

Goal-Equivalent Secure Business Process Re-engineering ^{*}

Hugo A. López, Fabio Massacci, and Nicola Zannone

Università degli Studi di Trento, Via Sommarive, 14 I-38050 Povo (Trento), Italy
{lopez,massacci,zannone}@dit.unitn.it

Abstract. The introduction of information technologies in health care systems often requires to re-engineer the business processes used to deliver care. Obviously, the new and re-engineered processes are observationally different and thus we cannot use existing model-based techniques to argue that they are somehow “equivalent”. In this paper we propose a method for passing from SI*, a modeling language for capturing and modeling functional, security, and trust organizational and system requirements, to business process specifications and vice versa. In particular, starting from an old secure business process, we reconstruct the functional and security requirements at organizational level that such a business process was supposed to meet (including the trust relations that existed among the members of the organization). To ensure that the re-engineered business process meets the elicited requirements, we employ a notion of equivalence based on goal-equivalence. Basically, we verify if the execution of the business process, described in terms of the trace it generates, satisfies the organizational model. We motivate and illustrate the method with an e-health case study.

1 Introduction

It is common knowledge that the introduction of new information technologies (IT) into a system for Health care requires a large restructuring of the business process (BP for short). Such business process re-engineering (BPR) is an expensive task, normally assigned to external consultants, with the objective of “optimizing” the current organizational model in order to take advantage of the newly available technologies.

In the case of Health care this task is particularly challenging as we must guarantee that the “optimized” version is still optimal not just with respect to abstract efficiency notions but also against the primary goal of providing care and the secondary goal of protecting the privacy of the concerned individuals.

How do we know that the old BP and the new, re-engineered BP are somehow equivalent? Indeed the whole point of BPR is that the old and new processes should *not* be equivalent. Also from a security perspective the two processes are not equivalent. New actors or authorization profiles can be introduced in the IT system; old and effective “The doctor is out ask the chief nurse, as she is trusted” procedures might be disabled and so on.

^{*} This work has been partially funded by EU SENSORIA and SERENITY projects, MIUR-FIRB TOCAI project, and PAT MOSTRO project.

The alternative advocated in this paper is to look at the functional and security requirements in order to develop coherent models that understand *what* activities the system will do and which actor is entitled to perform them, rather than to specify *how* the system will implement such activities. In contrast BPs tell us the how. Finally, the knowledge gathered in BP specifications will be of utmost importance in defining which parts of the system can be distributed as services, implemented in later stages of the software development process.

So our idea can be sketched as follows;

- start from the old secure BP;
- reconstruct from that BP the functional and security requirements at organizational level that the old BP was supposed to meet (including the trust relations that existed among the members of the organization);
- compare the re-engineered BP with the requirements and see if they are equally met or possibly improved.

The requirements provide us the model of the BP process and the yardstick to compare the various re-engineered BPs.

One may object that we should rather always start from requirements when designing the system, but our practical experience of industrial cases is that this is never the case. After all this is BPR. The old BP is there, most likely was there since 5 or more years. Requirements of that time are equally likely to be lost or significantly evolved. One can see legacy systems as a perfect example of the importance of BPR. The old system will provide support to some aspects of the organization that has not changed their requirements, while new services will interact with the legacy system to cope with new requirements of the organization. At this point the issue becomes understanding whether or not a defined business process meets the business goals of an organization.

In this paper we present a reasoning method for passing from SI* [6], a modeling language that captures the functional, security and trust requirements of the socio-technical systems, to BPMN specifications and vice versa. Both translation processes are complementary, in the sense that requirements models can have multiple BP concretizations, and different business models can be equivalent in terms of the goals they achieve. We thus investigate the connection between BPs and requirements models, introducing the notion of *goal equivalence*. We apply this methodology to an e-health case study, finding important insights to their architecture and implementation.

