Business Modelling. Languages and Tools

Audris Kalnins

University of Latvia, IMCS, 29 Raina blvd, LV-1459, Riga, Latvia
audris@mii.lu.lv

Abstract. The paper provides an overview of business modelling techniques - both at language and tool level. The basic elements of some most popular business modelling languages are briefly outlined, including the GRAPES-BM language developed by IMCS, University of Latvia. The paper presents also the basic principles, how metamodel based generic modelling techniques can be used for supporting several modelling notations simultaneously.

1 What is Business Modelling

There exist two types of mathematical modelling:

- the standard (continuous) one - what ECMI is typically about and which is based on various sorts of mathematical equations
- the discrete modelling based on diagramming techniques with mathematical logics and set theory as deep foundations.

Discrete models are sets of related diagrams with various constraints on their structure, representing some relevant aspects of the system to be modelled. The degree of model formality may be different - from semiformal to completely formal.

The discrete modelling itself may be of different types, the most popular ones are:

- *object modelling* (modelling of static structure of systems)
- *business process modelling*
- *data modelling*
- *concurrent system modelling* by means of sets of finite state machines.

This paper deals mainly with business process modelling (or Business Modelling (*BM*) for short) - a set of methods for semiformal description of business processes.

Business modelling appeared as a discipline in early nineties and suddenly got a limelight after the appearance of the well-known book [1] by M. Hammer and J.Champy, the "inventors" of BPR (Business Process Reengineering) in 1993. Though the book has nothing to do with modelling in the exact sense, it stimulated the interest of corporate managers to look for BM methodologies and tools as a support for this "silver bullet" - BPR. Namely then the idea of building a pair of business models - those of "as-is" and "to-be" appeared, with the hope of finding the optimal "to-be" model in the end. But BPR is not the sole goal of BM, there are more:

- to understand the behaviour of large business systems (e.g., by managers)
- to specify a business system as part of the requirements specification for a software system supporting it.

But with the BPR glory fading away in late nineties, a new buzzword-2002 has appeared - Business Process Management (BPM), which basically uses the same methodologies and tools, though combined with up-to-date tools for building web-based IT systems relying on the results of process modelling.

So methodologies and tools for business modelling have become both a serious research area and a booming software market.

The introduction concludes with a form of a definition for business modelling.

Business modelling is a set of rules and methods for semiformal diagrammatic definition of processes within a business system, their structural breakdown up to elementary activities, the flow of control between these parts, the necessary resources (performers) for the activities and the flow of data between parts of the system.

2 Business Process Modelling Languages

Formally a business modelling method is defined by a *business modelling language* - the set of syntactic and semantic rules for building a set of diagrams and related objects constituting a valid model of a business system.

The world of BM languages is characterised by the fact that there is no one leading or dominant BM language, no officially accepted standards exist in the area. This is in sharp contrast e.g. to object modelling where UML [2] strongly dominates. At the same time the basic principles of leading BM languages are quite similar. Even more, a great deal of concepts (in other words, metamodel elements) are common to most BM languages, though named differently in each language:

- *activities* or tasks - the elementary steps of a process, typically represented by a kind of boxes in process diagrams
- transitions (flows) from activity to activity, represented by lines
- facilities to describe alternatives in a process (branching)
- facilities to describe activity concurrency (fork, join)
- resources required to perform the activities (performers), shown either in the same diagram or as references to a separate organizational diagram
- possibilities to refine an activity by a subprocess (a process hierarchy).

The semantics of BM languages typically is based on some well-known concept of discrete mathematics, such as finite state machines, Petri nets [3], some other precisely defined abstract machine, or even a very small fragment of the set theory. But here two quite confronting requirements appear:

- the notation must quite concisely express the underlying semantics, with the possibility to apply also some formal analysis
- it must be easy readable by non-professionals in a semiformal way, without much time spent on learning the syntax and semantics.

Namely because of these requirements the main graphical element of BM languages - *business process diagrams* are so similar and remind simple flowcharts, though the underlying formal semantics differ. The main challenge in BM language design is to balance the formality and informality in a reasonable way.

