Towards a Framework for Comparing Process Modelling Languages

Eva Söderström[1], Birger Andersson[2], Paul Johannesson[2], Erik Perjons[2], and Benkt Wangler[1]

[1] Department of Computer Science, University of Skövde
Box 408, 541 28 Skövde, Sweden
{eva.soderstrom,benkt.wangler}@ida.his.se
[2] Department of Computer and Systems Sciences
Stockholm University/Royal Institute of Technology
Electrum 230, 164 40 Kista, Sweden
{ba,pajo,perjons}@dsv.su.se

Abstract. The increasing interest in process engineering and application integration has resulted in the appearance of various new process modelling languages. Understanding and comparing such languages has therefore become a major problem in information systems research and development. We suggest a framework to solve this problem involving several instruments: a general process meta-model with a table, an analysis of the event concept, and a classification of concepts according to the interrogative pronouns: what, how, why, who, when, and where. This framework can be used for several purposes, such as translating between languages or verifying that relevant organisational aspects have been captured. To validate the framework, three different process modelling languages have been compared: Business Modelling Language (BML), Event-driven Process Chains (EPC) and UML State Diagrams.

1 Introduction

Business Process Modelling has become a major focus of attention in Information Systems Engineering, in order to create efficiency, quality and customer satisfaction. Process models can be used for planning, design, simulation and automatic execution of business processes, e.g. in Workflow Management Systems and Process Brokers [1, 2]. Furthermore, methods like Total Quality Management and Business Process Reengineering, and software packages like SAP R/3 and Baan ERP all have put the business processes in the centre of analysis. As a result, several different process modelling languages have been developed, e.g. Business Modelling Language, Event-driven Process Chains, and UML Activity and State Diagrams. However, these languages often define and use concepts in different and sometimes ambiguous ways, which makes comparisons between and integration of process models difficult.

A. Banks Pidduck et al. (Eds.): CAISE 2002, LNCS 2348, pp. 600-611, 2002.

Language comparison can be made easier by using e.g. a meta-model. The reason is that the meta-model provides a graphical illustration of the basic concepts and their relations that is easy to grasp even for non-experts on process modelling languages. The purpose of this paper is to suggest a framework that aims at making comparisons between business process modelling languages easier to perform. The framework consists of several instruments: a general process meta-model, an analysis of the event concept, and a classification of concepts according to the interrogative pronouns: what, how, why, who, when, and where. The intended users of the framework are IS/IT-managers, business people and other stakeholders involved in business process management in different domains. The framework must therefore: firstly, be easy to understand for people not familiar with process modelling; secondly, include basic business concepts that are central to business process management (e.g. activity, event, actor, location, resource and time); and thirdly, be extensible to enable users to complement it with concepts of interest to a certain business domain.

The paper is structured as follows: Related research in Chapter 2 presents approaches such as meta-modelling and ontology analysis for comparing and evaluating process modelling languages. Chapter 3 presents a process meta-model, an analysis of the event concept and a classification of the concepts according to the interrogative pronouns. Chapter 4 introduces three process modelling languages: EPC, UML State diagram and BML. In chapter 5, the meta-model is used to compare the three such languages, and the result is presented in a in a table (referred to as a "comparison matrix"). Chapter 6 contains conclusions and directions for further research.

2 Related Research

Meta-models, ontologies and conceptual models are often used to describe and analyse the relations between concepts. A model is an abstraction of phenomena in the real world, and a meta-model is yet another abstraction highlighting properties of the model itself [3]. *Meta-modelling* is closely related to, and to some extent overlapping, ontology analysis and conceptual modelling.

