A Business Process Design Language

Henk Eertink, Wil Janssen, Paul Oude Luttighuis,
Wouter Teeuw, Chris Vissers

Telematics Institute, P.O. Box 589, 7500 AN Enschede, The Netherlands
{eertink,janssen,luttighu,teeuw,vissers}@telin.nl

Abstract. Business process modelling and analysis puts specific requirements on models used and the language for expressing those models. The models should be easily understandable and analysable. In this paper we study the requirements for such a language and introduce a language that satisfies those requirements to a large extent. It was developed in the Testbed project, which aims at developing a systematic approach to business process change.
The language, called AMBER, has a graphical representation, and allows to model processes, data, and the organisation and people involved in a uniform and integrated way. On the basis of a formal foundation of the language, different analyses and tool support are available. We illustrate our approach with a realistic example.

1 Introduction

Organisations are complex artefacts. They involve different customer groups, business units, people, resources and systems. They stretch over numerous different processes that interact in a seemingly chaotic way. When trying to change business processes within organisations one is confronted with that complexity. In order to cope with that complexity and improve the grip on changing business processes, a systematic and controlled approach is needed.

The Testbed project develops a systematic approach to handle change of business processes, particularly aimed at processes in the financial service sector (Franken, Janssen, 1998). A main objective is to give *insight* into the structure of business processes and the relations between them. This insight can be obtained by making *business process models* that clearly and precisely represent the *essence* of the business organisation. These models should encompass different levels of organisational detail, thus allowing to find bottlenecks and to assess the consequences of proposed changes for the customers and the organisation itself; see also Jacobson et al. (1995) and Ould (1995). Formal methods allow for detailed analysis of models and tool support in this process.

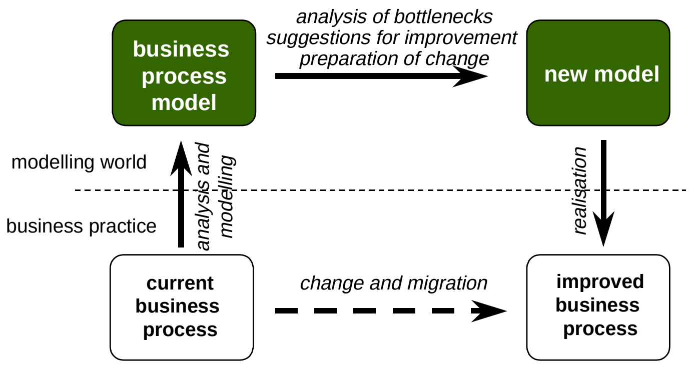


Figure 1. A model-based approach to business process change

As shown in Figure 1, business processes models can be used for analysis and manipulation of business processes without having to actually build these processes first. This model-based approach allows to identify the effects of changes before implementing them. The models constitute an important means for the preparation and actual implementation of organisation and IT change.

In order for such models to be of real help, they have to combine different, competing requirements. On the one hand, they have to be easily accessible and highly understandable to serve as a means of communication between people involved. The primary users of the models are business analysts. They are not especially trained in modelling or formal languages. Moreover, they have to discuss the results with management and people in the shop floor. Thus, the representation should be a *graphical* one: textual representations are not acceptable for the intended users. Moreover, the language should be based on concepts that are relevant to the domain of discourse. It should have a clear *architectural* meaning for business modelling (Vissers, 1994).

On the other hand, the models should be analysable using computer tools and serve as a starting point for (information) systems design. This implies that the models should have a rigorously defined meaning and syntax. Mathematical rigour and communicability are qualities that are not often found in close companionship.

In this paper we introduce a language designed for business process engineering, called AMBER (*Architectural Modelling Box for Enterprise Redesign*). We show the ingredients of the language and how it fulfils the larger part of the requirements needed for business process modelling and analysis. The emphasis in this paper is on the language, and not on the tools and the methodology that support it.

This paper is structured as follows. In section 2 we define requirements for a business process modelling language. Thereafter we mention a number of current approaches and match them to these requirements. Section 3 then gives an overview of AMBER and the means of specialising the language for specific purposes. Section 4 illustrates the approach using a larger example. We end with a summary of our findings and an overview of the current state and plans for (tool) support.

2 Requirements and current state

2.1 Requirements for a business process engineering language

When entering the field of business process analysis and redesign, one is confronted with an overwhelming number of modelling tools and languages. Often, these languages and tools have little in common. Some emphasise elements of workflow in the models, others concentrate on quantitative analysis, yet others try to integrate business processes and supporting information technology. If there are so many different viewpoints on business process modelling, then how should one make a choice?

As the Testbed project is concerned with supporting business process analysts throughout a BPR project, we have judged modelling languages and BPR tools for their suitability in different stages. From this perspective an evaluation framework was developed (Janssen et al., 1997). It has four dimensions of evaluation criteria (see Figure 2).

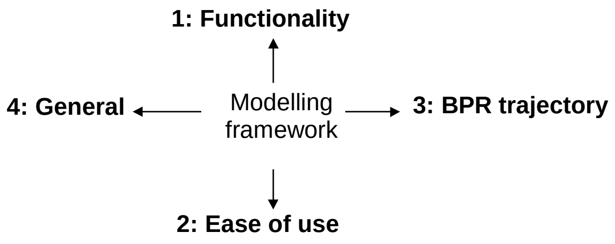


Figure 2. The dimensions of the evaluation framework

The most prominent dimension for modelling languages is *functionality*. It covers:

- *Expressiveness*: we want to express activities and their relations, repetition, and co-operation;
- *Structuring*: the language must support abstraction levels and decomposition;
- *Analysis*: formal semantics are required, extensibility with respect to required analysis parameters;
- *Relevance*: all concepts should be relevant to the domain being modelled.

A second dimension is *ease of use*. This dimension is partly conflicting with the previous one: high functionality often leads to difficulties in using the language due to the number and complexity of the language concepts. As we are aiming at business analysts or business consultants to build and use models, the language should be highly intuitive and communicable.