For a BM language to become usable in practice, it must be supported by a set of *tools*. The first component of these tools is a graphical editor, enabling an easy

drawing and editing of process and other types of diagrams present in the language. But there must be also analysis components, from simple reporting tools presenting object definition-reference relations and other simple queries, to advanced *simulation* facilities. In business process modelling world mainly the discrete event simulation, based on a process model, but extended with more numeric data, is used. Typical results include average costs of processes, performer workloads etc. The nature of BM languages fits well into the discrete event simulation framework - a discrete advance of time when an event occurs. These simulators are the main tools in BPR area for comparing several "to-be" models and finding the optimal one. An intermediate position is taken by static evaluation tools, the most popular being finding the critical (longest) path in a process graph and performing Activity Based Costing (ABC) analysis for finding approximate costs of a process execution. The simulation can deliver much more precise results than static methods but is more complicated and requires higher accuracy of the numeric input data. On the other hand, static evaluation is properly applicable only to processes with simple logic. There are more than thirty BM languages known, with one or more tools supporting each. Here we consider only some of the most popular ones, with a good tool support and being a commercial success according to the Gartner report [4]. The remaining part of this chapter is devoted to short overview of these languages, based on a simple process example.

2.1 Classical BM Languages

Possibly the oldest (the origin about 1991) and officially the most used BM language is the BM view of ARIS [5], supported by the equally named tool (being the market leader, according to [4]). Actually ARIS is a union of methodologies, where BM with its eEPC (extended Event-Process-Chain) diagram and other related diagrams is only a part. Fig. 1 shows a small fragment of an order processing as an eEPC diagram, where rounded rectangles represent activities, hexagons - events triggering (and generated by) the activities. The circles with logical symbols (AND, XOR etc.) represent flow junctions - triggering conditions, branches, forks. This type of diagram may also show performers of activities (a performer symbol linked to an activity), the data used and a lot of other things. In general, the ARIS language contains an enormous quantity of various graphical symbols, but its semantics is not very precisely defined. ARIS tool supports also process simulation.

Fig. 1. Order example as ARIS eEPC diagram

Another very popular BM language, more exactly, a family of languages is the IDEF languages, the most proper BM language of them being IDEF3 [6]. Originated by US Air Forces, the IDEF project was then taken over by KBSI, and IDEF3 BM language has in a way become an unofficial US standard for business modelling (after publication of its first draft in 1992 there was an attempt to standardize it). IDEF3 is supported by many BM tools, the most popular of them being System Architect, Bpwin, Procap/Prosim, Design/IDEF. An activity in a process here is called UOB (Unit Of Behaviour) and again is represented by a box, but a line (called precedence link) between two activities asserts that the second activity does not start before the first completes. The IDEF3 standard defines several link types differing by constraints they impose on e.g. the existence of successor activity, but in practice (in tools supporting the language) they are considered to be simple control flows. Some tools add also object flows to the notation. The language contains junction symbols (rectangles containing logical symbols), where *disjunctive* fan-out junction represents branching, a conjunctive one - fork and so on. The standard provides no performer specification, but tools borrow notation from other methodologies, e.g. via swimlanes. Fig. 2 shows the previous example as an IDEF3 diagram (in System Architect).

Fig. 2. Order example as IDEF3 diagram

There are also some other popular BM notations based on so called *process charts* (introduced by G. Rummler and A. Brache) - a variation of a comparatively old flowchart-like notation, but having the important concepts of business event and result. Some BM methodologies are based on the traditional software design flowchart notation - surprisingly many BM tools still use such a notation.

2.2 UML Activity diagrams for Business Modelling

UML has been originally defined as a universal Object Oriented modelling language and as a OO software design language [2]. It has become a de-facto standard in these areas, and is supported by a great number of tools, the most popular being the Rational Rose [7]. But the UML Activity diagram, originally meant for flowcharting of complicated algorithms, can be used as a BM language too. Since the BM step has now been recognised as an important software requirements specification step, the pressure increases to use Activity diagram as a standard BM notation. On the one hand, Activity diagrams are well suited for BM, activities (represented by boxes) have their natural meaning, control flows define pure consecutive execution of activities, but object flows (with an object symbol - a box in the middle) represent data

