Business Process Modeling for Organizational Knowledge Management

Luca Abeti[1], Paolo Ciancarini[2], and Rocco Moretti[2]

[1] IMT Institute for Advanced Studies, Piazza S.Ponziano, 55100 Lucca, Italy
luca.abeti@imtlucca.it
[2] University of Bologna, CS Dept., Mura Zamboni 7, 40126 Bologna, Italy
ciancarini@cs.unibo.it, moretti@cs.unibo.it

Abstract. The growing complexity of a networked and information-dependent economy requires the innovation of the adopted processes together with their related services. In particular, many Small and Medium-sized Enterprises (SME's) currently base their organizational models in a resource-centric view rather than in a knowledge-based organizational model which is a fundamental bound to their innovation capabilities. This paper presents a framework for organizational knowledge management. Our approach is based on Business Process Modeling (BPM), that is the main modeling practice connecting the management and engineering disciplines in software development. The aim is to present how the software requirements analysis can help in formalizing and sharing the knowledge concerning the business processes. Besides, we show how the service and ontology abstractions can be useful for software development.

1 Introduction

In recent years Ugo Montanari published as sole author or as coauthor a number of papers on some fundamental theoretical issues related to Service-Oriented Architectures, see for instance [8, 9, 10, 13, 22]. Service-Oriented Architectures are software architectures that enable new application scenarios in which small, loosely coupled pieces of functionality are published, consumed, and combined with other functions over a network. A key feature of the SOA's is a two-level model to implement global enterprise systems, where business functions are implemented by individual services and business processes are built as combinations of services.

This technologically-oriented research trend has to confront another, more socio-organizationally-oriented research trend. In fact, the studies about enterprise organizational processes have brought deep changes in the economy and society [11]. The evolution of Business Process Modeling (BPM) has been strongly influenced by its relationships with the new technologies like business process reengineering [16]. Despite the close relationship between the business process view in technology and economy, this concept is considered differently and for different goals by the software engineers and managers [26].

P. Degano et al. (Eds.): Montanari Festschrift, LNCS 5065, pp. 301–311, 2008.

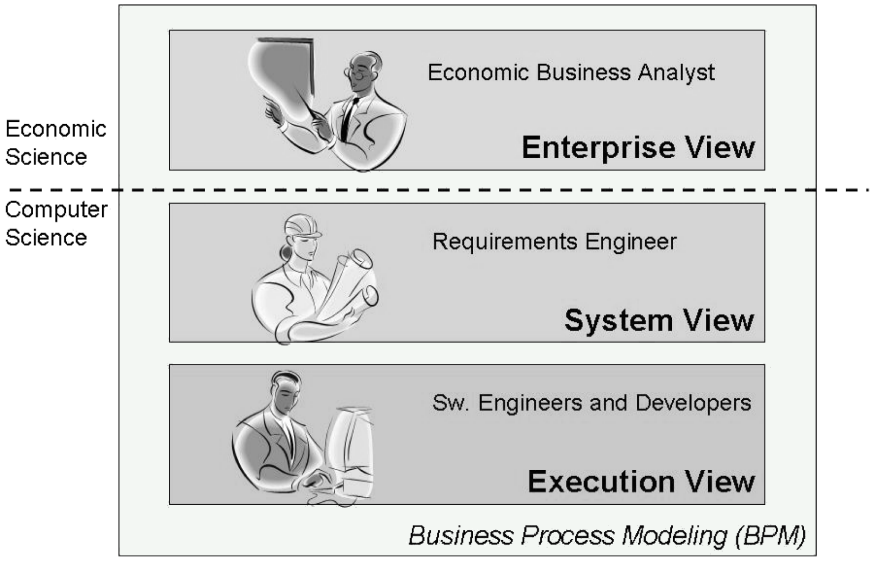

Fig. 1. The three views for BPM

The Business Processes (BP's) modelers of contemporary enterprises have to consider the changes enacted by the new technologies in their processes [29]. On the other hand, the new software engineering technologies influence BP's in the development of service systems [14]. Besides, the emergence of new enterprise models, such as networked and service-oriented enterprises, requires open and interoperable technologies supporting their processes.

