Knowledge Component-based Architecture for Process Modelling

Olivier Glassey
Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, DE-10589 Berlin
Tel +49 3463 7167, Fax +49 3463 8000
olivier.glassey@fokus.fraunhofer.de

Abstract: In previous work we compared three process modelling techniques and applied them to e-Government processes. We identified major issues, amongst them the lack of formalism for knowledge modelling and of formal semantic links between different abstraction levels. In this paper we propose an architecture for process modelling that supports knowledge models and formally integrates them in process models.

1 Introduction

Public administrations provide many different types of services to their clients, whether they are citizens, businesses, self-employed professionals, other units from the public sector, etc. The interactions between a public agency and its clients are mostly process-based and can be categorised as follow: structured procedures or routines, semi-structured decision processes and negotiation-based case-solving [Lenk & Traunmüller 1999].

Research in the field of business process modelling is quite active, also in the domain of e-government. The use of process models has many purposes. [Wyssusek & al. 2001] identified several of them in the literature and we selected those we found most relevant: **facilitating** human understanding, **communication**, organisational learning and transfer of know-how; supporting **process improvement**; **benchmarking** process **performance**; helping the **integration** of **technical** aspects and **organisational** issues.

In previous work, the author and colleagues [Glassey & Chappelet 2002] studied two specialised process modelling methods: Adonis [Jungiger & al. 2000] and OSSAD [Chappelet & Snella 2004, Dumas & Charbonnel 1990] and furthermore used the UML graphical notation language [Booch & al. 1999] to create corresponding models. Although the main focus of UML is on information systems modelling, we were able to produce equivalent process models in most cases. This comparison aimed at providing a global view of an organisation and wanted to answer three general questions:

- At the abstract level: **what** are the **strategic goals** of the organisation?
- At the structural level: **who** is doing **the work** and **which resources** are available?
- At the descriptive level: **how** does the organisation operate?

We found this approach to be satisfying at the abstract and structural levels, but we discovered that it lacked several important concepts, particularly in terms of knowledge modelling. Indeed these process modelling techniques use concepts such as data, documents or information resources, but they do not take knowledge into account. We therefore tried to integrate a new layer in this general view of an organisation and introduced a new question: **why** is a given decision taken within a process? In this publication we describe how to integrate a knowledge component-based architecture into process modelling.

2 Knowledge Components

We needed a formalism to represent knowledge in a detailed manner and we analysed what was being done in process modelling methodologies. One basic way to represent knowledge in organisations is the use of business rules [Ross 1997]. They can be found in all sectors of activity and do not have to be linked to an information system. Some of them are implicit, meaning that they are not written anywhere but they belong nevertheless to the "business culture". However the basic formalism proposed by [Ross 1997] is not sufficient in all cases for modelling the "know-why". Indeed, a "**knowledge unit**" is anything worth storing that may help things to be done better in the future: help, best practices guidelines, examples, stories, lessons learned, troubleshooting advice or training material [Fraser & al. 2003] and business rules cannot model all of these types of knowledge. A different approach is described by authors such as [Gamper & al. 1999] and [Gruber 1993] that use ontologies (explicit specification of a conceptualization, the latter consisting of identified concepts and relationships assumed to exist and to be relevant) in order to model knowledge. We prefer this method as we previously used RDF to build a data-model for e-Government [Glassey 2004] and found it more powerful and flexible than classical data models such as Entity-Relationship-Model used in ARIS [Scheer 2001] or than business rules. RDF (Resource Description Framework) is a W3C standard for defining metadata and encoding machine-readable semantics [Noy & al. 2000]. It is based on XML and uses graph theory to represent knowledge. It is also a suitable format for specific domain ontology modelling. Fig. 1 shows a basic RDF data model for the registration of a new business, a simple example that we will use throughout this paper.