The third dimension is the *BPR-trajectory*. Modelling business processes is only a small part of BPR. In a BPR-trajectory different steps are distinguished, from developing a corporate strategy to the introduction and implementation of new, redesigned processes. Whether or not languages and tools are helpful in different steps in the BPR trajectory is vital for their applicability. This dimension is often overlooked when evaluating tools and languages. This includes modelling for *design, redesign and migration*: language support for component-based design, sharing model parts, sequences (trajectory) of models.

The last dimension considers *general properties* of (especially) tools supporting the language, e.g. cost, user support and market acceptance. In general, tool support is a prerequisite for broad acceptance. Though processes can be visualised using drawing tools such as ABC Flowcharter, more extensive support is needed to tackle complexity and ensure maintainability.

2.2 Current approaches to business process modelling

The problem of understanding complex system behaviour and the challenge of developing easy-to-use models are apparent in the field of business processes. Until recently, many organisations used only spreadsheet models to forecast the implications of change. Nowadays, more sophisticated modelling and analysis techniques —often based on simulation or workflow techniques— are used in predicting the effects of change.

A simulation model represents a (discrete event) system, setting (input) parameters and observing output parameters and the way they behave in time. For this purpose, many tools have been built, for example based on simulation languages like Simula ’67 (Dahl et al., 1970) or SIMAN (Pegden et al., 1995). Arena (Systems Modelling Corporation) and SimProcess (CACI Products Company) are examples of such tools. The resulting simulation models are often used for problem solving in organisations (Dur, 1992).

For some purposes, simulation is less suited, because it is too time-consuming or does not deliver the level of completeness required. In such cases, analytical techniques for quantitative analysis (Lazowska et al., 1984) can be used. Especially for computer systems performance modelling such techniques are well suited.

Workflow approaches (Lawrence, 1997) tend to use activity models like Petri-nets or role activity diagrams or non-standard modelling techniques to define the business processes and their relations. Workflow is often used to automate or streamline business processes.

When considering people and their positions in an organisation, it is not sufficient to focus only on the procedural aspects of process definition. Aspects of responsibility and accountability must also be taken into account. The theoretical framework of Flores and Winograd (1986), commonly referred to as *speech-act theory*, provides concepts for modelling these issues (Scherr, 1993). Examples of methods and tools built on this theory are ActionWorkflow (Medina-Mora et al., 1992), DEMO (Rijst, Dietz, 1994), and SAMPO (Auramaki et al., 1988).

The Unified Modelling Language UML (Rational, 1997) provides a set of concepts and language elements for different aspects: use-case diagrams show the actors, class and object diagrams define objects and their behaviour, state or scenario diagrams show life-cycles and scenarios, etc. Although the approach was originally developed for software process design, the application of object-oriented design principles as provided by UML for the (re)design of business processes is extensively described by Jacobson (Jacobson et al, 1995).

A limited number of more systematic approaches to *business process design* are found in the literature. One of the most extensive ones is STRIM (Systematic Technique for Role and Interaction Modelling) (Ould, 1995). This method uses Role Activity Diagrams (RADs) to graphically model processes. The method comprises a global approach to model a process, and more detailed guidelines for specific situations. It defines process patterns (a kind of templates) for frequently occurring process types. The Architecture of Integrated Information Systems approach, supported by the tool Aris, can be regarded as a systematic approach (Scheer, 1998) as well. Aris allows for different visualisations of processes for different purposes. Moreover, it provides a powerful repository that can serve a kind of collective organisational memory. The conceptual basis of the approach has certain similarities to the language we use in Testbed. Due to lack of formal foundation, however, analysis is limited. Simulation is possible, but other types of analysis are missing.

If we place the above approaches along the dimension of the evaluation framework, we can see that all of them provide ingredients applicable to business process modelling and analysis. They provide expressiveness (simulation languages, Petri-nets), analysability (simulation, analytical techniques), insight (STRIM, speech-act theory), relative ease of use (workflow modelling tools). However, none of them combine those aspects in a uniform way. Moreover, the fact that business process change involves a series of models (plateaux, migration stages) is missing from almost all approaches mentioned (Janssen et al., 1997).

We may conclude that the existing modelling languages each emphasise some elements, but do not provide an overall solution for business process engineering. This is partially caused by the fact that most of these methods originate from information system development. In the next section we define AMBER and illustrate how this language aims at fulfilling the requirements stated, in a unified, graphical yet formally defined language.

3 AMBER explained

Business modelling languages may be deployed for many different purposes. Not only do they supply a sound foundation for communicating and discussing business process designs, they may be used as well for e.g.

- *analysis* of business processes, that is, assessment of qualities and properties of business process designs, either in quantitative or qualitative terms;
- *model checking*, that is, providing answers to functional queries posed on business models, e.g. “Will every customer request result in an answer?”;
- *export to implementation platforms*, such as workflow management and enterprise resource planning systems;
- *job design*, that is, designing detailed job specifications and generating job instructions; and

- *domain-specific modelling*, that is, incorporating concepts specifically for a certain application domain, such as a business sector.

Every specific purpose of a business modelling language brings about its own specific demands on the language. Yet, it should be possible to use the language for only a limited purpose, without being burdened with the peculiarities of other purposes. Therefore, AMBER was designed to have a lean core language, containing the basic concepts underlying any of the purposes served. On top, it can be tailored for specific purposes by means of a specialisation mechanism called ‘profiles’. The top of the pie holds the modelling and analysis extensions, the bottom contains the extensions that can be used to map models to real-world implementations. This design makes AMBER flexible, extendible, and comprehensible (Figure 3).