Ontology is a philosophical discipline where the nature of the real world is studied. Some ontologies attempt to define basic concepts in a certain domain, e.g. medicine or automobile manufacturing, while others try to be more domain independent. The Bunge, Wand and Weber (BWW) ontology [4, 5] is domain independent. Wand and Weber [5, 6] have used the BWW ontology to provide a theoretical foundation for the evaluation of information systems models. They assume that an information system (IS) represents a real world system, and that it is built to manage information process functions in this real world. They present a set of basic real world concepts that IS models should be able to express. By mapping the concepts (e.g. thing, state, event and system) to concepts in different languages, Wand and Weber discuss strengths and weaknesses in these languages. However, some concepts in this ontology are difficult for non-experts to map to concepts in everyday process modelling languages.

Examples of *models of process concepts* are the Workflow Management Coalition WfMC reference model [7] and the FRISCO report [8]. In the WfMC reference model, terminology, structure, components and interfaces for workflow management

systems and languages are defined. The FRISCO report also defines central concepts in the IS area (e.g. process, state and action), but its definitions of are different from those given by WfMC. None of these models show how concepts from different types of process modelling languages, e.g. activity-oriented, state-oriented and communication-oriented languages, relate to one another. Activity-oriented languages primarily describe which activities follow and precede another in a process. Examples of such languages are UML Activity Diagram [9], Task Structures [10], and Event-driven Process Chain (EPC) [11, 12]. State-oriented languages, for instance UML State Diagram [9], describe which states follow and precede another in a process. SDL [13] and Business Modeling Language (BML) [2, 14] are examples of communication-oriented languages that focus on the interaction between people and systems, and between systems.

The basic grammar of most process modelling languages derives from Petri nets [15], which provide both a graphical description and formal definition of processes. Researchers [16, 10] have mapped EPC and Task Structures to Petri nets to give the languages formal semantics, and hence to verify the correctness (soundness) of the process definitions. This approach could be used to compare different process modelling languages, but Petri net analysis are for method experts and cannot easily work as a platform for communication between business people. Furthermore, a process modelling language for business process management must include more concepts than just places, transitions and tokens that are present in classical Petri nets.

We have grouped the meta-model concepts according to a set of interrogative pronouns to clarify what aspects of processes that different concepts represent. Other researchers that use interrogative pronouns to structure their analysis are Zachman [17] who uses them to classify a descriptive representation of an enterprise with focus on managing and developing an enterprise's system and Bunge [18] who uses the pronouns to understand and separate basic types of domain-dependent problems.

3 The Framework

The framework will be explained starting with the basic concepts, before the meta-model is introduced. Finally, we explain how and why the interrogative pronouns have been used in the meta-model.

3.1 Basic Concepts

Most process modelling languages include at least four basic concepts: time point, activity, state and event, i.e. these four concepts are the common denominators of process modelling languages. Intuitively, a *time point* is an instant in time, not further decomposable. An *activity* is a performance of some sort, possibly changing some thing's *state,* i.e. its set of properties. An *event* is a noteworthy occurrence. Usually, one is interested in particular events associated with changes of state, i.e. activities are involved in some way. Activities, states and the running of time can be thought of as existing regardless of an observer but events are some facts about a thing that an observer notice and records by some means.

Most process modelling languages agree on the general meaning of these concepts and their definitions are usually precise but differ in some respects. The greatest differences between the languages lie in the understanding of the relations between the concepts. We claim that it is useful to consider an event as a connector that connects states and activities in time as is schematically illustrated by the lines from event to the other three concepts in Fig.1. One reason why languages differ is that activities, states and time can be connected in many ways and this suggests that the relations between the concepts are best understood by analysing the event concept.

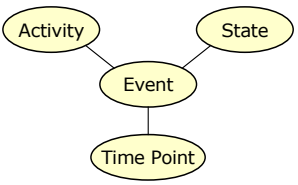

Fig. 1. The intuitive relation between the event concept and other basic concepts

The following event types were identified in an analysis of the event concept from the languages BML, UML State Diagram, and EPC:

- Events can either record a certain point in time (*time point events*) or record the time between two time points (*time duration events*).
- Events can either record the start of an activity (*pre-activity events*) or record the end of an activity (*post-activity events*).
- Events may record the change of a state (*state change events*) or not.