Figure 1 identifies the three points of view representing the three main perspectives for BPM:

- **Enterprise view**: its goal is the business improvement. At this level the models deal with the organization, the strategies, the business rules, the business domains, the internal and external BP's, etc. It is the BPM perspective commonly used by the economic business analysts.
- **System view**: at this level the goal is to acquire the early-requirements of a system by means of a business analysis. The models concern the organization, the internal BP's, the business entities, the systems, the architectures, etc. It is the perspective of the software business analysts.
- **Execution view**: at this level the goal is to define an executable model for BP's. It is the software engineers perspective for the system design and implementation. This view of BP's is shared with the developers.

The capability to design a system by means of the service abstraction for its components has important implications in software engineering [22, 30] and can be easily connected to the BP models by means of a design based on the Model-Driven Architecture (MDA) [1]. This issue can be addressed in the *System*

view perspective and can be managed by software engineering languages and methods [6].

However, the BP-to-Service mapping can be treated differently in the *System* and *Execution views* because we can distinguish two further perspectives, namely: a *dynamic* and *static* representation of BP's and services. Our approach concerns only the static mapping of the knowledge related to the BP's into services and ontology abstractions. The dynamic behaviors and reconfigurations for services and architectures are considered orthogonal issues for the design of distributed systems by means of the BP abstraction [10, 17].

Currently do not exist methods and tools for BP's development that enable to connect all the three BPM views exposed in Figure 1. Indeed, the current Integrated Development Environments (IDE's) do not fully support the management of both the service abstraction and the related BP technologies evolution.

This paper honors Ugo focussing on the study of the relationship between software engineering and BP modeling. We propose a requirements-driven software development method that considers all the needs and motivations related to the BP's view in order to support the software systems development. Our approach uses the service as intermediate abstraction in order to incrementally model the system starting from its BP's representation. In our approach, the services are considered as autonomous computational entities that can be managed by means of model-driven tools in order to implement the systems.

In this method others modeling languages can be added in order to better support the implementation of BPM. For instance, constraints for service models can be added by means of the SENSORIA Reference Modeling Language (SRML) metamodel [7] in our model-driven tool.

In the next section, we show an overview of our approach. Section 3 presents some IDE and tools supporting the service abstraction management. In Section 4, we present our conclusions.

2 BP's for Organizational Knowledge Management

In this section, we present a method for product knowledge formalization based on software engineering and BPM techniques. This method presents three general phases which correspond to the three BPM views exposed in Section 1. These phases are: *Business Modeling*, *System Modeling* and *System Implementation*.

In order to define a method able to manage the entire business and to achieve the software knowledge formalization, we assume that the method itself can be tuned for a specific project. Indeed, the BPM process may vary for many reasons:

- BPM can be realized for merely knowledge acquisition or for analysis and requirements identification
- the system development can require a sketch, blueprint or detailed representation
- the development requires more emphasis on structural or behavioral modeling
- the stakeholders, analysts or managers can be not confident with the adopted modeling language

– the reasons of software development could be not well understood
– the methods and processes used inside the company can be inadequate and difficult to change

Such reasons are an important aspect of our work because high varying business and problems require to vary also the processes used for BPM [6].

This approach allows to address high-level-requirements in a distributed application and can be exploited in the development of a SOA.

2.1 Business Modeling

The goals of the Business Modeling phase is to wrap the current management with a technology support and to constantly share the knowledge among the economists and the system engineers inside a company.

The focus of the Business Modeling phase is on collecting and maintaining all the knowledge about the company organization, the strategies, the goals, the risks and the management-related issues. In order to face these aspects, this phase is decomposed into two sub-phases: the *informal brainstorming* and the *enterprise knowledge formalization*. The former sub-phase is a collective learning activity [23] trying to realize a shared knowledge-base about the company organization, the strategies and the goals.