transferred from activity to activity. The junction symbols represent either concurrent fragments (between fork and join symbols - both heavy black lines) within a sequential process or flow branching and merging (both shown by hollow diamonds). Branching conditions are specified as texts in brackets along the branch lines. Performers can be specified via swimlanes. Formally the syntax and semantics is concisely defined - as an extension of finite state machines where the only events are activity completions. But this somewhat simplistic definition leads to a great mess for control and object flows - it cannot be simply decided when object flows imply control flows [8]. Possibly because of that many tools improperly implement object flows (e.g., example in Fig.3. cannot be drawn by Rose). The situation is expected to change radically by advent of UML 2.0 in 2003, where Activity diagrams will be significantly modified (the semantics will be based on Petri nets).

Fig. 3. Order example as UML Activity diagram

Another UML diagram having relevance to BM is Use case diagram, which could serve as a high level overview of business processes in a system, but the current UML version also is insufficient for this goal.

3 Contribution of IMCS LU in Business Modelling - GRADE

More than a decade the Computer Science department of IMCS, University of Latvia has been active in the research and development in modelling area. In 1992, upon some initial input from Siemens-Nixdorf, the first version of GRAPES-BM business modelling language was designed. The final version of the language [9] was defined in 1996, and it has withstood the test of time, when compared to some BM languages which appeared later. Initially under a contract with Siemens-Nixdorf, the support tool for the language - GRADE has been built in close cooperation with Infologistik GmbH, DATI and later on Exigen. The development of GRADE tool still continues, the current version is 4.0.17. GRADE is used widely both in Latvia (several banks and insurance companies, DATI, Exigen Latvia) and worldwide (Germany, Italy, UK, US, Netherlands).

The basic element of process notation in GRAPES-BM is *task* (activity), which is represented by a rounded rectangle, links between the tasks represent either control

flows (unnamed arrows) or message flows (named arrows). Flow branching is represented via decision symbols (hexagons) containing the branch conditions, but a concurrent fork is shown simply by several outgoing flows from the same task symbol or decision. The flow merging and joining is shown as a triggering condition (OR or AND respectively) within the subsequent task, more general triggering conditions are also supported. A process start point is denoted by a timer symbol. The performers of a task are specified by means of a performer expression, referencing elements in ORG diagrams, which are used to describe the organizational structure of a business system. It should be noted that the reference (actually, hyperlink) principle is widely used in GRAPES-BM, e.g. a message flows references the message definition (which is contained in the message table), but this definition in turn can reference its data type, which is defined in a Data type diagram. All GRAPES-BM elements can have additional numeric attributes used for simulation.

A concise semantics definition of GRAPES-BM language [10] is based on a custom abstract machine which actually is a sort of Petri net extension with explicit queues introduced. Thus an unlimited behaviour concurrency constrained only by the available resources is obtained, which is typical to most real business processes. The semantics is so natural that in most cases a GRAPES-BM diagram can be unambiguously read without thinking of the formal definition, like any flowchart-like diagram. But the semantics definition contains also the *transaction* concept for defining the behaviour of a group of concurrent subactivities of an activity - a natural dynamic equivalent of static fork-join pairs used e.g. in current UML activity diagram semantics.

Fig. 4. Order example as GRAPES-BM process diagram

The first objective for the GRADE tool development has been to provide a maximum support for the user when building or modifying the model diagrams. Mainly this has been achieved by means of elaborated automatic graph layout algorithms [11], as far as possible providing automatically the most expected positioning of diagram elements. The degree of automaticity can be varied from

completely automatic to completely manual. For example, a swimlane-based layout of process diagrams (as in UML Activity diagrams) can be obtained automatically from the standard layout (as in Fig.4). Another step in this direction is the maximum support for hyperlink-like navigability between diagrams and their elements, an elaborate prompting system for selecting always only a relevant reference, as well as advanced dictionary service for tracking e.g. all references to a definition of an object. One more aspect important for building readable models is the use of stereotypes (similar to UML) for defining several default styles (corresponding to some semantic variations) for the same diagram element.

GRADE supports business model simulation via integrated simulator. Most of valuable statistics, such as process times and costs, performer workloads etc can be obtained automatically. The transaction concept is important also for simulation, permitting e.g. to obtain automatically the breakdown of process costs into costs for its subprocesses.

