BUSINESS PROCESS AND MOTIVATION OF OBJECTIVES IN ONE MODEL

Ella Roubtsova
Open University of the Netherlands
Valkenburgerweg 177
6419AT Heerlen,
The Netherlands
Ella.Roubtsova@ou.nl

Keywords: Business Process, Business Objectives, Protocol Model, Motivation Model

Abstract: The Object Management Group predicts that the Business Process Modelling Notation will be eventually merged with the Business Motivation Model to be implemented in integrated tool suites. However, conventional modelling semantics have asynchronous semantics and therefore have difficulties to accommodate motivation of objectives specified on the basis of synchronous semantics. This paper shows how Protocol Modelling semantics can be used both for business process modelling and motivation modelling corresponding to objectives. Protocol Modelling uses synchronous composition and this synchronization gives to Protocol Modelling the expressive means needed to accommodate motivation of objectives and business processes in one model.

1 INTRODUCTION

Goals, objectives and motives are very important parts of system specification. Goals are usually formulated as non-functional requirements. They are abstract. The goals can be even unrealizable. The objectives corresponding to goals are specific and measurable. They show realisability of goals. Presentation of goals, objectives and motives in business process specification can be seen as transformation of goals into the corresponding objectives and motives expressed as elements of business processes. Such a transformation is a way to estimate realisability of goals.

The need of combining goals and business process modelling standards is emphasized by the Object Management Group and the Business Rules Group. They predict that "eventually specifications such as the Business Process Modelling Notation (BPMN) together with the Business Motivation Model (BMM) should be merged into a single business-oriented modelling architecture, and implemented in integrated tool suites" (OMG, 2010; BRG, 2010).

In this paper we relate the BMM with business processes and show how business modelers can benefit from modelling of motivation of objectives.

The structure of the paper is the following.

Section 2 presents elements of the Business Motivation Model (BMM).

Section 3 formulates semantic problems of combining goals and business processes in one model identified in related work.

Section 4 formally presents the semantic basis for motivation modelling.

Section 5 shows how the Protocol Modelling semantics can accommodate motivation of objectives and business process in one model.

Section 6 describes applications of motivation models as future work and concludes the paper.

2 BUSINESS MOTIVATION MODEL

The BMM provides a structure for developing, communicating, and managing business plans. The structure covers four related elements:

- The *Ends* of a business plan.
  "Among the *Ends* are things the enterprise wishes to achieve, for example, *Goals* and *Objectives*" (BRG, 2010).

- The *Means* of a business plan.
  "Among the *Means* are things the enterprise will employ to achieve the *Ends*, for example, *Strategies, Tactics, Business Policies*, and *Business Rules*".

- " The *Influences* that shape elements of a business plan".

- "The *Assessments* that are made about the impacts of such *Influencers* on *Ends* and *Means* i.e., *Strengths, Weaknesses, Opportunities, and Threats*."

The OMG predicts that "three types of people are expected to benefit from the BMM: developers of business plans, business modellers, and implementers of software tools and repositories".

The BMM is not a full business model and it does not prescribe in detail business processes, workflows and business vocabulary. However, business processes are key elements of business plans and the BMM does include a placeholder for Business Processes. The relations between *Goals* and other elements of BMM are left open.

3 GOAL MODELLING

Goal modelling has its roots in the well known requirements engineering approach KAOS (Knowledge Acquisition in autOmated Specification) (Dardenne et al., 1993). Goals are specified in Linear Temporal Logic and organized using the AND and OR refinement structures.

Van at al (Van et al., 2004) proposed goal-oriented requirements animation. The modelling formalism is the UML State Machines that are generated from the goal specifications and called Goal State Machines (GSMs). A GSM contains only transitions that are justified by goals. The GSMs receive events through event broadcast. A GSM that can't accept an event in its current state keeps it in a queue. These events will be submitted to GSMs internally. This means that the composition of GSMs contains extra states that cannot be composed from the states of separate GSMs. Therefore the GSMs cannot be seen as purely goal models as they also deal with the events from the queues.