Usually different stakeholders assign different meanings to the constructs (e.g., actors, goals, strategies, organizational units) of the organizational knowledge based on their mental models [2]. In this sub-phase, the models are mainly used to structure the problems and the organization. In the latter sub-phase the business process analysts and the managers try to formalize the knowledge in order to check its consistency and to discuss with the stakeholders. Thus, these sub-phases are cyclic Business Modeling sub-phases performed many times taking into account the specific company, project or stakeholders needs. These sub-phases enable to move from a tacit knowledge of the company into a codified and consistent knowledge consisting of BP's for the represented organization.

In the Business Modeling phase the degree of competency, the background and the belief of each participant may vary significantly. Thus, it does not make sense to propose a standard sets of steps and restrictive guidelines for Business Modeling. Besides, the language provided by Si* [21], UML [25] and the Business Process Management Notation (BPMN) [26] can be useful for enterprise knowledge formalization. This set of guidelines and languages is not restrictive because usually the stakeholders and managers are reluctant to spend time on brainstorming, process formalization, learning, training and becoming confident with formal specification languages and methods.

We now briefly discuss the three languages mentioned above. The UML Use Case diagrams have been chosen because of their proved efficacy in stakeholder-analyst collaboration [12]. The Use Case diagrams use intuitive and quick to understand concepts such as: *actor*, *use case*, *inclusion*, *extension*, *system*

boundary etc. BPMN is a specification of the Object Management Group (OMG) and the Business Process Management Initiative (BPMI.org) [26]. It synthesizes the best practices of the BPM community and defines a graphical notation for the BP's similar to a flow-chart. Such a notation is both consistent with UML and understandable by the stakeholders, analysts, business users, developers, etc. Moreover, BPMN is widely extensible and can be incrementally adopted. Si* considers a set of primitive concepts such as *actor*, *goal*, *task*, and *resource* in order to model a socio-technical system. An *actor* is an active entity having strategic goals and performing actions to achieve these goals. A *goal* is a strategic interest of an actor. A *Soft-goal* is similar to a goal, but the fulfillment condition is not clearly defined. A *task* specifies a sequence of actions that can be executed to achieve a goal. Finally, a *resource* represents a physical or an informational entity.

Goals, tasks, and resources are often related among them in many ways. In particular, three relations have been identified, namely *AND/OR decomposition*, *means-end*, and *contribution* relations. The AND/OR decomposition combines AND and OR refinements of a root goal into sub-goals. The Means-end relation identifies tasks providing means for achieving a goal and the resources produced or consumed by a task. The contribution relation identifies the impact of the achievement of goals and tasks on the achievement of other goals and tasks. Due to the general meaning of the Si* concepts, we use them without a prescriptive formalism in the informal brainstorming sub-phase. Thus, starting from the beginning of our practice, the tacit knowledge is acquired in terms of the concepts that we will use in the enterprise knowledge formalization sub-phase and in the System Modeling phase.

We distinguish between functional and non-functional specifications. The functional specifications are formalized using the UML Use Case diagrams as regards the static aspects of BP's, and BPMN as regards the dynamic interactions among the BP's concepts. The non-functional specifications closer to tacit knowledge are formalized by using Si* diagrams. The informal brainstorming sub-phase allows to identify an unorganized and not-formalized set of early-requirements that giving a first sketch of the domain and the knowledge concerning the organization. The enterprise knowledge formalization sub-phase is the first phase trying to obtain a formal and analyzable BP representation. The cyclic informal brainstorming sub-phase helps the stakeholders in understanding their implicit knowledge and defining a shared knowledge base of the processes. The enterprise knowledge formalization sub-phase tries to organize the informal knowledge in order to analyze it in the System Modeling phase.