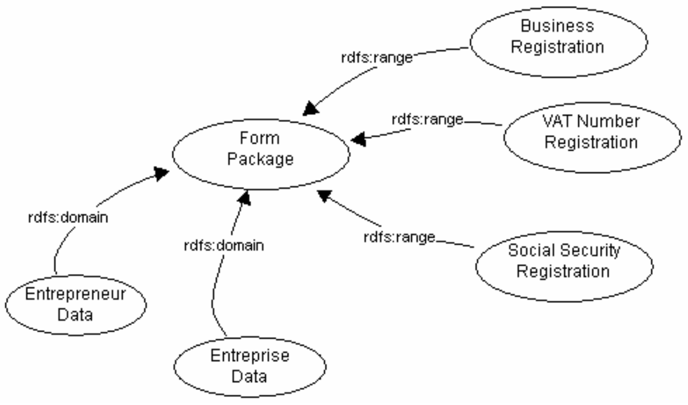

Figure 1: RDF Schema for Business Registration

However ontologies still cannot represent complex knowledge such as storytelling or human advice. As Samuel Johnson put it in the 18th century already, "Knowledge is of two kinds. We know a subject ourselves or we know where we can find information upon it". The goal of the component-based architecture we propose is to model "the information upon knowledge" and to describe this knowledge. We identified several attributes that allowed us to do so, beginning with the type of knowledge. [Capurro 2004] compares the knowledge typology proposed by [Zahn & al. 2000] with the classical Aristotelian one. Here we will only summarize the main points of Capurro's knowledge typology:

- **Know-how**: knowledge about how to make things (**technical** knowledge) and knowledge acquired through experience and remembrance (**empirical** knowledge).
- **Know-why**: logical reasoning (**scientific** knowledge).
- **Know-what**: knowledge about the best means to achieve given goals, usually a combination of know-how and know-why (**practical** knowledge).

[Capurro 2004] furthermore states that what can be managed is information or explicit knowledge and that implicit knowledge can only be “enabled”. In this context, **explicit** means that it can be clearly observed and expressed (and also digitalised), as opposed to **implicit** knowledge that can not be directly formulated (skills, experiences, insight, intuition, judgement, etc.) When knowledge is explicit, it can be represented as declarative or procedural knowledge. We are aware that in the domain of cognitive sciences, the distinction between procedural and declarative models is related to the brain memory system (see for example [Ullman 2001]), but here we used these terms here in a limited sense, as defined in computer science:

- **Declarative** knowledge components represent domain knowledge (facts, events, etc.) in terms of concepts and relations.
- **Procedural** knowledge components describe actions to be taken in order to solve a problem step by step.

For cases where knowledge is implicit and cannot be formalized, we introduced the concept of distribution: knowledge can be **individual** or **collective**, and in both cases components identify who has this knowledge or where it can be found. Finally we added a set of metadata (know-where, know-when, know-who, etc.) that describe these knowledge-components and that make it possible to manage them. Fig. 2 shows the complete component-based architecture under the form of a class diagram, but it can also be formalised in RDF.

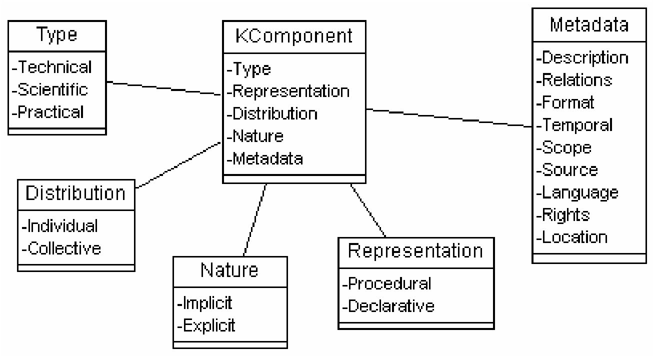

Figure 2: Knowledge Component

3 Process Models

[Mentzas & al. 2001] write that process modelling requires workflow models and techniques for capturing and describing a process. They present several techniques for workflow modelling, but we will concentrate on activity-based workflow modelling, where they define a workflow as a partial or total order of a set of tasks, the latter being partial or total orders of operation or descriptions for human actions. These workflow models provide know-how in procedural form. [Mentzas & al. 2001] also integrate the concepts of manipulated objects (documents, data records, hardware, etc.), roles (responsibility for a particular task) and agents (humans or information systems filling roles). These make it possible to represent know-how in declarative form.

We based our process models on their approach and, as mentioned in the introduction we used UML to develop them. Out of the nine types of diagrams supported by UML, we worked with use case, sequence, collaboration and activity diagrams because they are the only ones related to process modelling.

Use case diagrams are intended to capture the behaviour of a system by describing its interactions with its environment, particularly with the actors constituting it, human beings or machines. Use cases diagrams represent graphically an abstract set of actions and activities accomplished by such a system, what the creators of UML call its behaviour. These diagrams are often completed by textual scenarios. Figure 3 shows a rather **abstract view** on the process of creating a new business and of a set of administrative procedures an entrepreneur has to follow.