The User Requirements Notation (URN) (ITU, 2008) is a standard that recommends languages for software development in telecommunication. The URN consists of the Goal-Oriented Requirements Language (GRL), based on i* modelling framework (Yu, 1995), and Use Case Maps (UCM) (Alsumait et al., 2003), a scenario modelling notation. The GRL provides a notation for modelling goals and rationales, and strategic relationships among social actors (Yu et al., 2001). It is used to explore and identify system requirements, including especially non-functional requirements. The UCM is a convenient notation to represent use cases. The use cases are selected paths in the system behaviour and they can be related to goals by developers. The goals are used to prioritize some use cases. If a use case presents alternative behaviours or cycles, then the goals prioritize alternatives. The use cases can be simulated. However, use cases do not model data and the state of the system and they present only selected traces. This means that behaviour model as well as the motivation model shown by use cases are incomplete and cannot guarantee the achievement of goals in the whole system.

Letier at al. (Letier et al., 2008) derive event-based transition systems from goal-oriented requirements models. Then the operations are derived from goals as triples of domain pre-conditions, trigger-conditions and post-conditions for each state transition. The declarative goal statements are transformed into the operational model. To produce consistent operational models, a required trigger-condition on an operation must imply the conjunction of its required preconditions. The problems of goal-oriented approaches are mostly caused by different semantics used by process modelling and goal modelling techniques. Letier at al (Letier et al., 2008) explained that the operational specification and the KAOS goal models use different formalisms. KAOS uses synchronous temporal logics that are interpreted over sequences of states observed at a fixed time rate. The operational models use asynchronous temporal logics that are interpreted over sequences of states observed after each occurrence of an event. Temporal logic operators have very different meanings in synchronous and asynchronous temporal logics. Most operational formalisms have the asynchronous semantics. Letier at al. (Letier et al., 2008) admit that in order to be semantically equivalent to the synchronous KAOS models, the derived event-based models need to refer explicitly to timing events or include elements of synchronization.

4 SEMANTICS FOR MOTIVATION MODELING

The need of synchronization is not the only one semantic need to direct business processes to objec-

tives. Let us identify the necessary semantics in a state transition system.

We take a state transition system which is usually presented as a triple of

$$P = (S, A, T), where$$

- S is a finite set of states $\{s_1, ...s_i, ...s_j...\}$,
- A is the alphabet of P, a finite set of environmental actions ranged over $\{a, b, ...\}$,
- T is a finite set of transitions (s_i, a, s_j).

The semantics of a transition contains two relations (Milner, 1980):

1. $C \subseteq (A \times S)$ is a binary relation, where $(a, s) \in C$ means that action a is a possible action for P when in state s. C is called the *can-model* of P because it models the actions that P "can do" in each state.
2. U is a total mapping $C \rightarrow S$ that defines for each member of C the new state that P adopts as a result of the action. $U(a; s_i) = s_j$ means that if P engages in action a when in state s_i it will then adopt state s_j. U is called the *update-model* of P because it models the update to the state of P that results from engagement in an action.

With separation of the can- and update-models a process P is a tuple:

$$P = (S; A; C; U).$$

There are always states in the process where particular goals are achieved. Let us name them the goal-states. From the goal perspective, the actions, leading to a goal-state, are the priority actions or wanted actions in the states preceding the goal-state. So, a state preceding a goal-state and the action that is on the path may lead to the goal-state, form a new binary relation:

- $W \subset (A \times S)$, $(a; s) \in W$ means that action a is a wanted action for P when in state s. We call relation W the *want-model* to show its semantic difference from the relation C.

In order to model motivation we propose to add the want-model W to the process:

$$P = (S; A; C; U; W).$$

The can- and want-models of a process are independent of each other, so when a process is in a given state, an action can have different combinations of can- and want- alternatives:

$$\{can\ happen;\ can\ not\ happen\} \times \{wanted; not\ wanted\}$$

Usually $W \subseteq C$ and W is included into the process model. However, the new goals emerging in the life cycle of the modeled system may challenge the process and may need actions that do not belong to the alphabet A.