The modeling of goals and strategies (that are not caught by Use Cases and BPMN models) are essential for BPM design and knowledge acquisition. For this purpose, our approach uses the Si* notation. Si* helps to model strategical and operational aspects of the business. By means of the Si* concepts, we are able to connect formally represented systems and actors to, for instance, business units, manager aims, practices and company policies.

2.2 System Modeling

At the beginning of the System Modeling phase, we use the outputs of the Business Modeling phase in order to derive an analysis similar to the Tropos early-requirements analysis [19]. The Business Modeling phase provides three inputs to the System Modeling phase:

1. A formalized knowledge about the business and business processes represented in a set of diagrams.
2. Some indicative choices or purposes to develop new systems and products.
3. A first analysis concerning the technologies that can be used to realize new goals or improve the existing business.

In the System Modeling phase, further technical choices are made in technical brainstormings. In particular, we consider the non-functional requirements defined by means of Si*. Besides, the goals and the soft-goals and their relationships with the system can be deeply analyzed by exploiting the Tropos early-requirements analysis process [15, 19]. The early-requirements analysis concerns with the understanding of a problem by studying an existing organizational setting. The intentions of the stakeholders are modeled as goals and goal dependencies among actors, and analyzed by means some form of goal analysis. The output of this phase is an organizational model including the relevant actors and their respective dependencies for the achievement of the goals and the soft-goals, and for performing or obtaining resources. The System Modeling phase includes six models that can be defined by means of Si*:

- **Actor Model**: allows to identify the actors and their objectives, entitlements, and capabilities. The agents are also described in terms of the roles they play.
- **Social Model**: allows to identify and analyze the social relationships among the system actors and the stakeholders. Trust and distrust relationships between actors are discovered and modeled (i.e., expectations of actors about the capabilities and behaviors of other actors).
- **Goal Model**: allows to model the goals from the point of view of an actor. The impact of the goals on the achievement of other goals is analyzed and modeled in order to refine the requirements models and to elicit new social relations among the actors.
- **Execution Dependency Model**: allows to identify the actors depending on other actors for achieving their goals, executing the tasks and supplying the resources. In this model the assignments of responsibilities among the actors are discovered and modeled.
- **Delegation Model**: allows to model an actor delegating to other actors the achievement of goals, the execution of tasks and the access to the resources. This model enables the transfer of rights among the actors.
- **Task/Resource Model**: allows to elicit and model the tasks and the resources providing means for the achievement of goals. The impact of the tasks on the achievement of goals is analyzed and modeled.

These models enable to define a rigorous representation of the enterprise knowledge by means of the Actor, Social, Execution Dependency and Delegation Models. Besides, they enable to perform some analysis on such a knowledge by means of the Goal model and Task/Resource Model. The analyses that can be carried out are:

- **Means-end analysis**: aimed to identify tasks, goals or resources that provide means for achieving a specific goal.
- **Goal/Resource refinement**: analyzes and decomposes the goals and/or resources in terms of AND/OR decompositions.
- **Contribution Analysis**: studies the impact of the tasks and the achievements of goals on the achievement of other goals.

Starting from the Business Modeling phase, if in the System Modeling phase the need emerges to reengineer BP's or new IT system, the Si* diagrams must be considered together with UML Use Cases and BPMN diagrams. In this way, the IT System requirements can be derived and used in one or more System Implementation phases.

One of the most important activities of the System Modeling phase is the mapping from the business process concepts into the concepts useful in the System Implementation phase. We do not suggest to map BP's directly into the programming languages concepts (e.g., objects and classes), but we try to exploit the ontology and service concepts as intermediate abstractions enabling to move from business processes to implementation paradigms. Starting from the Si* and BPMN models, one or more application service models, depending on the number of the systems to implement, are defined.

