Another possible mapping is therefore the one from SIMM-performer to UML-object which allows for a proper translation of interaction graphs.

2.2. Actions

The concept of action exists in both SIMM and UML and the language descriptions agree largely. It is therefore valid to map SIMM-action to UML-action. It should be noted, though, that SIMM provides a more sophisticated concept of human action as purposeful, social action that is performed with the help of some instrument (artefact), whereas non-human action by artefacts is secondary. In UML these roles are reversed but there is no general conflict (only a shift in focus). SIMM also defines interaction as a special form of action directed towards another actor. Action objects involved in an interaction become interaction objects in that case (see 2.3). This has no effect on the nature of the objects themselves, though.

2.3. Action objects

The term action objects is an explicit concept of SIMM and it refers to objects that are involved in an action. As human action is purposeful it is performed to achieve some result(s). Action objects produced by an action can be such results. But action objects serve also as input for other actions. In the UML objects are a fundamental concept. They constitute a system and all behavior of that system consists basically of messages that are exchanged between objects and the objects' internal behavior. An UML-object can be the resource for another object and thereby play the role of a SIMM-action object. The mapping from SIMM-action object to UML-object is therefore valid. But in addition to that an UML-object can also be the performer of some action as discussed in 2.1.

3. Deriving the Integration Framework

We have extended the comparison described in section 3 to the remaining concepts. For this purpose we made use of the language specification (OMG 2004, 2006; Röstlinger and Goldkuhl 2005) but also of the more detailed information available from the ontological analyses (Evermann

and Wand 2001; Goldkuhl 2002; Goldkuhl 2005; Opdahl and Henderson-Sellers 2002). By following this process we arrived at suitable matches for all SIMM concepts. The resulting mappings are shown in fig. 2 in the form SIMM concept → UML concept.

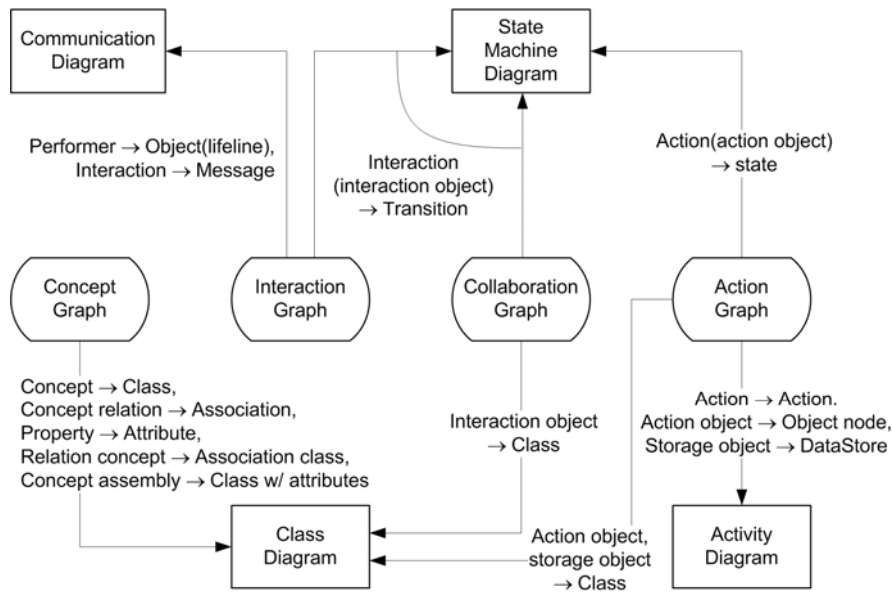


Fig. 2. The integration framework

But the mappings do not only help us in translating concepts from one language to another. The SIMM concepts are related to specific SIMM graphs and the UML concepts are related to certain UML diagrams. If we follow up these relations we can also establish a relation between SIMM graphs and UML diagrams, i.e. we can determine which information from which SIMM graphs we need to construct a certain UML diagram. In other words, we can derive a framework that supports the translation of SIMM models into UML models (see fig. 2).

To test the feasibility of this approach we used the framework in a project where we developed enterprise computing models in UML based on an enterprise model in SIMM. The following section reports on our experiences from that case study.

4. Co-designing Models: A Case Study

The case we investigated involved a retail chain in the home textile and home decoration industry. The logistics operation of that company had been outsourced to a third-party logistics provider. In the beginning of the project we performed an analysis of the business situation that led to an enterprise model that we documented with the help of the SIMM method. Two of the SIMM graphs serve as an example of this model, the interaction graph (fig. 3) and the action graph (fig. 4).

