Motivation Modelling for Human-Service Interaction

Ella Roubtsova

Open University of the Netherlands
ella.roubtsova@ou.nl

Abstract. Web services are goal-oriented software systems and need to influence or motivate, favour or dis-favour particular behaviour of their communication parties: humans and other services.
This paper investigates modeling of motivation for human-service interaction. It shows why motivation needs a separate model different from the service process model, how to specify motivation and compose the motivation model with the service process model.
Depending on the goals, the same service process model may have different motivation models. We provide an example of a service model with different motivation models that stimulate different behaviour of humans interacting with the web service. We show that motivation modelling is yet another way of web service reuse.

Keywords: Motivation Model, Service, Process, Protocol Model

1 Introduction

In 2010, the Object Management Group and the Business Rules Group completed their work on the Business Motivation Model (BMM), Version 1.1 [13, 15]. The BMM provides a scheme or structure for developing, communicating, and managing business plans. The schema covers four related elements:
1. The *Ends* of a business plan. "Among the *Ends* are things the enterprise wishes to achieve, for example, *Goals* and *Objectives*" [15].
2. 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".
3. "The *Influences* that shape elements of a business plan".
4. "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 Business Motivation Model: developers of business plans, business modelers, and implementers of software tools and repositories". The Business Rule Group believes that "Eventually specifications such as the Business Process Modeling Notation (BPMN) together with the Business Motivation Model (BMM) should be merged into a single business-oriented modeling architecture, and implemented in integrated tool suites"[15].

In this paper we make a step in direction of relating BMM with business processes and show how business modelers can benefit from motivation modelling. The motivation modelling is especially important for the modern electronic business that covers any area of human life. Web services govern job application, purchasing orders, booking requests, testing and requesting official documents - the list of web services is endless. Depending on their goals, web services need to motivate their users to choose particular actions among all possible actions. If business services are provided by people, these people motivate actions of customers. The web services themselves favour the choices of their customers and therefore they should benefit from having well designed motivation models built into them. This motivation of users is a some sort of intelligence that we need to add to services. The first step to systematic use of this intelligence is propagating the business *Ends (Goals and Objectives)* to the business process.

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.

Goals are usually formulated as non-functional requirements. They are usually abstract. The goals can be even unrealisable. The *Objectives* corresponding to goals are specific and measurable. They show realisability of goals. The motivation modelling can be seen as transformation of *Goals* into the corresponding *Objectives* described using elements of business processes and as a way to estimate realisability of *Goals*.

This paper presents a model of Business Motivation in Business Processes. We show how the semantics of Protocol Modelling [10] allows for localizing the motivation model in the business processes.

The structure of the paper is the following. Section 2 discusses related work. In Section 3 we formally define a process and introduce a motivation model on a process. Section 4 shows how to propagate the business goals and combination of goals to the motivation model of the process. Section 5 discusses the advantages of our approach to motivation modelling. Section 6 concludes the paper and identifies future work.

2 Related Work

There are many approaches that try to relate goals and processes.

The User Requirements Notation (URN) [4] 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 [17], and Use Case Maps (UCM) [1], a scenario modelling notation. The GRL provides a notation for modelling goals and rationales, and strategic relationships among social actors [18]. 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 be used for code generation.

Letier at al. [5] derive event-based transition systems from goal-oriented requirements models. The goal-oriented models are defined in the well known declarative approach KAOS (Knowledge Acquisition in autOmated Specification) [2]. Goals are specified in Linear Temporal Logic and organized using the AND and OR refinement structures. 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.

Van at al [16] propose goal-oriented requirements animation. The modelling formalism is the UML State Diagrams 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 are synchronized 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 goal state machines 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 motivation models for the human-service interaction as they also deal with the events from the queues.

The problems of goal-oriented approaches are mostly caused by different semantics used by process modelling and goal modelling techniques. Letier at al [5] 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. [5] 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.

In this paper we present the protocol modelling semantics that uses synchronous composition of concurrent behaviours and can be used both for process modelling and motivation modelling.