Using the event types described above in a language analysis, some conclusions can be drawn about the languages and their use of the event concept. Firstly, the languages define an event as non-similar combinations of the above mentioned event types. For example, one language defines an event as a combination of post-activity and state change events, i.e. the completion of an activity always leads to a change of the process' state. Other languages define event as a combination of time point and pre-activity events, and so on. Secondly, some languages do not recognise differences between event types as they are presented above, e.g. by not distinguishing between pre-activity and post-activity events.

3.2 The Meta-model

Figure 2 shows the meta-model, which is divided into two levels: type and instance. On the type level, we find activity, resource, role and logical dependencies between activities. On the instance level, we find event, state, actor, and temporal dependencies between events. Thus, the meta-model includes some additional concepts besides the basic ones to make it more useful in a business setting.

A process is modelled on a type level as a structure of logical dependencies between activities. These activities use one or more resources as input, and produces one or more resources as output. One specific type of resource, the role, is regarded to be responsible for that one or more activities will be performed.

The execution of a process is regarded to be a time-bound series of events, caused by an actor. These events may result in a state change for a resource. An actor who causes an event always has an intention with his/her actions to achieve a specific state for a resource. The state can be either different from the resource's current state, or it can be the same state as the current one.

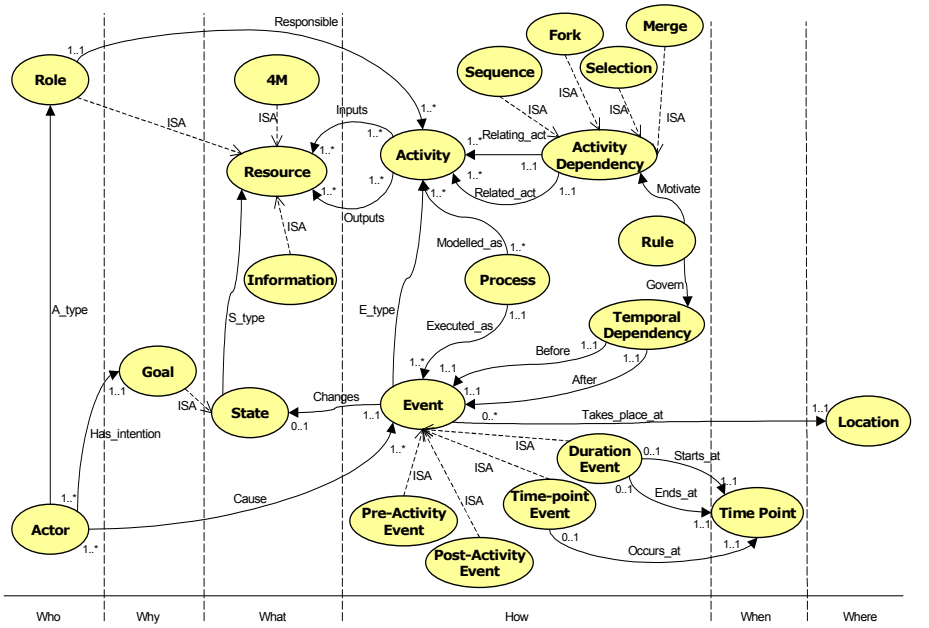

Fig. 2. The meta-model

An actor is regarded to be an instantiation of a role, a state is viewed as an instance of a resource, and an event is an instance of an activity. In the latter case, our perception of an activity is through the event that initiates and/or terminates it. An event always occurs at a certain time point or between two time points on a certain location.

3.3 Interrogative Pronouns

The interrogative pronouns: where, who, how, when, what and why, have been inserted into the meta-model to enhance readability and understandability. They provide an easy-to-grasp classification of the concepts according to various aspects, and also give an intuitive meaning to the abstract concepts.