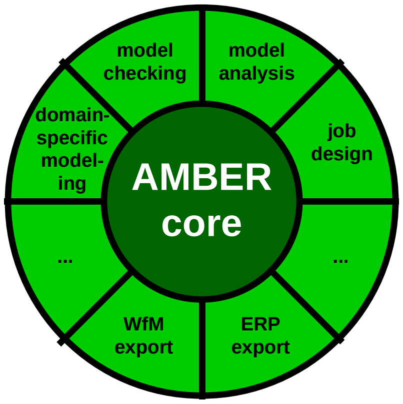


Figure 3. AMBER’s pie-wise design

3.1 The core language

The core of the business modelling language contains concepts that enable basic reasoning about business processes. AMBER recognises three aspect domains:

- the *actor* domain, which allows for describing the organisations, departments, systems, and people carrying out business processes;
- the *behaviour* domain, which allows for describing what happens in a business process;
- the *item* domain, which allows for describing the items handled in business processes.

The actor domain

The basic concept in the actor domain is the *actor*. It designates a resource (used for) carrying out a business process. Actors are structured: they may contain other actors. Also, they are related. Therefore, actors are equipped with *interaction points*, indicating physical or logical locations at which the actor may interact with its environment. They are the hooks where interaction-point relations couple actors. Interaction points may be involved in multiple relations. Also, an interaction-point relation may involve more than two interaction points.

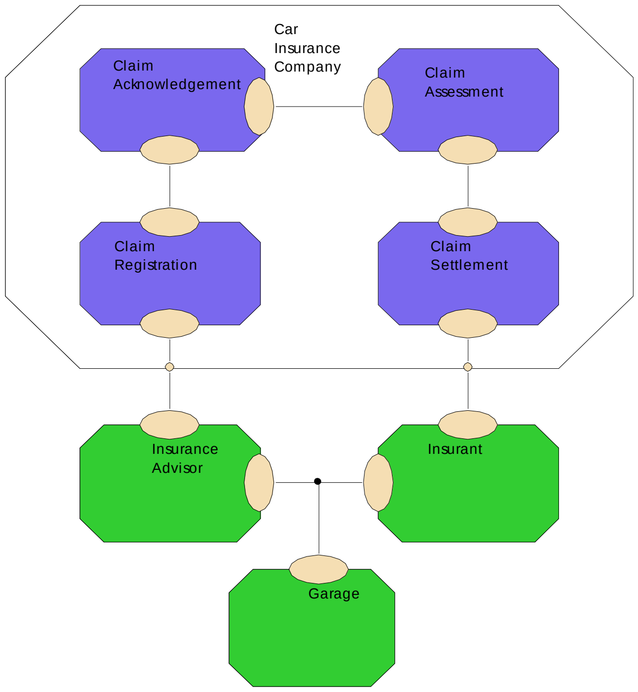


Figure 4. A typical actor model

Figure 4 depicts a typical actor model, showing the parties involved in car insurance claims. It includes a triple interaction-point relation, between the insurant, his garage, and his insurance adviser. It also shows the internal structure of the car insurance company, that is, the four departments involved in processing the claim. The legend of this domain is given in Figure 5. Note that colours have no formal meaning in these models.

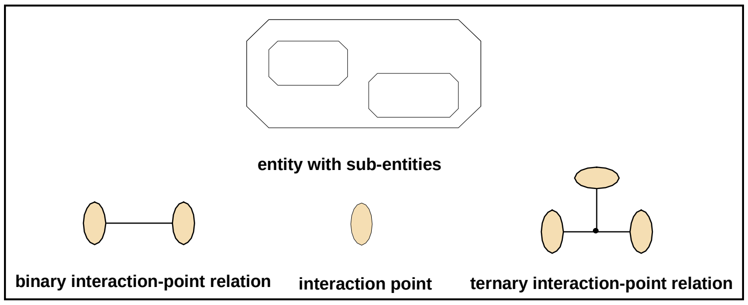

entity with sub-entities
binary interaction-point relation
interaction point
ternary interaction-point relation

Figure 5. Ingredients of the actor domain

Actions, relations and behaviour structuring

The basic concept in the behaviour domain is *action*. It models a unit of activity in business processes. Actions carry two essential attributes: the actors involved in the action and the result in terms of their outputs.

An action can only happen when its *enabling condition* is satisfied. These conditions are formulated in terms of other actions having occurred yet, or not. The most simple is the enabling relation. When put between actions *a* and *b*, it models the fact that *b* cannot happen but after *a* has finished. Its mirror image is the disabling relation, modelling that *b* cannot happen any more, once *a* has occurred.

Causality relations can be composed using Boolean-like operators. Also, they can be supplied with additional *constraints*, which further restrict the relation with conditions on attribute values of preceding actions.

A special kind of action, used in many business process models, is a *trigger*. Triggers are like actions, accept that they have no causality condition, that is, they are immediately enabled.

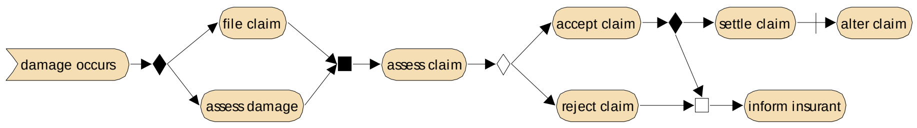


Figure 6. A typical unstructured behaviour model

Figure 6 depicts a typical behaviour model, containing a sequence of actions, related by enabling conditions. The occurrence of damage triggers both the filing of a claim as well as the assessment of the damage. Only when both have been carried out, the claim is assessed and then either accepted or rejected. In any case, the insurant is informed about the outcome. Only in case of acceptance, the claim is settled. After settlement, the claim cannot be changed. We did not model what happens when claims are altered. The legend is shown in Figure 7.