3 Model of Motivation

3.1 Process with can-semantics

In order to relate motivation and business process, we need a model of a process, 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 [11]:
- $C \subseteq (A \times S)$ 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.
- 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:

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

3.2 Motivation Modelling

There are always states in the process where particular goals are achieved. Let us name them 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 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 [8] .

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.

3.3 Human-Computer Interaction

As want-models do not contribute to behaviour of the systems but motivate the human communication with the service, the most simple application of motivation models is the justified design of human-computer interface.

A service presents to a human the possibilities and wishes in form the can- and want-model. For example, two events can be submitted, but only one of them is wanted:

$$((a; s_P) \in C) \wedge ((b; s_P) \in C) \wedge ((a; s_P) \in W).$$

The human can choose any possible action, but the action indicated by the want-model leads in this step to achieving a goal of the service:

$$((a; s_P) \in C) \wedge ((a; s_P) \in W)$$

Having a chosen goal in mind it is possible to favour the paths leading to the goal states by indicating wanted actions in any state of the process.

3.4 Several Goals

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 [14].

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 motivation modelling.

3.5 Motivation Modelling in Workflows and State Machines

Conventional behaviour modelling techniques use only can-semantics and therefore they do not provide means for motivation modelling.

For example, if a process is presented as a workflow, as an activity diagram, then to specify a want-model extra means are needed to identify the wanted outgoing transitions in each state. For, example, we can colour the wanted transitions. The state of a workflow is a set of marked nodes, so the combinations of nodes have to be built to formulate a want-model. If several want-models should be presented, then an incomprehensible spaghetti of coloured sub-diagrams will cover paths of the workflow.

If a system is specified as a composition of communicating state machines then this model often contains states that cannot be described as composition of

states of composed state machines. Such states appear because the semantics of state machines includes queues to keep the events which were submitted to the system when the system was not able to accept them. Such events are waiting for acceptance and may affect the wanted transitions in any state.

4 Motivation Models in Protocol Modelling

The semantics of the Protocol Modelling approach [10] offers an easy and practical way to model motivation separately from the can-update-model of the process. The Modelscope tool [9] supporting Protocol Modelling enables execution of can-models with different motivation models.

A Protocol Model is a synchronous CSP parallel composition of protocol machines [10]. This composition has its roots in the algebra Communicating Sequential Processes (CSP) proposed by Hoare [3]. McNeile [10] extended this composition for machines with data. 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.

We will introduce the relevant semantics of Protocol Modelling on a simple example and show how one can-update-model can be used with different motivation models (want-models). We show how the motivation model justifies different human-computer interface for the same can-update-model.

4.1 Web service: Insert Credit Card Number Goals and Requirements

Our simple case study is an *Insert Credit Card Number* web service that can be seen in many electronic booking systems. 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 always be cancelled before inserting the credit card number.

In any set of requirements there are goals and other concerns. Figure 1 shows them in notation of the KAOS method. We recognize two goals for this service, namely,

- 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.

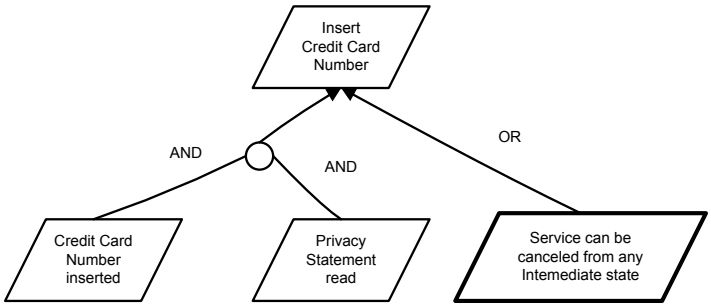

Fig. 1. Goal model of the web service: Insert Credit Card Number

4.2 Process model of the service

After the identification of the goals the KAOS suggests to identify objects, agents, entities and operations. Using Protocol Modelling we also fulfil this identification but we do this with the help of protocol machines. The advantage is that we result in an executable model of identified objects, agents and entities and can verify and even motivate achievement of chosen goals. Not all scenarios of system behaviour lead to goal states. There should be a can model that presents the life cycles of objects, agens and entities mentioned in the requirements.