In this paper we base the modeling of motivation on this extra relation W added to the process.

A service may have several (n) goals. In this case several want-models should be taken into account

$$P = (S; A; C; U; W_{G1}, ..., W_{Gn}).$$

The goals can be OR-composed or AND-composed (Pohl and Rupp, 2011).

In real systems, some goals can be conflicting. For instance, information goals may conflict with security and privacy goals. Wishes of different user roles may also conflict. Two goals are conflicting if the system has a state from which it is impossible to reach a state where both goals are satisfied simultaneously. It is important to identify any conflicting goals and corresponding motivation models as soon as possible in the software life cycle. One of the ways to do this is modelling of motivation corresponding to objectives in process models.

5 PROTOCOL MODELS WITH MOTIVATION MODELS

A Protocol Model is a synchronous CSP parallel composition of protocol machines (McNeile and Simons, 2006). This composition has its roots in the algebra Communicating Sequential Processes (CSP) proposed by Hoare (Hoare, 1985). McNeile (McNeile and Simons, 2006) extended this composition for machines with data.

Protocol Modelling semantics accumulates can- and want- semantics needed for accommodation of objectives in business processes.

We will demonstrate the use of Protocol Modelling for business process and motivation modelling on a simple case study.

An *Insert Credit Card Number* web service can be seen in many electronic booking systems. The behaviour of the service is the following. The user of the service instantiates the service. The user is asked to insert his credit card number and read the privacy conditions of the service. The user may insert the credit card number without reading the privacy conditions and after reading and accepting the privacy conditions. When the user has accepted the privacy conditions, he can rethink and read the statement again. The service can be cancelled before inserting the credit card number.

We recognize two goals for this service (Figure 1), namely,

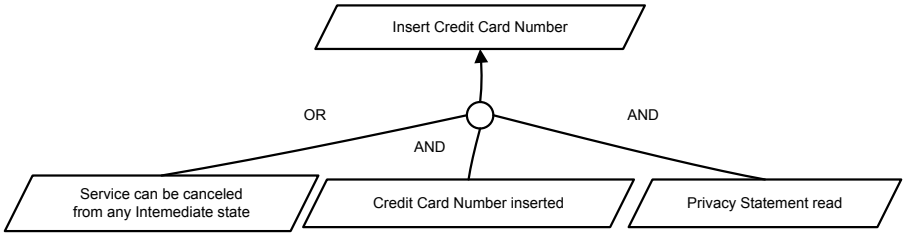


Figure 1: Goal Model

- to get the credit card number inserted and
- to get the privacy conditions read by the user.

The possibility of service cancelation is yet another concern. It is obvious that cancelation cannot be called a goal of the service.

5.1 Business Process

After the identification of the goals the KAOS approach suggests to identify objects, agents, entities and operations.

Using Protocol Modelling we also identify entities, objects, agents and yet aspects but all of them are presented as protocol machines.

For example, we model the can-update process of the *Insert Credit Card Number* web service as a CSP composition of protocol machines `Input`, `Decision` and `Cancelation`. These protocol machines correspond to formulated goals and the cancelation requirement. Figure 2 shows the graphical presentation of our model. The executable Modelscope metacode is shown in Figure 2. The metacode is the complete artefact. As we show later the graphical form does not contain all the modelling constructs of protocol models.

It is not always the case that one requirement is mapped onto one protocol machine. However, the compositional protocol machines allows for any ways of decomposition.

The protocol machine `Input` describes behaviour of an `OBJECT` of type `Input`. Each object has its identification name.

The protocol machines `Decision` and `Cancelation` specify `BEHAVIOURS`. They do not have own identification name and included into each instance of object `Input`. The Include relations are shown between protocol machines depicted as arcs with half-dashed ends.

A human interacts with the service (and with a protocol model) by submitting events. Each protocol machine has an alphabet of recognized events. The events recognized by protocol machines are specified as types. Each type is a data structure. Each instance of an event type contains own values of specified types.