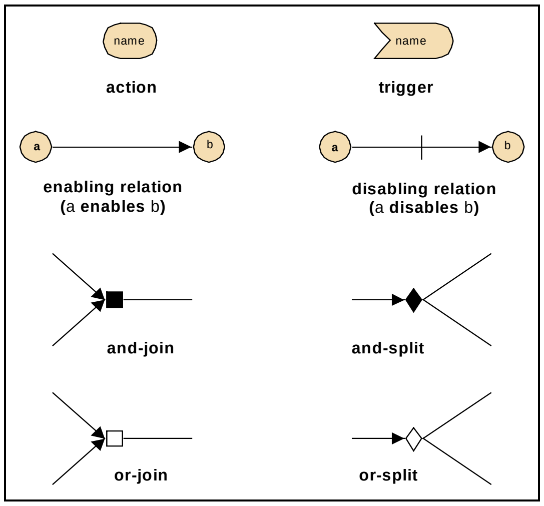


Figure 7. Actions and enabling conditions

Extensive modelling of all relevant actions in a business process, and their enabling conditions, will result in huge and cluttered models for any serious business process. Therefore, structuring concepts are included in AMBER to tackle the complexity of behaviour models.

Behaviour can be grouped in *blocks*. Like actors, blocks can be nested. There are basically two points at which some behaviour can be separated from its environment. One is *between* actions, the other is *inside* actions.

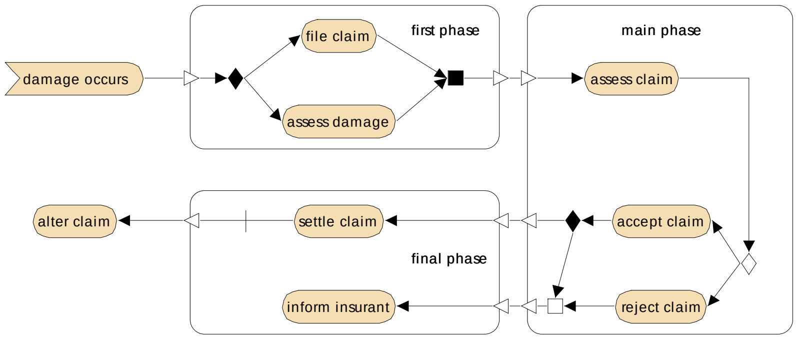


Figure 8. A phased behaviour model

When a block separates behaviour between actions, a causality relation is cut. A so-called entry or exit, depending on the direction of the causality relation, indicates the cutting point (at the block's edge). This type of structuring is typically, but not exclusively, used for structuring in phases. Figure 8 shows a phased version of Figure 6.

When a block separates behaviour inside actions, the action is divided in a number of *interactions*. Interactions are related like interaction points in the actor domain. An interaction can only happen simultaneously with all of its related interactions, which must therefore all be enabled. This type of structuring is typically used for modelling interaction between model elements. Figure 9 shows a version of Figure 6, in which this type of structuring is used.

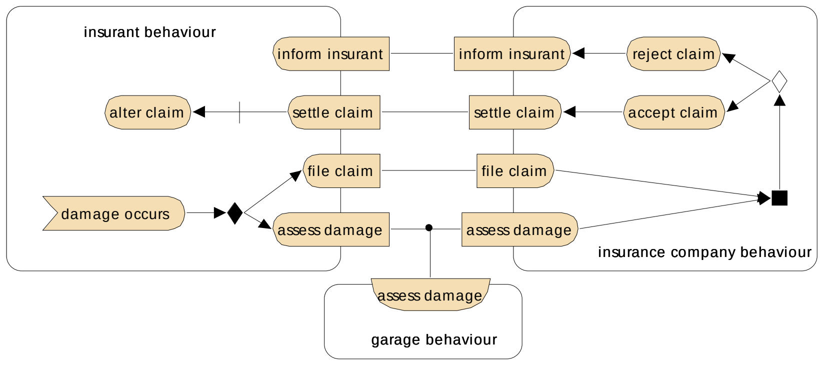
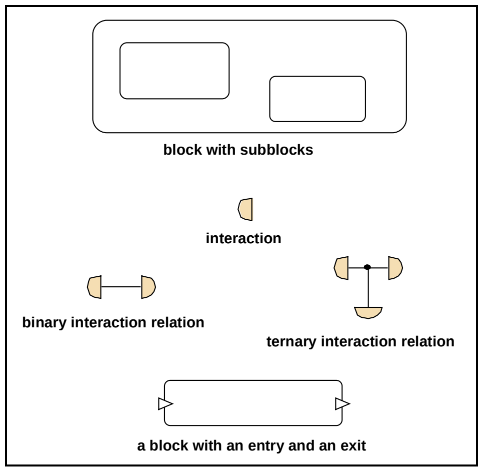


Figure 9. Interacting blocks

Although separately introduced here, phased and interaction-based structured may be arbitrarily mixed.



block with subblocks
interaction
binary interaction relation
ternary interaction relation
a block with an entry and an exit

Figure 10. Behaviour structuring

Next to structuring, another way to tackle complexity in business models is to allow similar elements to be described at once, as a group, instead of having to copy each of them out individually. AMBER offers two facilities for this: replication and iteration.

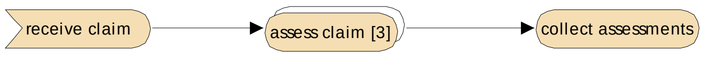


Figure 11. A replicated action

Replication can be used for actors, actions, interactions, and blocks, and indicates that a number of replicas exist next to each other. Figure 11 shows a typical example, in which a received claim is assessed by each of three assessment actions, e.g. as a triple check for high-value claims. When all three assessments have finished, their results are collected.

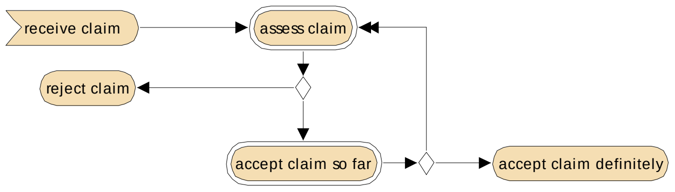


Figure 12. Iterative behaviour

Iteration is used for modelling repeated behaviour, that is, the repeated occurrence of similar behaviour over time. Figure 12 shows an example. Here, different assessment actions are carried out subsequently, for instance for checking different aspects of the claim in different turns. Every assessment may result in (definite) rejection of the claim, or in partial acceptance. When all aspects have been assessed, partial acceptance may result in definite acceptance. Notice that all actions involved in the loop carry double edges and a double-headed arrow separates different loop traversals. The legend of replication and iteration is shown in Figure 13.