We model the can-update process as a CSP composition of protocol machines `Input`, `Decision` and `Cancelation`. The graphical presentation of these protocol machines is in Figure 2. The executable Modelscope code is shown in Figure 3.

The protocol machine `Input` is an `OBJECT`, it has its identification name. The protocol machines `Decision` and `Cancelation` are `BEHAVIOURS`. They are included into each instance of object `Input`. This is shown as include relations between protocol machines depicted as arcs with half-dashed ends.

A human interacts with the service 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`. All three machines are synchronously instantiated accepting event `Instantiate`. Generic `Finalize` is an alias of events `Insert` and `Cancel`.

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) [12].

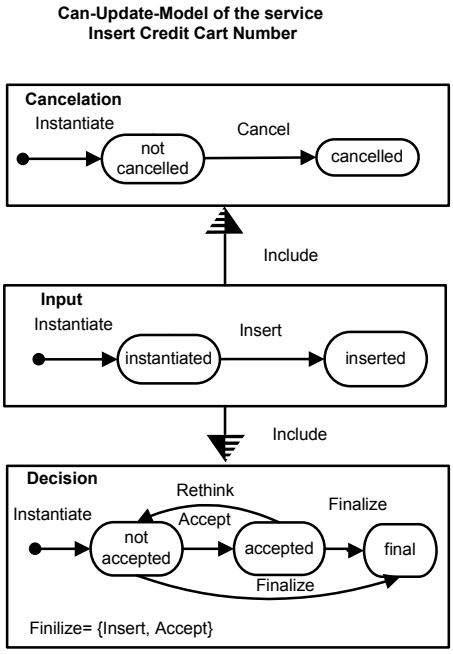

Fig. 2. Can-Update-Model: Insert Credit Card Number

```
MODEL InsertCreditCardNumber
OBJECT Input
    NAME Session
    INCLUDES Decision, Cancelation
        ATTRIBUTES Session: String, Card Number: Integer
    STATES instantiated,inserted
    TRANSITIONS @new*Instantiate=instantiated,
                instantiated*Insert=inserted

BEHAVIOUR Decision
    STATES instantiated ,not accepted, accepted, final
    TRANSITIONS @new*Instantiate=not accepted,
                not accepted*Accept=accepted,
                accepted*Rethink=not accepted,
                accepted*Finalize=final,
                not accepted*Finalize=final
BEHAVIOUR Cancelation
    STATES not cancelled, cancelled
    TRANSITIONS @new*Instantiate=not cancelled,
                not cancelled*Cancel=cancelled,

EVENT Instantiate
    ATTRIBUTES Input:Input, Session:String,
EVENT Insert
    ATTRIBUTES Input: Input, Credit Card Number: Integer,
EVENT Accept
    ATTRIBUTES  Input:Input,
EVENT Rethink
    ATTRIBUTES  Input:Input,
EVENT Cancel
    ATTRIBUTES  Input:Input,
GENERIC Finalize
    MATCHES Insert, Cancel

```

Fig. 3. Meta code of the Can-Update-Model: Insert Credit Card Number

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 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* [10, 7].

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.

4.3 Protocol Machines of Motivation Models

There are some semantic properties of Protocol Modelling that allow for localization of motivation modelling and separation it from the Can-Update-model.

1. Thanks to the abilities of protocol machines to read but not modify the state of other protocol machines and to have an associated state function, it is 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. Thanks to different types of protocol machines, the use of composition can be changed.

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.

3. It is also important that the refusal of events arriving, when the system is not able to accept them, guarantees that any state of a protocol model is always described as a composition of states of a final subset of composed protocol machines.

According to the definition given in section 3.2, a state of a want-model is related to some states of the process. Therefore, a want-model is presented as a protocol machine that does not have stored states but only derived states.

Want-models do not define new events, they indicate some events accepted by the corresponding can-update-models as wanted. A want-model cannot forbid any transition in the can-update-model and 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`. Summarizing, the semantic of Protocol Modelling provides expressive means to specify motivation.

```
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
```

Fig. 4. Motivate Accept.Motivate Insert.