In addition to business modelling, GRADE supports also object modelling by means of UML class diagrams, with a special orientation towards conceptual modelling.

4 Generic Approach to Business Modelling

As it was outlined in the previous sections, most of the different business modelling languages have their own value. Some of the concepts are nearly isomorphic in all languages, e.g. activity/function/UOB/task. Some tools which try to support several BM notations simultaneously (ARIS, System Architect) use a simple copy/paste support between the relevant diagrams for some of the isomorphic elements.

A more systematic approach to the problem is under development at IMCS LU. The approach is based on the modelling language *metamodel*. The metamodel concept has become popular in recent years especially due to the principle used in UML definition[2]. To put it short, a metamodel is a class diagram containing all modelling concepts, their attributes and their relationships. Some of the BM tools (ARIS, System Architect, STP) already use the metamodel approach to a certain degree, to accommodate easier a very large number of diagram types and their elements.

An alternative is a completely metamodel based *generic modelling tool* (previously called metaCASE). Such a tool has no built-in modelling methodology. It has to be filled up with a specific metamodel and additional information to start modelling something. The approach is not completely new, currently the most notable similar project is GME [12], which is oriented mostly towards building of specialized modelling environments for industry. The research in this direction was started in IMCS within the EU ESPRIT project ADDE [13]. The first results by IMCS and the basic principles are outlined in [14,15]. Now IMCS together with Exigen is building a commercial BM tool based on these principles.

It is obvious that to supply a metamodel for a modelling language or method is far to short for specifying a modelling tool within a generic environment. The additional information is environment specific, for the IMCS approach it is named Editor Definition Language (EdDL) [15]. To apply EdDL, the metamodel must be in a

specific form. EdDL itself consists of several parts (only some will be explained here):

- a fixed standard add-on to the metamodel (the core)
- rules how the metamodel has to be built, namely how it has to be separated into domain and presentation layers
- a special sublanguage for definition of the mapping from an element of the presentation layer to the corresponding element of the domain layer (and vice versa)
- the definition facilities for user stereotypes - the default styles of the diagram elements
- facilities to define the general diagram properties (layout style etc)
- facilities to define the attribute editors (dialogs) for all model element types
- facilities to define functionality of the model browser (model tree)

Certainly, for EdDL to become live, the generic environment itself - actually an EdDL interpreter is required.

We start the explanation with the principles of metamodel splitting. The *presentation* layer contains classes literally corresponding to all graphical elements in the diagram type - box and line types. For example, GRAPES-BM process diagram (simplified) would contain boxes - Task symbol, Decision symbol, Timer symbol and lines - Flow line and Decision line (visible in Fig. 4 as a line segment attached to a decision). The presentation layer will have more to do with the "geometric" properties of the diagram. If only one modelling notation were to be defined, there would be no need for another layer. But since we have several modelling notations to be supported, the *domain* layer is necessary. It contains the true modelling concepts, possibly common to several modelling notations. For example, it could contain the *Activity* element, which can be represented as a *Task* (GRAPES-BM), *Activity* (UML Activity), *Function* (ARIS), *UOB* (IDEF3). Namely the domain layer elements must contain "semantic" attributes. Thus *Activity* should have attributes *name*, *performer expression*, *duration* (and possibly something else). The life is much harder for elements which appear differently in different modelling notations - e.g. conditions appear in a box in the GRAPES-BM notation and along a line in UML Activity notation. Nevertheless, it occurred that for UML Activity and GRAPES-BM Process diagram it is possible to build a relatively natural common domain layer ("domain metamodel"), though GRAPES-BM notation had to be adapted slightly. The question whether there exists a common domain metamodel for at least the BM notations mentioned in this paper is still open, though concepts in all these notations informally are quite similar. The finding of such common domain metamodel would be interesting per se - it would express "the true essence" of business modelling.