For example, each instance of event `Insert` contains own identifier `Input:Input` and `Credit Card Number: Integer` (Figure 2). All three machines are synchronously instantiated accepting event `Instantiate`. Generic `Finalize` is an alias of events `Insert` and `Cancel`.

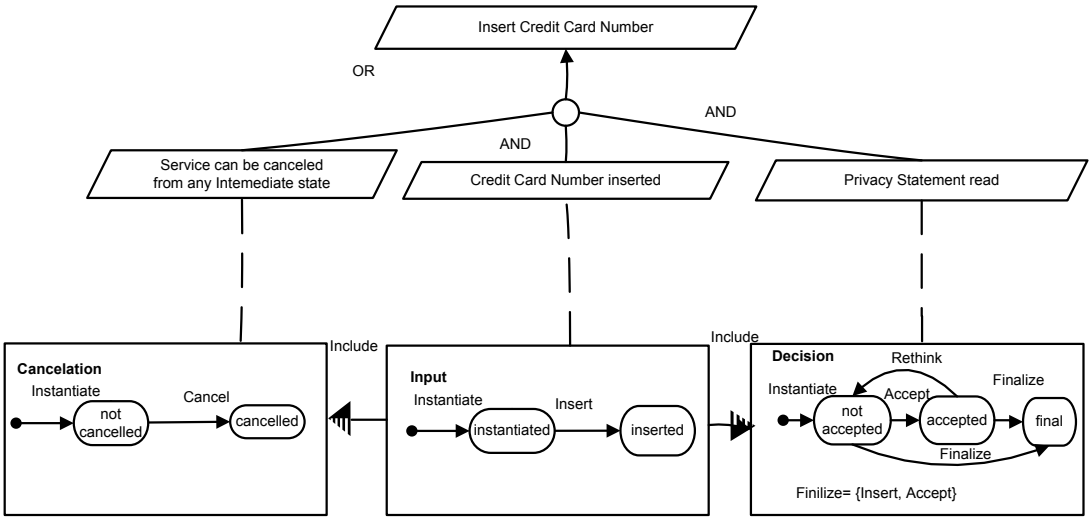
The generic interface generated from the model by the Modelscope tool shows to the user the state and the possible events at any execution step. The advantage of using Protocol Machine with goal models is that we result in an executable model of identified objects, agents and entities and can test achievement of chosen goals. We are able to identify the goal states. Not all scenarios of system behaviour lead to the goal states. Protocol machines present all life cycle scenarios of objects, agens and entities. The execution of the protocol models allows for testing realizability of goals and completing the incomplete or imprecise requirements.

Similar to a state machine, a protocol machine has a set of states and the local storage presented with attributes. However, the semantics of a protocol machine is different.

- A transition label of a state machine presents the pre-condition and the post-condition for enabling event to run to completion. A transition from state s_1 to state s_2 is labeled by

  $(s_1, [precondition]\, event\, /\, [postcondition], s_2)$ (OMG, 2003).

  The label shows that the transition in a state takes place only if the pre-condition is satisfied. If the pre-condition is not satisfied, the behaviour is defined by the semantic rules. Namely, the event is kept in a queue and waits for a state change to fire the transition.
- A transition label of a protocol machine presents an *event* that causes this transition. The storage




```
1   MODEL InsertCreditCardNumber
2   OBJECT Input
3       NAME Session
4       INCLUDES Decision, Cancelation
5           ATTRIBUTES Session: String, Card Number: Integer
6       STATES instantiated,inserted
7       TRANSITIONS @new*Instantiate=instantiated,
8                   instantiated*Insert=inserted
9
10  BEHAVIOUR Decision
11      STATES instantiated ,not accepted, accepted, final
12      TRANSITIONS @new*Instantiate=not accepted,
13                  not accepted*Accept=accepted,
14                  accepted*Rethink=not accepted,
15                  accepted*Finalize=final,
16                  not accepted*Finalize=final
17  BEHAVIOUR Cancelation
18      STATES not cancelled, cancelled
19      TRANSITIONS @new*Instantiate=not cancelled,
20                  not cancelled*Cancel=cancelled,
21
22  EVENT Instantiate
23      ATTRIBUTES Input:Input, Session:String,
24  EVENT Insert
25      ATTRIBUTES Input: Input, Credit Card Number: Integer,
26  EVENT Accept
27      ATTRIBUTES  Input:Input,
28  EVENT Rethink
29      ATTRIBUTES  Input:Input,
30  EVENT Cancel
31      ATTRIBUTES  Input:Input,
32  GENERIC Finalize
33      MATCHES Insert, Cancel
34
```