Motivate Insert. Motivate Accept. The meta-code and the corresponding call-back functions in Figure 4 show how motivation models corresponding to each goal are modelled as protocol machines.

Protocol machines `Motivate Insert` and `Motivate Accept` have exclamation marks that show to the Modelscope tool that there are call-backs java files with the same names. Each call-back function derives state of the motivation model from the state of the objects and behaviours of the can-update-model.

For example, if the state of object `Input` is `instantiated` or the state of the behaviour `Decision` is `accepted` then state `motivate insert` is derived for protocol machine `Motivate Insert`.

The motivation models are depicted in Figure 5. The graphical presentation does not contain call-back functions.

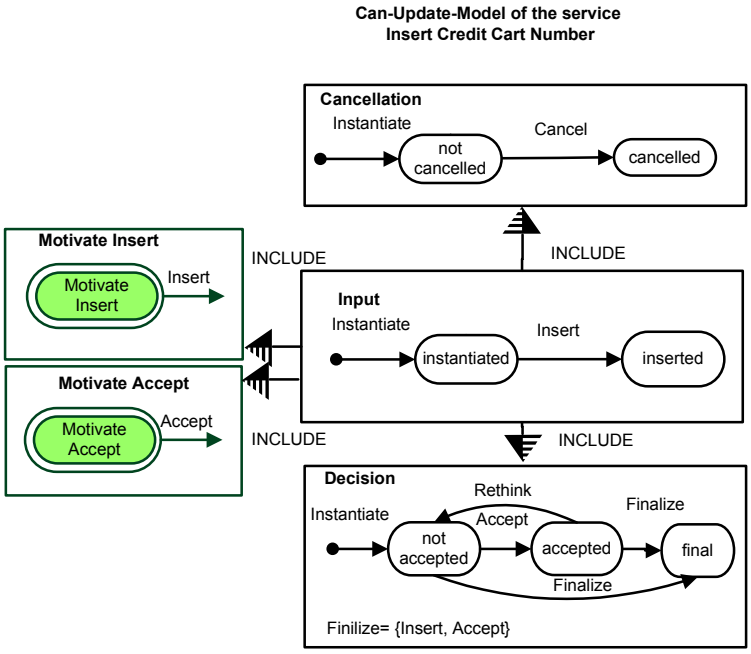

Fig. 5. Graphical Presentation of Motivation Models: Motivate Insert and Motivate Accept

4.4 Combination 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.

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. This state should be deleted from the call-back function `MotivateInsert` as shown in Figure 6.

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

Fig. 6. Call-back function Motivate Insert for AND-composition of goals.

5 Discussion

5.1 Motivation Models built into business process contribute to achievement of goals

It is known from the phycology studies that decisions of people are context-dependent. The human-computer interface may provide the context that leads to the choices that lead to goal states.

The want model can be transformed into human-computer interface of different sort: different visual elements, different colour or different position on the screen or another output device. In the generic interface of the the Modelscope tool, the wanted events are presented in green.

The visual elements of the human interface can be generated from the motivation model with the context related to the specified goals. The user of the system gets extra context information to choose the right action.

5.2 Motivation Models built into business process contribute to model reuse

Replacing one motivation model by another provides the reuse possibilities for services with different goals. This reuse possibility is actual for businesses looking for improvement of business processes built on web-services. Business models are changed very often with the changing market situation. Changing the motivation models gives to electronic business more flexibility.

5.3 Motivation Model built into business process relates the OMG BMM and BPM

Relating the OMG Business Motivation and Business Process Models serves to better understanding between managers making strategic decisions and requirement engineers preparing requirmetns for implementation.

In our approach the `Ends` of the Business Motivation model are presented as motivating protocol machines. The `Means` are events included into motivating protocol machines. they present the strategies. The `Influences` can be modelled as protocol machines. The choice of the objects included into the model as `Influences` is made on the basis of the `Assessments` about the impact of `Influences` relevant for the business process.