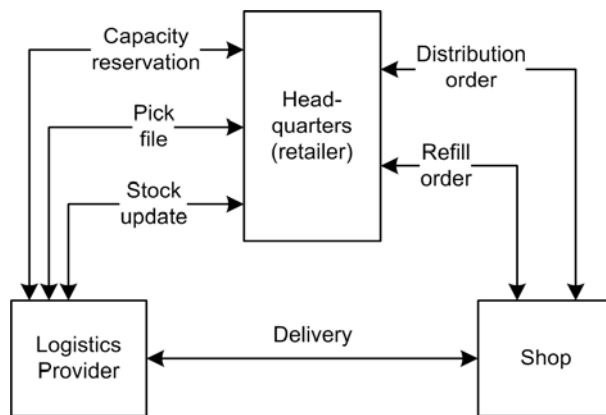


Fig. 3. Interaction graph

The main actors are the Logistics Provider (LogPro), the Headquarters of the retailer and a Shop. Fig. 3 shows the interactions between them with respect to order handling. It starts when Headquarters reserve capacity for handling a certain amount of items in advance of the actual order. LogPro allocates staff and space to provide for this capacity. The product assortment of the retailer consists of basic-range products and seasonal products. The latter are distributed to the Shop according to turnover quota (distribution order). This is triggered by Headquarters. Orders for basic-range products are initiated by the Shop. This happens when the Shop is running low on certain products (refill order). Headquarters forward both types of orders to LogPro in form of a pick list. LogPro performs delivery to the Shop. The confirmation can be accompanied by a complaint if items are missing or wrong ones have been sent. Periodically Headquarters ask for a stock update. This is necessary because they run their own warehouse management system which is not integrated with that of LogPro.

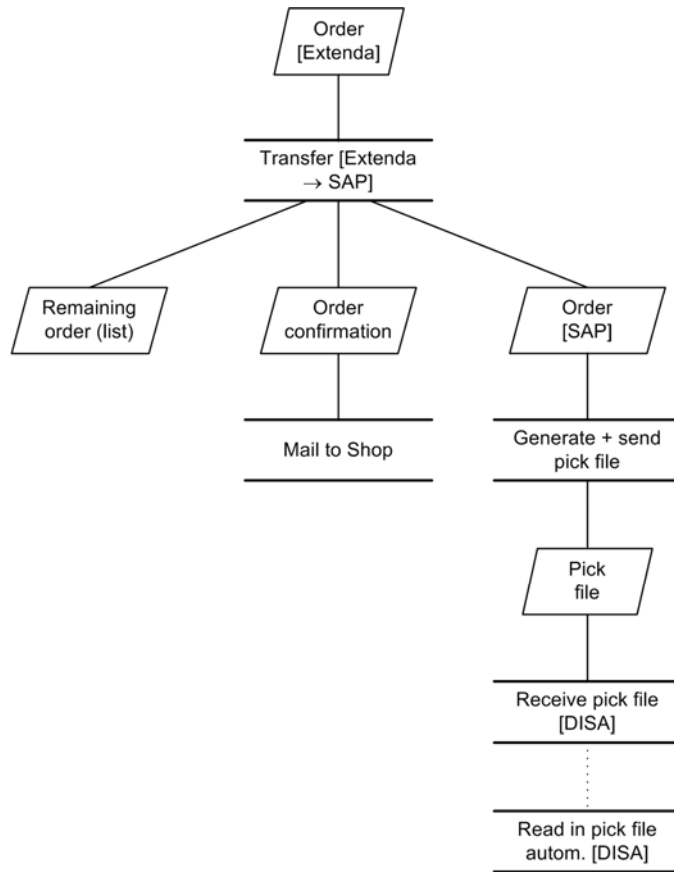


Fig. 4. Action graph

We take a closer look at one aspect of order processing, i.e. processing of the pick file. This routine requires the integration of different computing systems, the Extenda system of the Shop, the SAP system of Headquarters and the DISA system of LogPro. Fig. 4 shows an excerpt of the action graph for pick file processing. Based on the enterprise models we developed the UML models to support the design of respective enterprise computing systems. To give the reader an idea of this process we show the communication diagram (fig. 5) and the activity diagram (fig. 6) that have been derived from the interaction graph (fig. 3) and the action graph (fig. 4), respectively.