This classification can help organisations to ask questions about their processes and process models, and to aid organisations in selecting the process modelling language appropriate to their business. For example, an organisation for which actors are central can exclude languages that do not model actors, i.e. languages that disregard from the "who" aspect.

4 Process Modelling Languages

EPC, UML State Diagram and BML represent different types of process modelling languages: an activity-oriented, a state-oriented, and a communication-oriented (see section 2). The three languages are considered to be representative of their respective category, which motivates their presence in this paper. Following are a short presentation of the languages, and a description of how the event concept is used in respective language. This section includes as an example a very short snippet of a manufacturing process to show language symbols and how they are used.

4.1 Event-Driven Process Chains (EPC)

Event-driven Process Chains (EPC) [11] is used for instance to describe business processes in the SAP/R3 enterprise system. The EPC diagrams are also embedded and used in the process view of the Architecture of Integrated Information System (ARIS) framework that integrates five different perspectives or views (data, function, organisation, output and process) of an organisation [12].

EPC is a graph with active nodes, called functions (soft rectangles), and passive nodes, called events (hexagons), see Fig. 3 (left). In EPC, a process is considered to be a chain of business functions to be executed, and of events describing the situation before and after each function. EPC does not explicitly use the state concept. The logical relationships and dependencies between functions and events are described using logical connectors (circles including the logical AND, OR and XOR) and control flow (arrows). The EPC diagram of Fig. 3 shows that two events, *Order received* and *Production date arrived* must occur before the business function *Manufacturing* can take place. When the function is completed, two additional events must occur: *Product in store* and *Order executed*. The function nodes can be connected to information, material, product and services and responsible organisational unit, by adding the other views of the ARIS framework. This is not included in Fig. 3.

EPC explicitly uses the following event types:

- Pre-activity events (i.e. the events before the functions in the EPC diagram).
- Post-activity events (i.e. the events after the functions in the EPC diagram).

EPC does not use the following event types:

- State change events (because EPC does not explicit use the concept state)

EPC does not explicitly distinguish between following event types:

- Time point events and Time duration events

Note that EPC does not explicitly combine any event types.

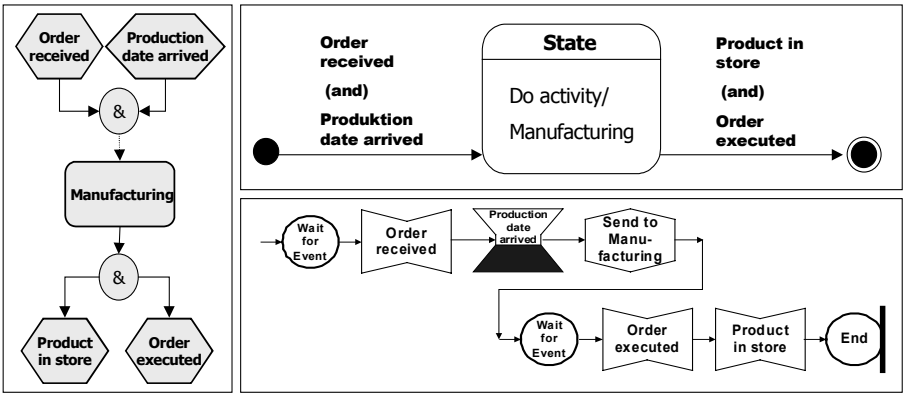

Fig. 3. Three different modelling languages: EPC (left), UML State Diagram (top, right) and BML (down, right)

4.2 UML State Diagram (SD)

The Unified Modelling Language (UML) [19] is an industry standard maintained by the Object Management Group (OMG) [9]. The state – or state-chart – diagram (SD) is one of several diagram types in UML. SD is a graphical representation of a state machine and visualises how and under what circumstances a modelled element, e.g. a UML class, a system or a business process, changes its state. SD is also used to show which activities that are executed as a result of events. The SD graph uses symbols for states (soft rectangles), transitions (arrows), events (text labels), and activities (text labels), and always includes a start state (solid circle) and one or more final states ("bulls-eye"), see Fig. 3 (right, top).