The concept of service is considered as a very general abstraction for software development and can be used to represent a wide range of interacting software components [30]. In the same way, the UML Use Cases are used to derive a set of ontological concepts to be represented in one or more ontology models. In order to perform the Use Case-to-Ontology mapping, we can use the Rational Unified Process Business Model-to-System mapping rules [20] to incrementally redefine the Use Case models in a system-centric perspective. The application service and ontology models are both represented in diagrams similar to the UML Class Diagrams [25]. We propose this type of diagrams since it is very intuitive and has an easy to remember semantics. Besides, the service abstraction is useful for a logical division of the software [28]. Such a division is more coarse-grained than the division obtained by components or objects. The selection of the service and ontology abstractions represents a meeting point between the designer and developer requirements in the translation from BP's to services. On the one hand, to derive services from BP's it is useful to realize high level interfaces that are representative of the actual provided business services. On the other hand, the granularity of the services helps to define software components that can be easily changed and reused. The definition of the appropriate level of granularity for the services should consider both cohesion (i.e., the degree of relatedness of service functions) and coupling (i.e., the degree of service independence). The cohesion and coupling information can be derived in the BP's-to-Service mapping. The

non-functional requirements can be derived from the Si* models by means of goals and soft-goals dependences analysis.

2.3 System Implementation

The last phase of our practice is the System Implementation phase whose goal is to model the concrete executable support for the BP's. The aim of this phase may change depending on the specific structure of the Business Modeling and System Modeling phases. The System Implementation phase is not necessarily performed because in the System Modeling phase it is possible to decide that no changes have to be made in the current BP's and that no new systems have to be realized.

The outputs of the System Modeling phase are the service and the ontology models. Such models are used to realize a detailed design of the system. Depending on the system nature (e.g., Web applications, centralized systems, Grid systems, agent-based systems, etc.), a specific system implementation abstraction is used in order to model the system (e.g., classes, agents, Web Services, Web pages, CORBA components, Entity Java Beans, etc.). In this context, the service model represents the behavioral aspects of the software, while the ontology model represents the concepts used in the static aspects. For instance, the services in the application service model can be used in the UML behavioral models (e.g., the activity diagrams) or in the execution business process languages (e.g., BPEL4WS [4]). In an analogous way, the ontology models can be used for the UML structural models (e.g., the class diagrams) or to define conceptual and logical models of databases.

3 Tools for the System Modeling Phase

In this section we briefly describe some tools useful in the System Modeling phase described in Section 2.2. In order to support the mapping of the BP's represented by means of Si*, UML Use Cases and BPMN into the service abstraction, we need tools allowing to define, model, and deploy in a simple and platform-independent way the software services. In particular, we focus on Uniframe [5] and the MOdeling TOol for Grid and Agent Services (MOTO-GAS) [1].

Uniframe [5] aims to overcome the platform heterogeneity of distributed systems by using MDA-based service-oriented models. Uniframe defines an abstraction for a unified architecture, relies on MDA in order to design the service models, and uses a formal specification language to define the components and to support their connection. Uniframe investigates many component-based and service-oriented issues and defines an abstraction for a unified architecture. It uses UML to design service models and MDA to realize the Unified Meta-Component Model (UMM) used as a glue putting together different technologies.

MOTO-GAS [1] is a tool that enables to model both stateful and stateless services. A meaningful aspect of MOTO-GAS is the support for the Web Service Resource Framework (WSRF) [27]. Such support is given through the WSRF MetaObject Facility (MOF) [24] metamodel. MOTO-GAS allows to define a platform-independent application service model which can be mapped into an

instance of the WSRF MOF metamodel. In this context, the application service model defined in the System Modeling phase represents an application defined by a set of services and resources in a Platform-Independent Model (PIM) described by UML. By means of the MDA-based ATLAS Transformation Language (ATL) [3, 18], it is possible to produce a Platform-Specific Model (PSM) complying with the WSRF MOF metamodel.

MOTO-GAS uses some existing plug-ins implementing the MDA specifications and allowing an effective consistency between the realized models and their respective metamodels. This set of plug-ins allows to define the application service models and to automatically generate: a WSDL definition, the Java service skeleton, the Web Service Deployment Descriptor (WSDD) file, and the Java Naming and Directory Interface (JNDI) deployment file for the service application used in the System Implementation phase.