5.4 Scalability of Motivation Modelling for Protocol Models

A motivation model is defined by analysis of the states of a protocol machine that have transitions to the goal states and motivating actions leading to the goal states. These procedure can be iterated in order to motivate events on the traces leading to the goal states. As protocol models are abstract pieces of behaviour and usually do not contain more than $5 - 9$ states, constructing of motivation models does not require complex iterative procedures and any computer help.

If a motivation model depends on the state of several protocol machines then the ability to make abstractions with derived states allows joining the motivation protocol machines to any number of protocol machines presenting the business process in hand by writing a call-back function [6].

6 Conclusion and Future Work

This paper has presented an approach to motivation modelling. The approach is based on an extra binary relation included into the process model. This relation is used to identify the transitions in the process that lead to goal states.

The presented motivation model uses the semantics of Protocol Modelling which combines the synchronous composition and concurrency and therefore avoids the semantic mismatch between process modelling and goal modelling techniques, identified by Letier at al. [5]. Synchronous goal models can be rendered in protocol models. The motivation model relates the processes to system

goals and transforms them into objectives in terms of goal states in the business process of services. Objectives are specific and measurable and motivation models can make services more effective by motivating actions leading to the goal states. Reflecting the goals and objectives in the models is important for requirements engineering. New goals can challenge the business process. Such questions as, if the process model supports some needed `Means` or if an `End` is no longer relevant to the enterprise, are the elements of business process analysis [15].

The most interesting direction for future work is connecting web services on the basis of matching motivation models. Motivation model can be used to direct communication of collaborative services and to verify the realizability of service collaboration. Creating collaborative businesses from web services with matching motivation models promises the possibility of building more effective e-businesses.

Acknowledgement. The author thanks A.McNeile for sharing ideas and fruitful discussions related to the topic of the paper.

References

1. A. Alsumait, A. Seffah, and T. Radhakrishnan. 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/*, 2003.
2. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.
3. C. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.
4. ITU. Formal description techniques (FDT). User Requirements Notation Recommendation Z.151 (11/08). http://www.itu.int/rec/T-REC-Z.151-200811-I/en.
5. E. Letier, J. Kramer, J. Magee, and S. Uchitel. Deriving event-based transition systems from goal-oriented requirements models . *Automated Software Engineering archiveD*, 15(2):1–22, 2008.
6. A. McNeile and E. Roubtsova. CSP parallel composition of aspect models. *AOM'08*, pages 13–18, 2008.
7. A. McNeile and E. Roubtsova. Composition Semantics for Executable and Evolvable Behavioural Modeling in MDA. *BM-MDA'09*, pages 1–8, 2009.
8. A. McNeile and E. Roubtsova. Motivation and Guaranteed Completion in Workflow. *submittered to SOSYM*, 2011.
9. A. McNeile and N. Simons. http://www.metamaxim.com/.
10. A. McNeile and N. Simons. Protocol Modelling. A Modelling Approach that Supports Reusable Behavioural Abstractions. *Software and System Modeling*, 5(1):91–107, 2006.
11. R. Milner. *A Calculus of Communicating Systems.* volume 92 of Lecture Notes in Computer Science. Springer, 1980.
12. OMG. *Unified Modeling Language: Superstructure version 2.1.1 formal/2007-02-03.* 2003.
13. OMG. Business Motivation Model. Version 1.1.formal/2010-05-01. 2010.

14. K. Pohl and C. Rupp. *Requirements Engineering Fundamentals.* Rocky Nook, 2011.
15. The Business Rules Group. The Business Motivation Model. Business Governance in a Volatile World. 2010.
16. H. T. Van, A. van Lamsweerde, and C. P. Philippe Massonet. Goal-oriented requirements animation. In *RE*, pages 218–228, 2004.
17. E. Yu. Modelling Strategic Relationships for Process Reengineering. *Ph.D. Thesis. Dept. of Computer Science, University of Toronto*, 1995.
18. E. Yu, L. Liu, and Y. Li. Modelling Strategic Actor Relationships to Support Intellectual Property Management. *LNCS 2224 Spring Verlag. 20th International Conference on Conceptual Modeling Yokohama, Japan*, pages 164–178, 2001.