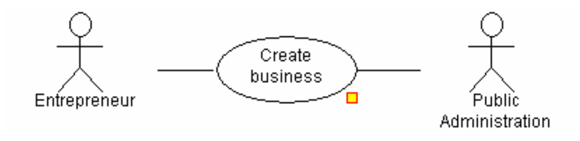

Figure 3: Use Case Diagram

Sequence diagrams and collaboration diagrams are called interaction diagrams in UML. Both types of diagrams are semantically equivalent, meaning it is possible to take one form and convert it to the other without any loss of information. They however present different points of view: sequence diagrams rely on a temporal scale and collaboration diagrams show the structural aspects of a system's interactions with a focus on the organisation of information flows. Figure 4 shows the **structural view** of the process of registering a new company.

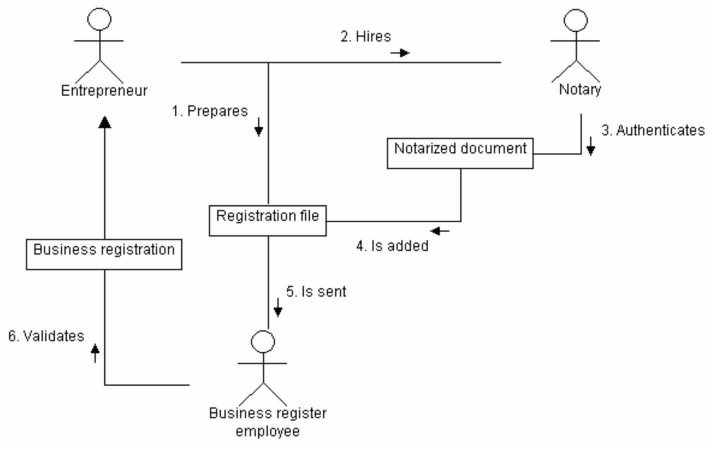

Figure 4: Collaboration Diagram

Sequence diagrams describe the interactions between objects contained in a system or a process; they can also be used to show the interactions of a system with its environment. Sequence diagrams are dynamic models of a system, showing mainly the circulation and the chronological order of exchanged messages. In our approach (Fig. 5) we used sequence diagrams as **a semantic link between the abstract and the descriptive levels**, for they are based on scenarios defined for each use case. However this does not provide a formal link between the two levels. To describe operational processes we used activity diagrams, which we will not show here.

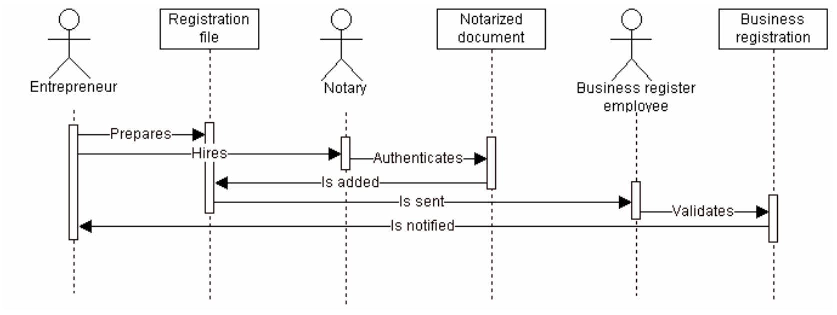

Figure 5: Sequence Diagram

4 Integration of Knowledge Components and Process Models

[Klischweski 2004] explains that there are two main strategies to achieve **interoperability** within an administration: **information integration** (providing access to structured informational resources across technical and organisational borders) and **process integration** (interrelating steps and stages of process performance across technical and organisational borders). He adds that in practice these two approaches seem to largely overlap but that there is no clear understanding on how they are perceived. We think that our architecture integrating knowledge components in process models constitutes a foundation for interoperable systems by providing a formal representation of informational resources and processes across technical and organisational borders.

While comparing specialised process modelling techniques (Adonis and OSSAD) and UML, we identified three major issues [Glassey & Chappelet 2002]. As UML is a description language that originated from the information systems world, some adaptation needs to be done to make UML applicable for process modelling. UML does not make any difference between a physical and an abstract actor (i.e. human being vs. machine or application); furthermore it does not formally distinguish actors and roles. Other than class models used for example to describe a database, UML does not provide any formalism for knowledge modelling, although authors such as [Schreiber & al. 2000] used it to support the CommonKADS knowledge management technique. Finally we found that there was no formal semantic link between the different modelling abstraction levels, such as provided by the ARIS metamodel [Scheer 2001]. Therefore in the following paragraphs we propose a new layer on top of the "classical" process modelling method described above.