Both Uniframe and MOTO-GAS are useful to support the System Modeling phase down to the System Implementation phase of our proposed practice. In particular, a model-driven support is considered an essential feature in order to develop platform-independent models of the system derived from the BP models designed in the Business Modeling phase.

4 Conclusions

In this paper we have presented a modeling approach considering a whole business and its organizational issues. It is aimed to obtain a full interaction between technologists and managers involved in the business innovation performed by means of the new technologies.

Our proposed method aims to analyze the use of BPM in software engineering. We have combined three complementary aspects represented by: Si*, UML Use Cases, and BPMN. Even though these aspects overlap, they enable to manage BP's from different points of view.

Considering a software engineering point of view, the service and ontology concepts represent strategic elements enabling to connect the economic and execution view of BP's. In particular, the service abstraction represents the bottleneck for BP's because it allows to map the enterprise BP's into the executive BP's and the system functionalities. Such BP's are mapped into services that are connected in order to create execution BP's and system functionalities.

A framework including both the support for the collaborative work among stakeholders, and the support for model-driven development of the system and ontology abstractions is far from being realized. In this context a further support of the Business Modeling phase in a model-driven IDE may consist in automating the mapping of the Si*, UML Use Case and BPMN models into the service and ontology models.

Acknowledgements

This work is partially supported by the MIUR FIRB project TOCAI.IT URL: http://www.dis.uniroma1.it/∼tocai/

References

[1] Abeti, L., Ciancarini, P., Moretti, R.: Service oriented software engineering for modeling agents and services in Grid systems. Multiagent and Grid Systems Journal 2(2), 135–148 (2006)

[2] Adamides, E., Karacapilidis, N.: A knowledge centred framework for collaborative business process modeling. Business Process Management Journal 12(5), 557–575 (2006)

[3] Allilaire, F., Idrissi, T.: ADT: Eclipse development tools for ATL. In: Akehurst, D. (ed.) 2nd European Workshop on Model Driven Architecture (MDA) with an emphasis on Methodologies and Transformations (EWMDA-2), Canterbury, UK, pp. 171–178. The Computing Laboratory (2004)

[4] BEA Systems International Business Machines Corporation and Microsoft Corporation. Business Process Execution Language for Web Services (BPEL4WS) specifications (May 2003)

[5] Benatallah, B., Dijkman, R., Dumas, M., Maamar, Z.: Service Composition: Concepts, Techniques, Tools, and Trends. In: Stojanovic, Z., Dahanayake, A. (eds.) Service-Oriented Software System Engineering: Challenges and Practices, ch. 3, pp. 68–87. Idea Group Publishing, Hershey, PA (2005)

[6] Berztiss, A.T., Bubenko, J.A.: A software process model for business reengineering. In: Proceedings of Information Systems Development for Decentralized Organizations (ISDO 1995), an IFIP 8.1 Working Conference, Norwell, MA, USA, August 1995, pp. 184–200. Chapman & Hall - Kluwer Academic Publishers (1995)

[7] Bruni, R., Lafuente, A., Montanari, U., Tuosto, E.: Service oriented architectural design. In: Barthe, G., Fournet, C. (eds.) TGC 2007. LNCS, vol. 4912, pp. 186–203. Springer, Heidelberg (2008)

[8] Bruni, R., Lafuente, A., Montanari, U., Tuosto, E.: Style based reconfigurations of Software Architectures. Technical Report TR-07-17, Dipartimento di Informatica, University of Pisa (2007)

[9] Bruni, R., Melgratti, H., Montanari, U.: Theoretical foundations for compensations in flow composition languages. In: POPL 2005: Proc. 32nd ACM Symp. on Principles of Programming Languages, vol. 40, pp. 209–220. ACM Press, New York (2005)