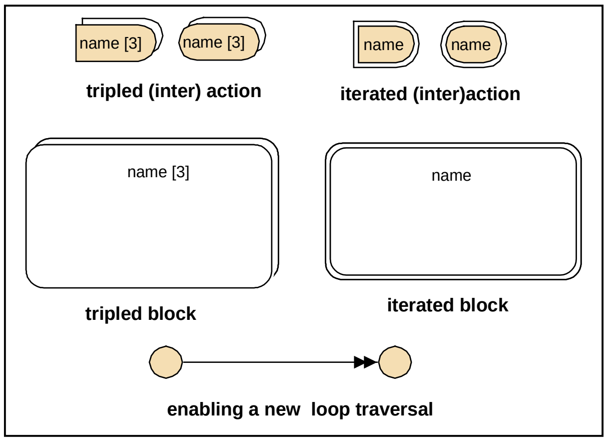


Figure 13. Replication and iteration

The item domain
The item domain enables to model the items on which the behaviour is performed. In the actor and behaviour models, items can be included and coupled to the various elements of these models. In the actor domain, items are coupled to interaction point relations, indicating that at the interaction point relation involved, the indicated item is used. Figure 14 shows a straightforward example.

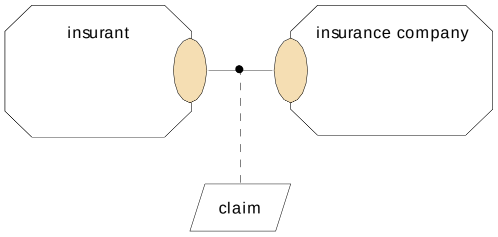


Figure 14. An item coupled to an interaction point relation

Coupling items to elements of behaviour models is different in two ways. First, the items are coupled to actions and interactions, instead of to interaction relations. This is because interaction point relations are symmetric (that is, the interaction points involved cannot have different roles), whereas interaction relations are not: each interaction involved may have its own distinctive contribution to the relation.

Second, item coupling in behaviour models distinguishes between five modes: create, read, change, read/change, and delete. This mode indicates what type of action is performed on the item involved. Figure 15 shows an example in which these respective modes occur, from left to right.

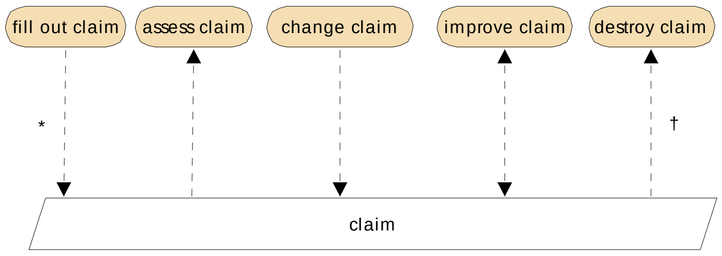


Figure 15. An item, coupled to actions in five ways

Notice that the example includes one single item, coupled to different actions. This way of representing an item in an actor or behaviour model is called an *item bar* and is very much like in IBM’s LOVEM (Helton et al., 1995). It is possible to use a different item bar for each coupling and let each of them refer to the same item.

Currently, AMBER does not include an item modelling language. Hence, so far items are not structured, nor related. At the moment, the inclusion of such a language is matter of investigation. A subset of UML will be used, extended with the link to process models.

3.2 Profiles for specialisation[1]

The concept of 'profile' is introduced in AMBER to allow for more tailored definitions of concepts, even by end-users. This tailoring is useful for a variety of reasons:

- Export mechanisms. For instance, when a model is being translated into a model for a workflow engine, additional parameters should be given for actions, like the executable that should be started for an end-user.
- Specific analysis tools. For quantitative analysis, for instance, aspects like average waiting time, or costs per activity or per resource are necessary. It is only useful to specify these attributes if the analysis is used. When modelling for a workflow engine, cost-aspects are rarely necessary and therefore do not need to be part of the model.
- Context-specific information. Some companies may have a need to specify company-specific information (like additional information for organisational units). Such information is only useful for models that are relevant in that company, and should therefore be visible only in these models.

The profile concept allows us to realise the structure visualised in Figure 3, where the AMBER core is re-used for various analyses and/or export mechanisms. Typically, for each of these pie charts one or more profiles are defined that contain the additional attribute definitions necessary to perform the analysis techniques or generate specific export formats. The profile concept makes it possible to use the same basic model for different purposes, by associating the necessary information with modelling concepts. Models need not contain all information, but only that information required for the analyses or exports desired. The Testbed methodology helps the business engineer in determining his or her information and analysis needs.

A profile is defined in a profile definition language containing type-definition constructs (for re-usable types) and supports sub-classing of profile definitions.

```
type Money    = alias real;
type TimeCat = enum {hour, day, week, month,
year};
type Rate = struct {
      Money        amount;
      TimeCat      period;
};

Profile Costs {
  assignable to Actor;
  Money      fixed = 0.0;
  Rate variable ={amount:100.0, period:hour};
};
```

[1] patent pending

The example above shows that there are a number of means to define types (aliasing of built-in types, definitions via structures, unions, enumerations, and collections). The syntax resembles that of the C-programming language. We also support a notion of single inheritance in the profile definitions. For instance, to declare a cost-profile that by default has a fixed rate and an empty variable rate you would define:

```
Profile FixedCosts isA Costs {
  Money      fixed  = 25.0;
  Rate variable;
};
```

3.3 Formalisation for analysis

One of the requirements for our language is *analysability*. The AMBER language combines organisational, behavioural, and data elements. For all these elements analysis should be possible, preferably in combination. Formal methods provide analysability. The current state of affairs, however, is that there in no sound mathematical model available in the literature that combines those different elements and still allows for tool supported analysis.