A state consists three components: name, variables (describing local variables like counters and timers), and activities; where all but the name component is optional. The state symbol is divided into three compartments to illustrate this division. The difference between an action and an activity is that an action is non-divisible and non-interruptible, and that an activity can contain several actions. There are three different types of actions/activities associated with the state of a modelled element: *entry action* (performed when entering the state); *exit action* (performed when exiting the state), and *do activity* (performed while in the state). Fig. 3 only shows the *do activity* Manufacturing. A state that does not contain any activities is called a wait state.

State changes are called transitions (arrows connecting the states). A transition occurs if an event occurs and aborts an ongoing do activity, or if a do activity has been completed and triggered a completion event. However, before a transition can occur, an optional condition called "guard" must be evaluated. If the guard evaluates to true, the transition occurs, otherwise it will not. During a transition certain actions can be performed. A SD considers events to occur instantaneously, while states have duration, but UML does not hinder modellers to redefine events to have duration and states to have no duration. Some additional event types from UML state diagrams are omitted in this paper for the sake of brevity.

SD explicitly uses the following simple or combined event types:

- Post-activity events (i.e. an event occurs, which aborts an ongoing do activity, but does not trigger a transition since a guard is evaluated to false).
- Post-activity events and State change events (i.e. an event occurs, which both aborts an ongoing do activity and triggers a transition).
- Post-activity events, and Pre-activity events (i.e. an event occurs, which both aborts an ongoing do activity and initiates the exit action, but does not trigger a transition because a guard is evaluated false).
- Post-activity events, Pre-activity events and State change events (i.e. an event occurs, which both aborts an ongoing do activity, initiates an optional exit action, triggers a transition, triggers some optional actions while in transition, and initiates optional entry action and do activities in the next state).
- State change events (i.e. an event occurs, and the state is a wait state, and the event triggers a transition).
- Pre-activity events (i.e. an event occurs, and the state is a wait state, and the event initiates an exit action, but does not trigger a transition because a guard is evaluated to false).
- Pre-activity events, and State change events (i.e. an event occurs, and the state is a wait state, and the event both initiates an optional exit action, triggers a transition, trigger some optional actions while in transition, and initiates optional entry action and do activities in the next state).

SD does not explicitly distinguish between the following event types:

- Time point events and Time duration events

4.3 Business Modelling Language (BML)

The Business Modelling Language (BML) is used in a Process Broker [2], the Visuera Process Manager [14]. BML focuses on describing interactions between systems through the sending and receiving of messages. The language has similarities to SDL (Specification and Description Language) [13], but is more adapted to application integration. One important feature of BML is that it can be used for business specification and design as well as for the execution of systems.

BML describes structure and behaviour of a system by using two kinds of diagrams. The Business Process Integration (BPI) diagram shows the system structure in terms of its logical decomposition into parts and the communication taking place between the system and its environment, i.e. external applications and human agents. The communication metaphor is sending and receiving messages through a channel. The Business Integration Application (BIA) diagrams describe the dynamic behaviour of the system, visualising when the system sends and receives messages, the rules for handling messages and what activities to perform depending on the messages' contents.

The BML symbols in the process diagram (BIA), some of which are presented in Fig. 2 (right, down), are: Receive message (concave box), Send message (convex box), and State (circle). Further BML symbols are: Start timer (hourglass "full of time"), Expire timer (hourglass "out of time"), Business activity (rectangle, not

shown), and Automated business decision (rhombus, not shown), which shows what paths should be taken through the process according to business rules. BIA always includes a start state (circle without name) and one or more End states (circle with the label "End"). A BIA can visualise to which application, human agent or process that a message is sent to or received from, by using labels (or icons) above the diagram's send and receive symbols (not shown in Fig. 2).