We based our work on the model theory approach developed by [Wyssusek & al. 2001] to integrate process modelling and knowledge management. They provide an epistemological foundation to justify their work, but they do not offer any practical methodology or examples. That is why we created a conceptual framework that aimed at the integration of both these approaches.

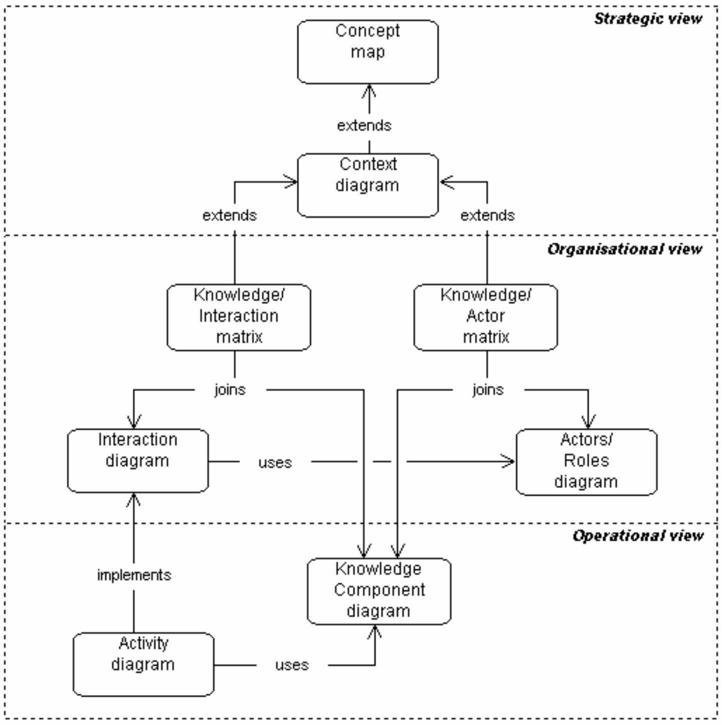

Figure 6: Metamodel

This framework consists of 8 types of diagrams, most of them being inspired or directly taken from existing modelling techniques. They cover the three different abstraction levels of an organisation that we defined in the introduction and integrate the “know-why” layer. As in UML or other modelling tools, it is not necessary to use all of them in order to provide a good representation of reality. Users should rather select the diagrams that suit their needs and goals in terms of modelling. Fig. 6 shows the **metamodel** of this framework and the formal relations between the different types of diagrams. We will provide examples for most of them.

Concept maps are the top-level diagrams and show the strategic goals of an organisation in terms of functions or processes (Fig. 7). Let us mention that the metamodel of our framework is in itself a concept map. These concept maps can be decomposed in several levels, a terminal node of this type of diagram is implemented by a context diagram.

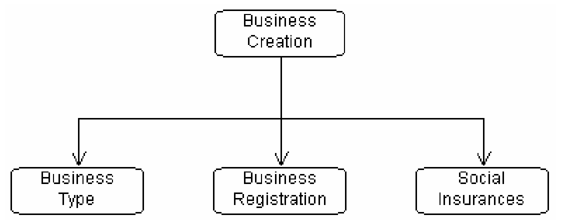

Figure 7: Concept Map

Context diagrams (Fig. 8) are almost exactly the same as use cases in UML, but we added the concept of knowledge packet. A knowledge packet is an abstract representation of a set of knowledge components such as described in section 2. These components encapsulate documents, databases, files, implicit knowledge and so on. They provide metadescriptions for “knowledge units”, thus allowing us to show what type of knowledge is necessary in order to complete a process and which knowledge is relevant in a given context. Context diagrams provide an abstract view of the “know-what”.