We therefore have taken a different approach. Instead of thriving for a single, unified semantic foundation, different non-conflicting semantic models are used for different aspects. Thus it becomes possible to develop means of analysis that function for realistic business cases. One loses, however, the possibility to analyse relations between aspects that are in different semantic domains.

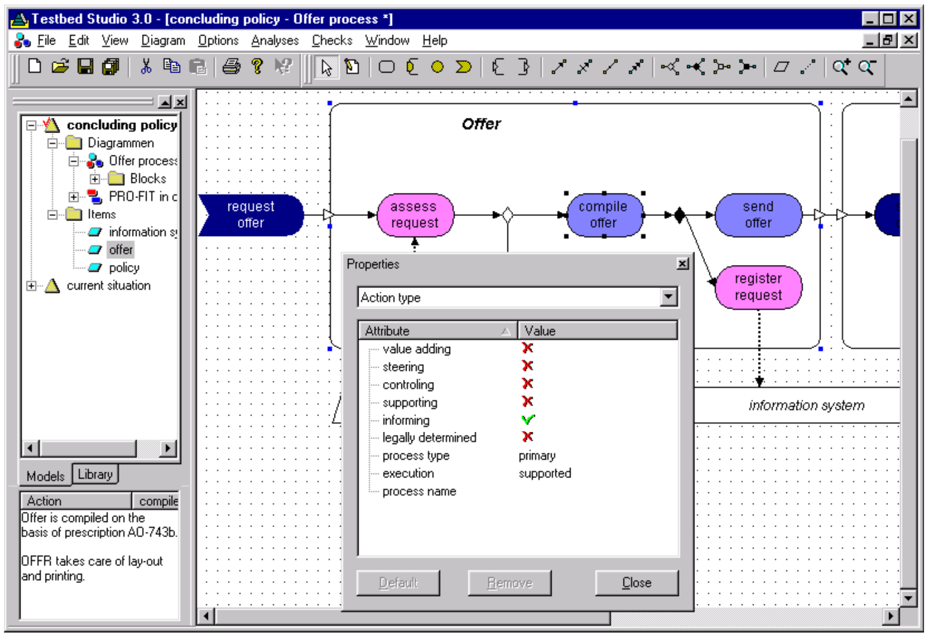
Many structural analyses are not really based on the underlying semantics, but on the formal syntax. Such analyses, called *views*, allow to highlight specific structural aspects of models, such as "what activities can precede this action?", or show the relationship between different modelling domain ("show what actors perform the activities", or a dataflow view of behaviour). These views make extensive use of information in profiles associated with objects in the model.

An operational semantics has been defined to analyse certain behavioural properties. It defines the meaning of processes in terms of state automata (Janssen et al., 1998). It allows for both an interleaving interpretation as well as a multistep interpretation. The operational semantics forms the basis of the stepwise simulation in the tool. Functional analysis using model checking is derived from the same semantics. We have defined a translation from the finite-state fragment of AMBER to the input language of the model checker Spin, called Promela (Holzman, 1997). This allows for a full state space verification of temporal properties of AMBER models (Janssen et al., 1998). To define verification properties for functional analysis a pattern-based interface is defined, allowing to check for sequences of activities, necessary precedence or consequence, and combinations of occurrences. (Janssen et al., 1999). These patterns are then translated to linear-time temporal logic and thereafter transformed to Spin queries (never claims).

For quantitative properties such an operational semantics is of no use. Instead, we use analytical techniques for performance analysis, based on queuing theory, graph models and hybrid models (Jonkers et al., 1998). Analysis of completion times, critical paths, resource utilisation, and cost analysis are currently available.

As our data language is not fully developed as yet, no particular means of analysis, other than structural analysis, are provided for items.

Finally, the tool allows to transform models to emphasise properties of the model. A powerful concept is that of *process lanes*. A business process model can be automatically structured with respect to any attribute. For example, a process can be structured into a single block (sub-process) for every actor involved, showing the change in responsibility in a process. Alternatively, the process could be structured with respect to the business function associated with an activity, or whether the activity belongs to the primary or secondary part of the process.



Action type

Attribute	Value
value adding	✗
steering	✗
controling	✗
supporting	✗
informing	✓
legally determined	✗
process type	primary
execution	supported
process name	

Action	compile

Offer is compiled on the basis of prescription AO-743b.

OFFR takes care of lay-out and printing.

Figure 16. Tool support for AMBER in Testbed Studio

3.4 Tool support

The language is supported by an integrated set of tool components, together referred to as *Testbed Studio.*

Testbed Studio has the following functionality:

- easy editing of models;

- quantitative analysis of models using various analysis methods;
- functional analysis of models using model checking;
- report generation;
- different views on models;
- step-by-step animation of models;
- management of trajectories of models (refinement of models, versioning);
- component libraries, to allow for re-use of business models.

Testbed Studio runs in the Windows environment. An example screen is shown in Figure 16.

4 AMBER illustrated

During the past three years, AMBER has been applied in over 30 pilot studies within the Testbed project. These pilots have actually been the steering wheel of the project. Initially, the pilots have been used to study the essential concepts of business process and formulate requirements on languages, methods and tools that should support the business consultants. Later on, the pilots were used for validating results. Only by doing real-life projects and solving real business problems one can optimise its languages, methods, and tools. Gradually, the cases became more complex, evolving from modelling to analysis to redesign and implementation studies. Sample projects concern decreasing the completion time of a insurance process with 30%, modelling and analysing a business function architecture, analysis of the relationship between a pension process and the organisations in its context it depends upon and so on/

In this section we present an imaginary redesign case concerning a car insurance company. It is a realistic case because it is a generalisation of real-life cases. In the following, we restrict ourselves to only one aspect of the overall Testbed approach for business process analysis and redesign: the use of modelling. The overall approach is discussed in Franken and Janssen (1998).