Figure 2: Goal Model and Can-Update Protocol Model

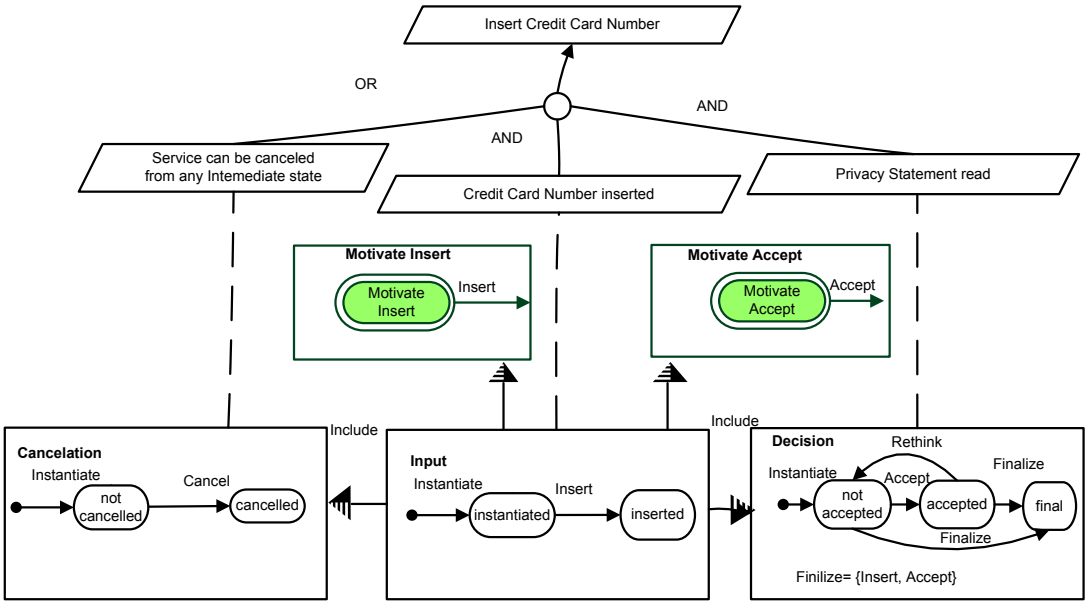


Figure 3: Goals, Protocols and Motivation

information is localized in the state. Being in a quiescent state in which the protocol machine can accept the submitted event, the protocol machine accepts one event at a time and handles it until another quiescent state. If the protocol machine cannot accept the event in its current state, the event is *refused* (McNeile and Simons, 2006; McNeile and Roubtsova, 2009).

The default type of protocol machines is `ESSENTIAL`. Essential protocol machines are composed (synchronized) using the CSP parallel composition and these machines are used to present the can-update-model, the business process.

The CSP parallel composition means that a Protocol Model accepts an event if all the protocol machines recognizing this event accept it. Otherwise the event is refused.

A protocol model accepts one event at a time and do not accepts any other event until it achieves the quitrent state. The results of this semantics are two important distinct properties:

- the state of a protocol model at any moment is a composition of state of protocol machines;
- the behaviour of any protocol machine is preserved in the whole protocol model and it is possible to reason locally on protocol machines about behaviour of the whole model.

5.2 Semantic Elements of Protocol Modelling for Motivation Modelling

There are some other semantic properties of Protocol Modelling for modelling of objectives and separation them from the can-update-model.