The next element to be explained is the *mapping* between presentation layer and domain layer. In the simplest case it is an association between a class in presentation layer (e.g., *Task symbol*) and the relevant class in the domain layer (e.g., *Activity*). The meaning of mapping is to define what domain element has to be built when a new graphical symbol has been inserted in a diagram. And conversely, what has to be built in the diagram, when a new relevant domain class instance has appeared (apparently - via another modelling notation). To become a mapping definition, this pure association must be augmented by constraints - in the case of EdDL in a language similar to UML OCL [2]. In addition, it has to be specified, how domain class

attributes (and associations) correspond to text fields (compartments) in the diagram symbol. A mapping defined by a single association is just the simplest case, there are several mapping types in EdDL, corresponding to the used diagramming patterns. Especially nontrivial are the mappings for lines, since their end points also have to be specified. But the mapping principle as such has occurred to be very adequate for defining the simultaneous modelling according to several notations, for example the mappings for GRAPES-BM Process and UML Activity diagrams occurred to be quite manageable (certainly they depend on the selected domain metamodel). Fig. 5 shows a simple example of mapping, with EdDL constraints inserted into the metamodel fragment.

Fig. 5. Mapping example

With metamodel, mappings, user stereotypes and all the other required EdDL elements defined, the Generic modelling environment becomes a true modelling tool supporting several modelling notations. And when a diagram in one notation is updated, the corresponding diagram in another notation is automatically updated too. Certainly, the modelling environment itself should provide all the basic tool services.

Fig. 6. The same process as both GRAPES-BM and UML Activity diagrams

Fig. 6 shows how the experimental implementation of this principle in Exigen Business Modeller presents the same data both as a GRAPES-BM process and UML Activity diagram. The implementation has confirmed the practical feasibility of the approach. It has shown that even for business models of large real systems it is possible to work with alternative representations.

References

1. Hammer, M., Champi, J.: Reengineering the Corporation: A Manifesto for Business Revolution. Harper-Collins, New York (1993)
2. Rumbaugh, J., Booch, G., Jacobson, I.: The Unified Modeling Language Reference Manual. Addison-Wesley, Reading Harlow Menlo Park Berkeley Sydney (1999)
3. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. PrenticeHall, Englewood Cliffs (1981)
4. Gartner Inc.: The BPA Market Catches Another Major Updraft.
<http://www.gartner.com/gc/webletter/idsscheer/issue1/article1.html> (2002)
5. Scheer, A.-W.: ARIS Business Process Modeling. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (2000)
6. Mayer, R., Menzel, C., Painter, M.: IDEF3 Process Description Capture Method Report http://www.idef.com/Downloads/pdf/Idef3_fn.pdf, Knowledge Based Systems Inc (1995)
7. Quatrani, T.: Visual Modeling with Rational Rose 2002 and UML. 3rd edn. Addison Wesley Professional, Boston (2002)
8. Bock, C.: Unified Behavior Models. Journal Of Object-Oriented Programming Vol. 12, No 5 (1999) 65-70
9. Kalnins, A., Barzdins, J. et al: Business Modeling Language GRAPES-BM and Related CASE Tools. Proceedings of Baltic DB&IS'96, Vol. 2, Institute of Cybernetics, Tallinn (1996) 3-16
10. GRADE Business Modeling Language Reference. Infologistik GmbH,
<http://www.gradetools.com/> (1998)
11. Kikusts, P., Rucevskis, P.: Layout Algorithms of Graph-like Diagrams for GRADE-Windows Graphical Editors. Lecture Notes in Computer science, Vol. 1027. Springer-Verlag (1996) 361-364
12. Ledeczki, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason IV C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The Generic Modeling Environment, Workshop on Intelligent Signal Processing, Budapest (2001)
13. ESPRIT project ADDE. <http://www.fast.de/ADDE>
14. Kalnins, A., Podnieks, K., Zarins, A., Celms, E., Barzdins, J.: Editor Definition Language and its Implementation. Proceedings of the 4th International Conference "Perspectives of System Informatics", Lecture Notes in Computer Science, Vol. 2244. Springer-Verlag, Berlin Heidelberg New York (2001) 530 – 537
15. Kalnins, A., Barzdins, J., Celms, E., Lace, L., Opmanis, M., Podnieks, K., Zarins, A.: The First Step Towards Generic Modelling Tool. Proceedings of Baltic DB&IS 2002, Vol. 2, Tallinn (2002) 167-180