4.1 Modelling for analysis

Consider RR, a company selling property insurance for cars. Recently, insurance sales have been decreasing. Only 25% of the quotation requests result in an insurance policy. The norm is 40%, however. Testbed and AMBER are used to find the bottlenecks, suggest improvements, and analyse the impact of changes. From RR management we find out that the service level provided is the most important success factor of RR. Costs are in second place. A constraint is that the division of RR in departments should not be changed. Finally, we get the suggestion that state-of-the-art technology should be used. We first model the actors involved (Figure 17).

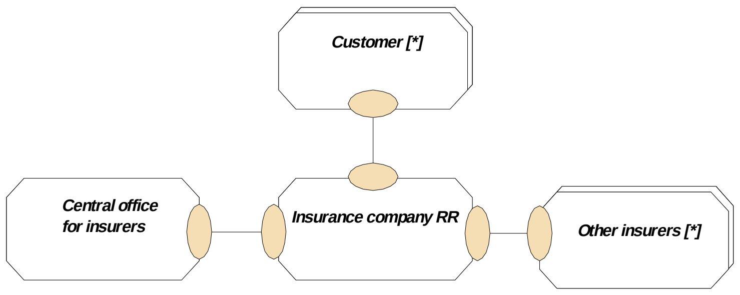


Figure 17. Actors involved in Claim Handling process

Next, we determine the service provided by RR and elaborate the internal processes (Figure 18)

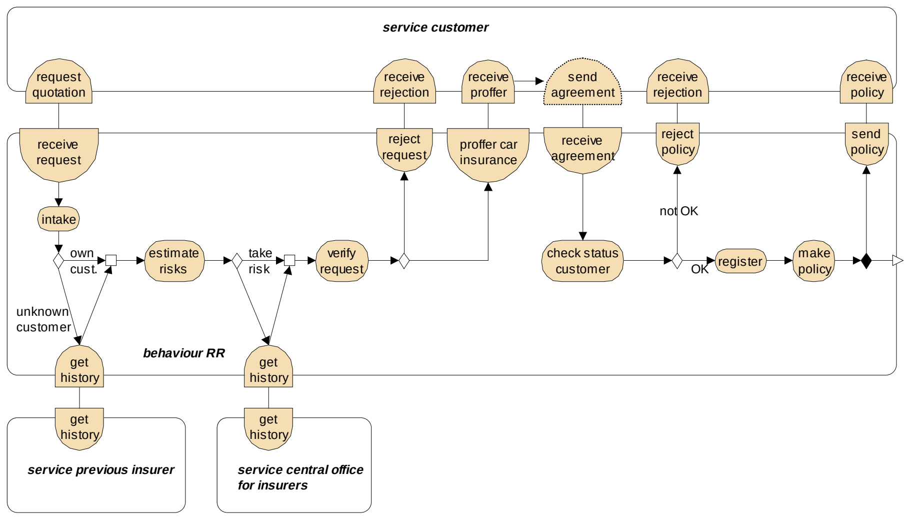


Figure 18. The Claim Handling process

The above behaviour is a *logical* model because it abstracts from physical implementation features. It just shows the request for quotation process. The exit shows the process continues (with the payment collection, which we omitted in the figure).

Using a number of iterations, the model may be extended and improved. For example, the department ‘Sales’, ‘Policy Management’ and ‘Finance’ may be distinguished and the behaviour may be structured correspondingly. The duration of actions may be specified, and probabilities may be added to each or-split. The resulting model is a *physical* model, showing the current implementation of activities. This model, with the *profile* for execution times filled out, can be used as input for analysis tools showing the critical path or response times.

4.2 Model-based analysis

The physical models are not shown, but in Figure 19 some results are presented to give an impression of Testbed analyses. The left plot shows the response time from ‘request quotation’ until either ‘receive rejection’ or ‘receive proffer’. The average response time is 12.7 days, with 50% and 90% percent of the request handled in 13 respectively 18 days.

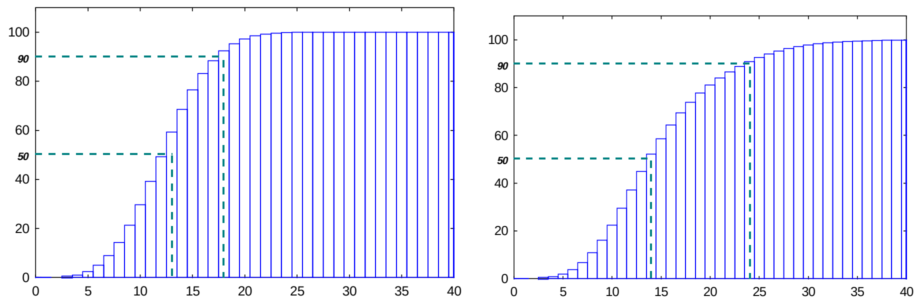


Figure 19. Completions times of Claim Handling proces

The right plot shows the response time from ‘request quotation’ till either ‘receive rejection’ or ‘receive policy’ or ‘receive proffer’ (in case of no customer response). The average response time is 15.3 days, with 50% and 90% percent of the request handled in 14 respectively 24 days.

These results were analytically obtained using completion time analysis in Testbed Studio. It does not require extensive simulation.

Modelling and analysis reveal the following facts:

- *Customer satisfaction* — Response to quotation request takes 12.7 days. After customer agreement, policy may be rejected.
- *The number of customer interactions* — Customer interacts with all three departments. Customer may be contacted twice before receiving quotation (not shown in the model above).
- *Performance indicators* — 25% of request results in policy. Overall response time is 15.3 days.
- *The logical flow of processes* — All customers are treated equally.

These facts are a starting point for the development of improved processes on the basis of the models of the current situation. The models of the improved situation can then be compared to the original situation, to assess the impact on people, throughput, cost, completion times and so on.