An instance of a BIA can either be in a state or in transition from one state to another. A transition is initiated when a message is received or when a timer is expired. Activities that may occur during the transition, e.g. Send message and Start timer, are considered to happen instantaneously.

BML explicitly uses the following combined event types:

- Time point events, Pre-activity events and State change events (i.e. a Receive Message or Expire Timer triggers one or several activities, i.e. Send message, Start timer and/or, Business activity, and a transition.)
- Time point events and State change events, (i.e. a Receive Message or BML Expire Timer triggers a BML transition, but not an activity.

BML does not explicitly use the following event types:

- Post-activity events
- Time duration events

5 Language Comparison

Results from comparing process modelling language using the meta-model is presented in Table 1. The table provides a structured overview of defined concepts, and clearly indicates some differences between the languages. For example, empty boxes in the table illustrate that a particular language does not explicity express this concept. Anyone creating and using the table for language selection is made aware of this fact and can eliminate a language that cannot express desired concepts or extend it with the missing concept. Furthermore, the table makes differences in definitions between the languages explicit, which enables organisations to discover languages which defines and uses concepts in ways similar to the organisation itself. Creating the table may also provide an opportunity for organisations to reflect about the way they define their own concepts and consider alternative definitions.

6 Summary and Further Research

The framework presented in this paper consists of a meta-model of process concepts with a comparison matrix, a classification of the concepts according to interrogative pronouns, and an analysis of the event concept. It aims at making comparisons between various process modelling languages easier. In this summary, we will highlight contributions and strengths of both the parts, and of the framework as a whole. Since the parts provide slightly different views of the same phenomenon, some of their contributions will be the same.

Table 1. Comparison of EPC, BML and UML SD

Concept	EPC	BML	SD
Time Point			
Event	Uses two simple events types: 1) Pre-activity events 2) Post-activity events	Uses two combinations of event types: 1) Time point+ Pre-activity+State change events. 2) Time point+State change events. Both correspond to the BML concepts "Receive Message" and "Expire Timer"	Uses seven simple or combined event types, see section 4.2.
State		Corresponds to the BML concept "Wait for Event"	Corresponds to the SD concepts "State" and "Wait State"
Activity	Corresponds to EPC concept "Function"	Corresponds to the BML concepts "Business activity", "Send message" and "Start timer"	Corresponds to the SD concepts "entry action", "do activity" and "exit action"
Process	Corresponds to a partially ordered set of logically dependent "functions"	Corresponds to a partially ordered set of logically dependent "business activities"	Corresponds to a partially ordered set of logically dependent "actions/activities"
Rules	Corresponds to a set of pre- and post conditions ("events") that must be true before and after an "function"	Corresponds the BML concept "Automated business decisions"	Corresponds to a set of conditions on the "transitions"
Resource	Corresponds to the EPC concept "Resource".	Corresponds to the BML concept "BIA "	Corresponds to the SD concept "Class "
Actor		Corresponds to the BML concepts "BIA", "application", and human agent, capable of sending "messages" at a time point.	Corresponds to the SD concept "Object" capable of producing an "event".
Location			

Starting with the *meta-model*, one contribution is that it explores basic definitions of concepts used in various process modelling languages. Furthermore, the meta-

model is extensible and can thus be adapted to different business domains. It also enables organisations to cover many different organisational aspects through the use of the *interrogative pronouns*. These pronouns provide understandability, readability, and an easy-to-grasp classification of concepts. The meta-model still needs some work, however, such as a deeper exploration into all the meta-model concepts, and real life cases through which the extensibility of the meta-model can be shown.

By using a *comparison matrix* (or table) to present the results from applying the meta-model, several contributions can be identified. The matrix provides a structured overview of defined concepts, and an explicit illustration of definitional differences between the languages. This enables organisations to identify the language that best suits the organisation's own definitions and intentions. Should an organisation lack explicit definitions of its concepts, the matrix can provide a basis for creating such definitions.