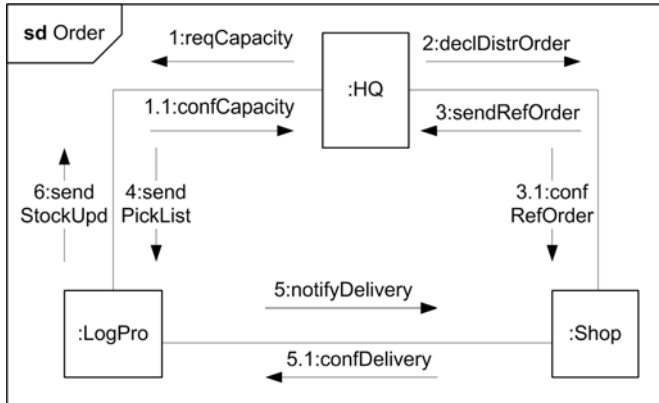


Fig. 5. Communication diagram

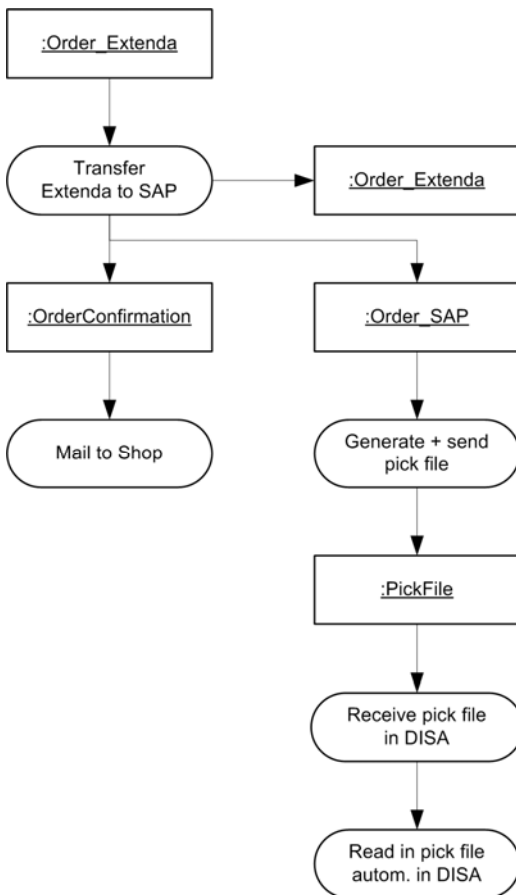


Fig. 6. Activity diagram

All in all the design involved one class diagram, six activity diagrams, one communication diagram and eight state machine diagrams. They were developed by one modeler in 68 hours. In a previous project of similar size the same modeler needed 92 hours without the help of the framework which was not yet developed at that time. These figures seem to indicate that the use of the framework has reduced the effort in that case by 35 %. But this has to be taken with a grain of salt as we cannot be sure that all other parameters were the same in both cases. Further empirical research in that area is therefore necessary.

5. Conclusion

The principal result of our investigation is that the ontological foundations of two particular languages for enterprise modeling and enterprise computing, respectively, exhibit a significant intersection. This common ground allows us to support the translation of enterprise models into computing models in a constructive way. This is done by mapping concepts of the “source language” (here: SIMM) to concepts of the “target language” (here: UML) and by specifying an integration framework that relates the graphs/diagrams of both languages to each other. The latter supports the translation of complete models, diagram by diagram.

We have applied this framework in a project that involved the development of both enterprise models and computing models. We found out that the time for developing the latter decreased by 35 % in relation to time spent on a similar project without the support of the framework.

References

- Ackoff RL (1981) *Creating the corporate future*. Wiley, New York.
- Axelsson K, Goldkuhl G, Melin U (2000) Using Business Action Theory for dyadic analysis. Paper presented at the 10th Nordic Workshop on Interorganisational Research.
- Axelsson K, Segerkvist P-A (2001) Interaction between actors and information systems in web-based imaginary organisations – Experiences from two case studies. Paper presented at the 1st Nordic Workshop on Electronic Commerce.
- Bunge M (1977) *Ontology I: The Furniture of the World*. Reidel, Dordrecht.
- Bunge M (1979) *Ontology II: A World of Systems*. Reidel, Dordrecht.
- Checkland PB (1988) Soft systems methodology: An overview. *Journal of Applied Systems Analysis* 15:27-30
- Churchman CW (1968) *The Systems Approach*. Dell Publishing, New York.