1. Ability of protocol machines to read but not modify the state of other protocol machines and to have an associated state function. This property makes it possible to build protocol machines with derived states. A *derived state* is a state that is calculated from the states of other machines using the state function associated with the protocol machine.

2. Different types of protocol machines are used to changes the use of CSP composition. The protocol machines of type `ESSENTIAL` are composed (synchronized) using the CSP parallel composition technique and these machines are used to present the can-update-model of the business process. The protocol machines of type `DESIRED` are not composed using the CSP parallel composition technique. These machines can be used to model the wanted behaviour or motivation.

5.3 BMM elements in Protocol Models

The elements of the BMM can be mapped onto protocol Models.

```
34
35   BEHAVIOUR !Motivate Insert
36   TYPE DESIRED
37       STATES motivate insert, other
38       TRANSITIONS motivate insert*Insert=@any
39
40   BEHAVIOUR !Motivate Accept
41   TYPE DESIRED
42       STATES motivate accept, other
43       TRANSITIONS motivate accept*Accept=@any
44
```

```
1  package InsertCreditCardNumber;
2
3  import com.metamaxim.modelscope.callbacks.*;
4
5
6  public class MotivateInsert extends Behaviour {
7
8      public String getState() {
9
10
11         String y=this.getState("Input");
12         String x=this.getState("Decision");
13              if (y.equals("instantiated")
14              || x.equals("accepted")
15               )  return "motivate insert";
16              else return "other";
17              }
18
19 }
20
```

```
1   package InsertCreditCardNumber;
2
3   import com.metamaxim.modelscope.callbacks.*;
4
5
6   public class MotivateAccept extends Behaviour {
7
8       public String getState() {
9
10                String x=this.getState("Decision");
11              if (x.equals("not accepted")
12               )  return "motivate accept";
13              else return "other";
14              }
15
16
17  }
18
```

Figure 4: Motivation Model

Ends or objectives are achieved in particular goal states.

Means or *Strategies* of the *Business Motivation Model* are events and sequences of events. Events or sequences leading to some chosen goal states form the corresponding motivation model.

Influences are presented as Protocol Machines included into the model. The influences add extra behaviour or constraints.

If an `Influence` is in the model, this means that this `Influence` is `assessed` as important.

5.4 Motivation model

In this section we add motivation models to the can-update protocol model in order to give to the user of the model the indication of means leading to objectives.

A want-model cannot forbid any transition in the can-update-model and it does not participate in the event synchronization with the can-update-models. Therefore, the want-models are not composed using the CSP parallel composition and have type `DESIRED`.

The motivation models are depicted in Figure 3.

- Protocol machine `Motivate Insert` models motivation for the goal "to get the credit card number inserted".
- Protocol machine `Motivate Accept` models motivation for the goal " to get the privacy conditions read by the user".

Each of those protocol machines has a derived state and an arc labeled with an event. The arc leads to any state allowed by the can-update-model. This structure is presented in the metamodel. Behaviours presented by lines $35-43$ in Figure 4) contain transitions described the arcs as transitions with the final state `@any`.

Each behaviour is labeled with an exclamation mark. The exclamation mark shows to the Modelscope tool that there is a call-back java file with the name of the marked behaviour. Each call-back function (lines $1-20, 1-18$ in Figure 4) derives state of the motivation model from the state of the objects and behaviours of the can-update-model. For example the state `Motivate Insert` is derived if `Input` is in state `instantiated` or if the `Decision` is in the state `accepted` (lines 11-15 in class `MotivateInsert`).

5.5 Motivation model for composition of goals

If the goals are OR-composed then achieving any of the goals is the goal and both call-back functions shown in Figure 4 are valid.

Figure 5: Execution of the Protocol Model with Motivation Models

Motivation model of the AND-combination of goals should not direct to states where at least one of goals cannot be achieved. In our case, motivation of event `Insert` when the object `Input` is in the state `instantiated` leads to the state where the goal `to get the privacy condition accepted` will never be achieved. Event `Insert` should not be motivated in state `instantiated` and lines 11 and 13 should be deleted from the call-back

function in Figure 4. The motivation models will first motivate event `Accept` and then the event `Insert`. The execution steps of the protocol model with the AND combination of motivation models are shown in Figure 5. The green light is given to the motivated events.