[10] Buscemi, M., Montanari, U.: CC-Pi: a constraint-based language for specifying Service Level Agreements. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 18–32. Springer, Heidelberg (2007)

[11] Davenport, T.: Process Innovation: Reengineering work through information technology. Harvard Business School Press, Boston (1993)

[12] Dobing, B., Parsons, J.: How UML is used. Communications of the ACM 49(5), 109–113 (2006)

[13] Ferrari, G.L., Hirsch, D., Lanese, I., Montanari, U., Tuosto, E.: Synchronised Hyperedge Replacement as a Model for Service Oriented Computing. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 22–43. Springer, Heidelberg (2006)

[14] Ghezzi, C.: Software Engineering: Emerging Goals and Lasting Problems. In: Baresi, L., Heckel, R. (eds.) FASE 2006. LNCS, vol. 3922, Springer, Vienna, Austria (2006)

[15] Giorgini, P., Kolp, M., Mylopoulos, J., Pistore, M.: The Tropos methodology: an overview. In: Gleizes, M.P., Bergenti, F., Zambonelli, F. (eds.) Methodologies And Software Engineering For Agent Systems, ch. 5, pp. 89–105. Kluwer Academic Publishing, Norwell, MA, USA (2004)

[16] Hammer, M.: Reengineering Work: Don't Automate, Obliterate. Harvard Business Review 68(4), 104–112 (1990)
[17] Hirsch, D., Montanari, U.: Consistent transformations for software architecture styles of distributed systems. In: Stefanescu, G. (ed.) Workshop on Distributed Systems. ENTCS, vol. 28, p. 4 (1999)
[18] Jouault, F., Kurtev, I.: Transforming models with ATL. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
[19] Kolp, M., Giorgini, P., Mylopoulos, J.: Organizational patterns for early requirements analysis. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 617–632. Springer, Heidelberg (2003)
[20] Kruchten, P.: The Rational Unified Process: An Introduction, 3rd edn. The Addison-Wesley Object Technology Series. Addison-Wesley Longman Publishing, Boston (2003)
[21] Massacci, F., Mylopoulos, J., Zannone, N.: An ontology for secure socio-technical systems. In: IGI Global (ed.) Handbook of Ontologies for Business Interaction. Information Science Reference, Hershey, PA, USA, vol. 1, p. 469 (December 2007)
[22] Montanari, U.: Web Services and Models of Computation. Electronic Notes in Theoretical Computer Science 105, 5–9 (2004) (invited talk)
[23] Morecroft, J.: Mental models and learning in system dynamics practice. In: Michael (ed.) Systems Modelling: Theory and Practice, ch. 7, pp. 101–126. John Wiley, Hoboken (2004)
[24] OMG. Meta-Object Facility (MOF), v. 1.4 (March 2002), `http://www.omg.org/technology/documents/formal/mof.htm`
[25] OMG. Unified Modeling Language (UML) specification v. 2 (2004), `http://www.omg.org/technology/documents/modeling_spec_catalog.htm`
[26] OMG. Business Process Modeling Notation (BPMN) specification v. 1.0 (2006), `http://www.bpmn.org/Documents`
[27] Organization for the Advancement of Structured Information Standards (OASIS). Web Services Resource, 2005. Working Draft (2005), `http://docs.oasisopen.org/wsrf/2005/03/wsrfWSResource1.2draft03.pdf`
[28] Papazoglou, P., Yang, J.: Design methodology for web services and business processes. In: Buchmann, A., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. (eds.) TES 2002. LNCS, vol. 2444, pp. 175–233. Springer, Heidelberg (2002)
[29] Paulson, L.: Services Science: A New Field for Today's Economy. IEEE Computer Magazine 39(8), 18–21 (2006)
[30] Tsai, W.T.: Service-oriented system engineering: A new paradigm. In: Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop, Washington, DC, USA, October 2005, vol. 0, pp. 3–8. IEEE Computer Society, Los Alamitos (2005)