- Evermann J, Wand Y (2001) Towards Ontologically Based Semantics for UML Constructs. In: Kunii HS, Jajodia S, Sølvsberg A (eds) ER 2001, 20th International Conference on Conceptual Modelling, Yokohama, Japan, November 27-30, 2001. Springer, Berlin, pp 354-367
- Forsgren O (2005) Churchmanian Co-design – Basic Ideas and Application Examples. Paper presented at the ISD 2005 - Information Systems Development, Karlstad, August 14-17, 2005.
- Goldkuhl G (1996) Generic business frameworks and action modelling. In: Dignum F, Dietz J, Verharen E, Weigand H (eds) Communication Modeling - The Language/Action Perspective, Proceedings of the First International Workshop on Communication Modeling. Springer, Berlin
- Goldkuhl G (1998) The six phases of business processes - business communication and the exchange of value. Paper presented at the 12th biennial ITS conference "Beyond convergence" (ITS'98), Stockholm.
- Goldkuhl G (2002) Anchoring scientific abstractions – ontological and linguistic determination following socio-instrumental pragmatism. Paper presented at the European Conference on Research Methods in Business and Management (ECRM 2002), April 29-30, 2002, Reading.
- Goldkuhl G (2005) Socio-Instrumental Pragmatism: A Theoretical Synthesis for Pragmatic Conceptualisation in Information Systems. Paper presented at the 3rd International Conference on Action in Language, Organisations and Information Systems (ALOIS), University of Limerick.
- Goldkuhl G, Lind M (2004) Developing e-interactions – A framework for business capabilities and exchanges. Paper presented at the 12th European Conference on Information Systems, June 14-16, 2004, Turku, Finland.
- Goldkuhl G, Melin U (2001) Relationship Management vs Business Transactions: Business Interaction as Design of Business Interaction. Paper presented at the 10th International Annual IPSERA Conference, Jönköping International Business School.
- Haraldson S, Lind M (2005) Broken patterns. Paper presented at the In Proceedings of the 10th International Conference on the Language Action Perspective, Kiruna, Sweden.
- Johansson B-M, Axelsson K (2004) Communication media in distance selling – Business Interactions in a B2C Setting. Paper presented at the 12th European Conference in Information Systems (ECIS), Turku, Finland.
- Johansson B-M, Axelsson K (2005) Analysing Communication Media and Actions – Extending and Evaluating the Business Action Matrix. Paper presented at the In Proceedings of the 13th European Conference on Information Systems, Regensburg, Germany.
- Lind M, Goldkuhl G (1997) Reconstruction of different business processes - a theory and method driven analysis. Paper presented at the In Proceedings of the 2nd International Workshop on Language/Action Perspective (LAP97), Eindhoven University of Technology, The Netherlands.
- Lind M, Hjalmarsson A, Olausson J (2003) Modelling interaction and coordination as business communication in a mail order setting. Paper presented

- at the 8th International Working Conference on the Language Action Perspective (LAP2003), Tilburg, The Netherlands.
- Melin U, Axelsson K (2004) Emphasising Symmetry Issues in Business Interaction Analysis and IOS. Paper presented at the Sixth International Conference on Electronic Commerce, ICEC'04, Delft University of Technology, The Netherlands.
- Melin U, Goldkuhl G (1999) Information Systems and Process Orientation - evaluation and change using Business Action Theory. In: Wojtkowski W (ed), Systems Development Methods for Databases, Enterprise Modeling, and Workflow Management. Kluwer Academic/Plenum, New York
- Mitroff II, Mason RO (1981) Creating a dialectical social science. Reidel, Dordrecht.
- OMG (2004) UML 2.0 Superstructure Specification. Retrieved December 20, 2005, from <http://www.uml.org/>
- OMG (2006) Unified Modeling Language: Infrastructure. Retrieved April 18, 2006, from <http://www.uml.org/>
- Opdahl AL, Henderson-Sellers B (2002) Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model. *Software and Systems Modelling* 1(1):43-67
- Röstlinger A, Goldkuhl G (2005) Grafnotation för SIMM metodkomponenter. VITS/IDA, Linköpings universitet, Linköping.
- Wand Y, Weber R (1989) An Ontological Evaluation of Systems Analysis and Design Methods. In: Falkenberg ED, Lindgreen P (eds) *Information Systems Concepts: An In-Depth Analysis*. North-Holland, Amsterdam, pp 79-107
- Wand Y, Weber R (1995) On the deep structure of information systems. *Information Systems Journal* 5:203-223
- Weber R (1997) *Ontological Foundations of Information Systems*. Coopers & Lybrand and the Accounting Association of Australia and New Zealand, Melbourne.