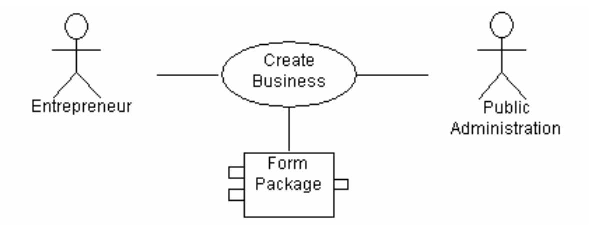

Figure 8: Context Diagram

As mentioned earlier we also missed the formal distinction between actors and roles in UML. Consequently we used **actor-role diagrams** in order to define the organisational structure of an administration. They can be either classical organisational charts or matrices that formally link actors and roles when the organisation is too complex to be shown graphically in an intelligible way. We will not show that here, but let use mention that the abstract actors represented in context diagrams can be linked to these actor-role diagrams. Moreover the actors described in such diagrams are used in the knowledge-actor matrices (see further on).

Fig. 9 shows a **knowledge-interaction matrix**, formally linking knowledge components to the interactions that implement a use case. In UML an interaction is the specification of how messages are sent between objects or other instances and interaction diagrams (sequence or collaboration diagrams) emphasize object interactions. We comply with this definition and use collaboration or sequence diagrams to specifically describe each interaction shown in this matrix. By matching the “know-why” (knowledge components) and the “know-how” (interaction diagrams), this matrix shows the “know-what” at the operational level.

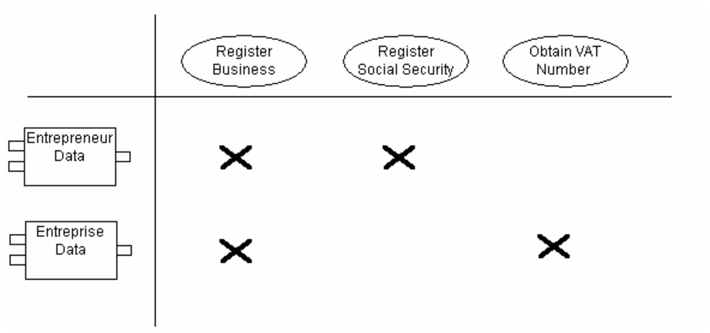

Figure 9: Knowledge-Interaction Matrix

Exactly as knowledge-interaction matrices link knowledge components and interactions, the concept of **knowledge-actor matrices** (Fig. 10) create a formal relation between knowledge components and real actors within an organisation. They provide an organisational view of the "know-what" or more precisely they show the "who-knows-what". That proves very useful in order to introduce implicit knowledge in a graphical model: it might not be possible to transform it into explicit knowledge but at least we know who has this knowledge within an organisation. Knowledge-interaction matrices can also link actors and interaction diagrams provided a small constraint: within interaction diagrams modellers should only use roles that were defined in actor-role diagrams.

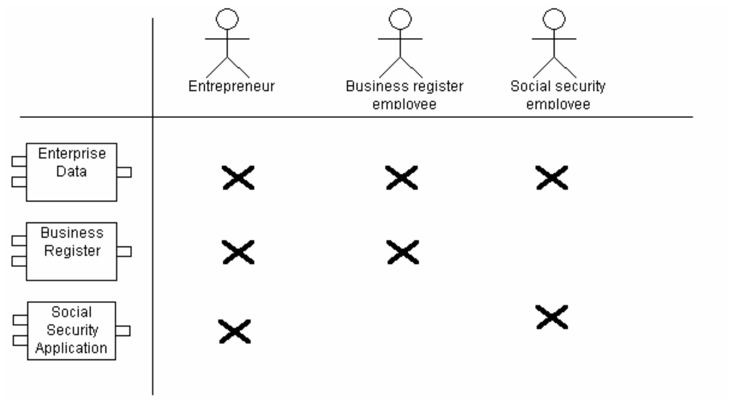

Figure 10: Knowledge-Actor Matrix

5 Conclusions and Future Work

On paper it might be difficult to see all connections between the different diagrams of our framework. However the metamodelling tool that we used in order to create these diagrams (UML Studio, from PragSoft Corporation) allows users to navigate between the different layers of the models and to follow the formal links by a simple click of the mouse, for example to retrieve the content of a knowledge component. It can also generate different formats of documentation. This makes it simple and efficient for acquiring, formalising and sharing domain knowledge, with both dynamic and static representations. We believe this provides the global view mentioned in the introduction and describes know-how, know-why and know-what within an organisation.