By analysing the *event* concept, we have highlighted that some major differences between the languages lie in how they use this concept. However, the other basic concepts (activity and state) also need to be analysed to enhance the usefulness of the framework, as do the concepts in the complete meta-model (rule, resource, actor, and goal).

The *framework as a whole* was validated by comparing three different process modelling languages: BML, EPC and UML SD. The comparison shows that the framework can be used for several purposes, many of them already described in this chapter. Different process modelling languages can be compared with respect to their concepts definitions, concept relationships, and correspondences of concepts between languages. The result provides a basis for selecting the process modelling language that best suites the organisational needs. The framework can also be used as a translation instrument between process modelling languages, and to identify what organisational aspects that the languages cover. As mentioned, the framework still needs some work: a method for using the framework, including step-by-step guidelines for how to perform e.g. language comparisons, needs to be developed. Furthermore, more formal definitions of the basic concepts are needed, as are experiences from real life case studies where the framework is applied, validated and verified.

References

1. Linthicum, D.: Enterprise Application Intergration, Addison-Wesley (2000).
2. Johannesson, P., Perjons, E.: Design Principles for process modelling in enterprise application integration, Information Systems, 26:165-184 (2001)
3. van Gigch, J. P (1991) System Design Modeling and Metamodeling. Plenum Press, New York. ISBN 0-306-43740-6
4. Bunge, M.: Treatise on Basic Philosophy Vol 3, Ontology I: The Furniture of the World, Reidel, Dordrecht, Boston (1977)
5. Wand, Y.: Ontology as a Foudation for Meta-modelling and method engineering, In: Information and Software Technology 38 (1996), 281-287
6. Wand, Y., Weber, R.: An Ontological Model of an Information System, In: IEEE Transactions on Software Engineering, 11 (1990), p 1282-1290

7. Reference Model - The Workflow Reference Model, WFMC-TC-1003, 19-Jan-95 (1995), 1.1, and Terminology & Glossary, WFMC-TC-1011, Feb-1999, 3.0 (1999). Available at: http://www.aiim.org/wfmc/mainframe.htm
8. The FRISCO Report, A Framework of Information System Concept, IFIP (1998), available at: http://www.liacs.nl/~verrynst/frisco.html
9. OMG Unified Modelling Language Specification, Version 1.3. (1999), available at: http://www.oml.org
10. van der Aalst, W. M. P, Ter Hofstede, A. H. M.: Verification of Workflow Task Structures: A Petri-net-based Approach, Information Systems, vol. 25, no. 1 (2000)
11. Keller, G., Nüttgens, M. , Scheer, A.W.: Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89, University of Saarland, Saarbrücken (1992)
12. Sheer, A.: ARIS-Business Process Modelling. Springer-Verlag, Berlin (1998)
13. Belina, F., Hogrefe, D., Amardeo, S.: SDL with Applications from Protocol Specification. Carl Hanser Verlag and Prentice Hall International, UK (1991)
14. Wåhlander, C., Nilsson, M., Törnebohm, J.: Visuera PM Introduction, Copyright Visuera AB (2001)
15. Reisig, W.: Petri Nets: an introduction. Springer-Verlag, Berlin (1985)
16. van der Aalst, W. M. P: Formalization and Verification of Event-driven Process Chains, In: Information and Software Technology, 41(10):639-650, (1999)
17. Zachman, J.: Enterprise Architecture: The Issue of the Century In: Zifa Framwork Articles (1996), available at: http://www.zifa.com
18. Bunge, M.: Scientific Research I, Springer-Verlag, (1967)
19. Rumbaugh, J., Jacobson, I., Booch, G.: The unified modeling language reference manual, Addison Wesley Longman Inc. (1999)