6 DISCUSSION AND FUTURE WORK

This paper has shown the expressive means of Protocol Modelling allowing combination of business processes and motivation of objectives in one model. It is the synchronous composition semantics of Protocol Modelling and the ability to derive states makes the combination possible.

There are several ways to use motivation models built into protocol models:

- `Generating user interface elements.` Motivation models can be used to generate the elements of the user interface. The wanted event and elements of user interface corresponding to them can be made of different form, color and use another order. In the generic interface of the the Modelscope tool (McNeile and Simons, 2011), the wanted events are presented in green.
- `Reuse of models.` The motivation models open another ways of reuse of business process models for systems with different goals.
- `Analysis of consistency and adequate completeness of requirements.` Motivation of goals in models stimulates analysis of realizability of goals and identification of contradicted goals and requirements. The psychological studies show that people tend to think contextually. Execution of requirements presented in protocol machines is better understood by users than the result of the formal methods applied on operation models. Execution of requirements provides better chance for recognizing of inconstant or incomplete requirements.
- `Composition of business processes on the basis of matching motivation model.` Such an approach promises creating effective business processes. This is especially importance in the context of the electronic business where the motivation provided by human is gone and the motivation should be built in to services as an element of service intelligence.

In the future work we plan the projects aimed to investigate analysis of consistency and adequate completeness of requirements and composition of business processes on the basis of motivation models.

REFERENCES

Alsumait, A., Seffah, A., and Radhakrishnan, T. (2003). Use Case Maps: A Visual Notation for Scenario-Based Requirements. *10th International Conference on Human - Computer Interaction, http://wwwswt.informatik.uni-rostock.de/deutsch/Veranstaltungen/HCI2003/.*

BRG (2010). The Business Motivation Model. Business Governance in a Volatile World.

Dardenne, A., van Lamsweerde, A., and Fickas, S. (1993). Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50.

Hoare, C. (1985). *Communicating Sequential Processes*. Prentice-Hall International.

ITU (2008). Formal description techniques (FDT). User Requirements Notation Recommendation Z.151 (11/08). http://www.itu.int/rec/T-REC-Z.151-200811-I/en.

Letier, E., Kramer, J., Magee, J., and Uchitel, S. (2008). Deriving event-based transition systems from goal-oriented requirements models . *Automated Software Engineering archiveD*, 15(2):1–22.

McNeile, A. and Roubtsova, E. (2009). Composition Semantics for Executable and Evolvable Behavioural Modeling in MDA. *BM-MDA'09*, pages 1–8.

McNeile, A. and Simons, N. (2006). Protocol Modelling. A Modelling Approach that Supports Reusable Behavioural Abstractions. *Software and System Modeling*, 5(1):91–107.

McNeile, A. and Simons, N. (2011). http://www.metamaxim.com/.

Milner, R. (1980). *A Calculus of Communicating Systems*. volume 92 of Lecture Notes in Computer Science. Springer.

OMG (2003). *Unified Modeling Language: Superstructure version 2.1.1 formal/2007-02-03*.

OMG (2010). Business Motivation Model. Version 1.1.formal/2010-05-01.

Pohl, K. and Rupp, C. (2011). *Requirements Engineering Fundamentals*. Rocky Nook.

Van, H. T., van Lamsweerde, A., and Philippe Massonet, C. P. (2004). Goal-oriented requirements animation. In *RE*, pages 218–228.

Yu, E. (1995). Modelling Strategic Relationships for Process Reengineering. *Ph.D. Thesis. Dept. of Computer Science, University of Toronto*.

Yu, E., Liu, L., and Li, Y. (2001). Modelling Strategic Actor Relationships to Support Intellectual Property Management. *LNCS 2224 Spring Verlag. 20th International Conference on Conceptual Modeling Yokohama, Japan*, pages 164–178.