We are currently using this framework in a project with a German public agency and will draw on this experience to validate and refine it. In parallel we are pursuing our work on formal description for knowledge components, notably by applying OWL (Ontology Web Language), a semantic layer built on top of RDF and XML. We also need to further develop our process models: they are suitable to represent structured activity-based processes, but they lack flexibility and semantics in order to model decision-making or negotiation processes.

6 References

Booch, G., Rumbaugh, J. & Jacobson, I. (1999) *The Unified Modeling Language User Guide*. Boston: Addison-Wesley.

Capurro, R. (2004). Skeptical Knowledge Management. In Hobohm, H.-C., *Knowledge Management: Libraries and Librarians Taking Up The Challenge* (IFLA Publication, 108, 47-57). Munich: Saur.

Chappelet, J. L. & Snella, J. J. (2004) *Un langage pour l'organisation: l'approche OSSAD (3rd Edition)*. Lausanne: Presses Polytechniques et Universitaires Romandes.

Dumas, P. & Charbonnel, G. (1990) *La Méthode OSSAD, pour maîtriser les technologies de l'information*. Paris: Les Editions d'Organisation.

Fraser, J., Adams, N., Macintosh, A., McKay-Hubbard, A., Pariente Lobo, T., Fernandez Pardo, P., Cañadas Martinez, R. & Sobrado Vallecillo, J. (2003) Knowledge Management Applied to E-government Services: The Use of an Ontology, in Wimmer, M.A. (Ed.) *Knowledge Management in Electronic Government, Proceedings 4th International Working Conference KMGov 2003*, LNAI 2645, Heidelberg & al.: Springer Verlag.

Gamper, J., Nejdl, W. & Wolpers, M. (1999) Combining Ontologies and Terminologies in Information Systems, *5th International Congress on Terminology and Knowledge Engineering*, Innsbruck, Austria.

Glassey, O. & Chappelet, J.-L (2002) Comparaison de trois techniques de modélisation de processus: ADONIS, OSSAD et UML. Working paper, Lausanne: IDHEAP.

Glassey, O. (2004) Developing a One-Stop Government Data Model. *Government Information Quarterly, 21(2)*, 155-168.

Gruber, T. (1993) Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies, 43(5-6)*, 907-928.

Junginger, Kühn, H., Strobl, R. & Karagiannis, D. (2000) Ein Geschäftsprozessmaganement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendungen, *Wirtschaftsinformatik, 42(5).*

Klischewski, R. (2004) Information Integration or Process Integration? How to Achieve Interoperability in Administration. In Traunmüller, R., *Electronic Government: Third International Conference EGOV 2004,* Lecture Notes in Computer Science 3183, 57-65. Berlin: Springer Verlag.

Lenk, K. & Traunmueller, R. (1999) Perspektiven einer radikalen Neugestaltung der oeffentlichen Verwaltung mit Informationstechnik, in: *Oeffentliche Verwaltung und Informationstechnik*. Schriftenreihe Verwaltungsinformatik, Heidelberg: Decker's Verlag.

Mentzas, G., Halaris, C. & Kavadias, S. (2001) Modelling Business Processes with Workflow Systems: An Evaluation of Alternative Approaches, *International Journal of Information Management, 21,* 123-135.

Noy, F.N., Fergerson, R.W. & Musen, M.A. (2000) The Knowledge Model of Protégé-2000: combining interoperability and flexibility. Available at: http://protege.stanford.edu/papers.html, accessed January 5, 2004.

Ross, R. (1997) *The Business Rule Book : Classifying, Defining and Modeling Rules, Version 4.0.* Houston: Business Rules Solutions Inc.

Scheer, A.W. (2001) *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen*. Heidelberg & al.: Springer-Verlag.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W. & Wielinga, B. (2000) *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge: MIT Press.

Ullman, M. T. (2001) A Neurocognitive Perspective on Language: The Declarative/Procedural Model. *Nature Reviews - Neuroscience, 2*(10), 717-26.

Wyssusek, B., Schwartz, M., Kremberg, B., Baier, F. & Kralmann, H. (2001) Business Process Modelling as an Element of Knowledge Management - A Model Theory Approach. *Managing Knowledge 2001: Conversations and Critiques*. Leicester University, UK.

Zahn, E., Foschiani, S., & Tilebein, M. (2000) Nachhaltige Wettbewerbsvorteile durch Wissensmanagement. In *Wettbewerbsvorteile durch Wissensmanagement*, 239-270. Stuttgart: Methodik und Andwendungen des Knowledge Management.