5 Conclusions and future work

In this paper a language was presented, suited to model, analyse, and redesign business processes. Using different semantic models for the language, and embedding the language in a methodology for business process redesign supported by tools, has lead to a powerful integrated approach. The approach has proven itself in numerous cases and shows that people that are not aware of intricacies of applying formal methods in general can apply formal methods. Our approach effectively supports business analysts in their daily work: business process engineering.

The link to implementation is not yet well supported by the approach. Amongst others, this requires incorporation of a language to model data. In our user audience UML is gaining increased popularity. Combing UML-like definition with formal process models is one of the challenges we face in the time coming. On the basis of UML, the link to CASE tools is currently under investigation.

Acknowledgement

We would like to thank the anonymous referees for detailed suggestions and Henry Franken for general comments on the paper.

This paper results from the Testbed project, a 120 man-year research initiative that focuses on a virtual test environment for business processes. The Testbed consortium consists of ABP, the Dutch Tax Department, ING Group, IBM and the Telematica Instituut (The Netherlands) and co-operates with several Dutch universities and research institutes. The project is financially supported by the Dutch Ministry of Economic Affairs. We appreciate to acknowledge all Testbed contributors.

References

Auramaki, E., E. Lehtinen, and K. Lyytinen, A speech-act-based office modelling approach, *ACM Transactions on Office Information Systems*, Vol. 6), No. 2, April 1988, 126-152.

Browne, J., P. Bradley, S. Jackson, and H. Jagdev, Business process re-engineering (BPR) - A Study of the software tools currently available. *Computers in Industry*, 25 (1995), p. 309-330.

Dahl, O.-J., B. Myrhrhaug, and K. Nygaard. *SIMULA 67 Common Base Language*. Norwegian Computing Centre, Oslo, 1970, Publication N. S-22.

Dur, R.C.J., *Business reengineering in information intensive organisations*. Ph.D. Thesis, Delft University of Technology, 1992.

Franken, H.M., and W. Janssen, Get a grip on changing business processes, *Knowledge & Process Management* (Wiley), Winter 1998.

Hansen, G. A., Tools for business process reengineering. *IEEE Software*, September 1994, p. 131-133.

Helton, A., E. Zulaybar, and P. Soper. *Business Process Engineering and Beyond*. IBM Redbook. Publication No. SG24-2590-00, 1995.

Holzman, G.J., The model checker SPIN, *IEEE Transactions on Software Engineering*, Vol. 23, No. 5, May 1997, 279-295.

Jacobson, I., M. Ericsson, and A. Jacobson, *The Object Advantage - Business Process Reengineering with Object Technology*, ACM Books, 1995.

Janssen, W., H. Jonkers, and J.P.C. Verhoosel, What makes business processes special? An evaluation framework for modelling languages and tools in Business Process Redesign. In Siau, Wand and Parsons (eds.), *Proceedings 2nd CAiSE/IFIP 8.1 international workshop on evaluation of modelling methods in systems analysis and design*, Barcelona, June 1997. (Available as http://www.telin.nl/publicaties/1997/caise97_final.doc)

Janssen, W., R. Mateescu, S. Mauw, and J. Springintveld, Verifying Business Processes using Spin. In G. Holzman, E. Najm, and A. Serhrouchni (eds.), Proceedings 4th International SPIN Workshop. Report ENST 98 S 002, pp. 21-36. Ecole Nationale Superieure des Telecommunications, Paris, France. November, 1998.

Janssen, W., R. Mateescu, S. Mauw, P. Fennema, and P. van der Stappen, Model checking for managers. In Proceedings 6th International SPIN Workshop on Practical Aspects of Model Checking. Toulouse, France. September 1999.

Jonkers, H., W. Janssen, A. Verschut, and E. Wierstra, A unified framework for design and performance analysis of distributed systems. Paper released to *IPDS'98, IEEE International Performance and Dependability Symposium*, Durham, NC, USA, 7-9 September 1998.

Lawrence, P. (Ed.), *Workflow Handbook*, John Wiley & Sons Ltd, Chichester, UK, 1997.

Lazowska, E.D., J. Zahorjan, G. Graham, and K. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, 1984.

Medina-Mora, R., T. Winograd, R. Flores, and F. Flores, The action workflow approach to workflow management technology. In *Proceedings of CSCW '92*, November 1992, pp. 281-288.

Ould, M.A., *Business Processes: Modelling and analysis for re-engineering and improvement*, John Wiley & Sons, Chichester, England, 1995.

Pegden, C.D., R.R. Shannon, and R.P. Sadowski, *Introduction to Simulation Using SIMAN*. Second ed. McGraw-Hill, 1995.

Quartel, D.A.C., L. Ferreira Pires, M.J. van Sinderen, H.M. Franken, and C.A. Vissers, On the role of basic design concepts in behaviour structuring, *Computer Networks and ISDN Systems* 29:4, March 1997, pp. 413-436.

Rational Software Corporation, *Unified Modeling Language, Version 1.0*, 1997 [Unpublished report]. http://www.rational.com/ot/uml/1.0/index.html.

Rijst, N.B.J. van der, and J.L.G. Dietz, Expressing production control principles in the DEMO communication model. In: A. Verbraeck, H.G. Sol and P.W.G. Bots (Eds.), *Proceedings of the Fourth International Working Conference on Dynamic Modelling of Information Systems*, Delft University Press, Delft, The Netherlands, 1994, pp 171-186.

Scheer, A.-W., *ARIS – Business Process Frameworks*. Springer-Verlag, 1998.

Scherr, A.L., A new approach to business processes, *IBM Systems Journal*, Vol. 32, No. 1, 1993, pp. 80-98.

Vissers, C.A., Report on the architectural semantics workshop. In: J. de Meer, B. Mahr and S.Storp (eds.), Proceeding of International Conference on Open Distributed Processing, IFIP, pp. 367-386. North-Holland, 1994.

Winograd, T., and F. Flores, *Understanding computers and cognition: A new foundation for design*, Ablex, Norwood, NJ, 1986.