This paper is organized as follows. In the next section we introduce a Smart Item system for Health Care that is used as a running example to explain the framework presented in this paper. Next, we present the general idea underlying the proposed framework (§3). We then present a trace model used to describe the system behavior (§4), a business process model (§5), and a requirements model (§6). These sections also discuss the relations among these models. Next, we introduce the notion of *goal equivalence* that allows one to compare business processes in terms of the goals they achieve (§7). Then, we present an approach for model driven re-engineering and verification (§8). Finally, we conclude the paper with final remarks and directions for future work (§9).

2 A Smart Item Infrastructure for Health Care

As the running example for explaining the framework proposed in this paper, we focus on the use of a smart items infrastructure in health care. In particular, we have considered a real-world case study where patients' health is monitored through a smart items infrastructure. This scenario has been studied in the context of the SERENITY Project¹ for the development and validation of Security and Dependability patterns. In this scenario a patient has been recently discharged from hospital after a cardiac arrest. Patient's health conditions need to be monitored 24 hours a day and so he has appointed the Health Care Centre (HCC), a provider of medical services. The HCC equipped the patient with a smart T-shirt that incorporates a motion sensor providing an alert as soon as he becomes passive for two minutes and monitoring devices that regularly measure his blood pressure, body temperature, etc. These data are communicated to his doctor via the Monitoring and Emergency Response Centre (MERC), a department of the HCC, that is responsible for receiving and handling patient's requests for assistance.

Among the possible situations that can be envisaged in this scenario, we focus on the faintness alert handling and delivery of the medicine scenes. In the first scene, the patient feels giddy and proceeds by sending a request for assistance to the MERC through his e-health terminal. To better understand the cause of the problem, this request is completed with patient's medical data getting automatically retrieved by patient's e-health terminal via a query to the smart T-shirt. The MERC first contacts the doctor of the patient. If he is not available, the MERC starts a doctor discovery process that consists in sending a broadcasting message to a group of doctors able to substitute that doctor. The appointed doctor interrogates the MERC to receive patient's medical data and medical history. The doctor analyzes patient information and decides for the most appropriate treatment. He writes the electronic prescription on his e-health terminal that promptly sends the prescription to the e-health terminal of the patient.

In the second scene, the patient feels weak and instead of driving to the pharmacy to get the medicine, he prefers to be supported by the MERC for this task. To this end, the MERC looks for an available social worker. The selected social worker receives a message from the MERC on his e-health terminal to go to the pharmacy and get the medicine to be delivered to the patient. He acknowledges the request and goes to the pharmacy. After a successful message exchange between patients' e-health terminal and the pharmacist's computer, the medicine is given to the social worker. After a successful message exchange between the social worker and the pharmacist, the medicine is given to the former that proceeds in delivering it to the patient.

3 The Approach

In this work, we propose a methodological approach that intends to assist requirement engineers and system designers in the development process. On the one hand, system designers are interested in analyzing the behavior of BPs and define a requirements model compliant with them. Oppositely, requirements engineers must verify whether

¹ EU-IST-IP 6th Framework Programme - SERENITY 27587 - <http://www.serenity-project.org>

or not a business process is a valid implementation of a requirements model. The proposed framework is divided in three conceptual steps, each of them at different levels of abstraction that provides concreteness to the high level models without forgetting the constraints introduced in more abstract levels.

1. The analysis of the requirements model, in terms of the actors involved in the system and their business goals.
2. The definition of BPs that implement business goals.
3. The analysis of the implementation.

Accordingly, we propose three modeling activities: (1) *Organizational Modeling*, (2) *Process Modeling*, and (3) *Execution Modeling*. In (1) the organizational context in which the system-to-be will eventually operate is captured. This activity intends to identify the actors participating to the business process along with their business goals and the relationships among them. Business goals are also refined in terms of the system functionalities necessary to achieve them. In (2) an implementation of the system is modeled in term of business processes. Finally, in (3) the system behavior is captured and analyzed. Figure 1 summarizes the different levels of analysis.

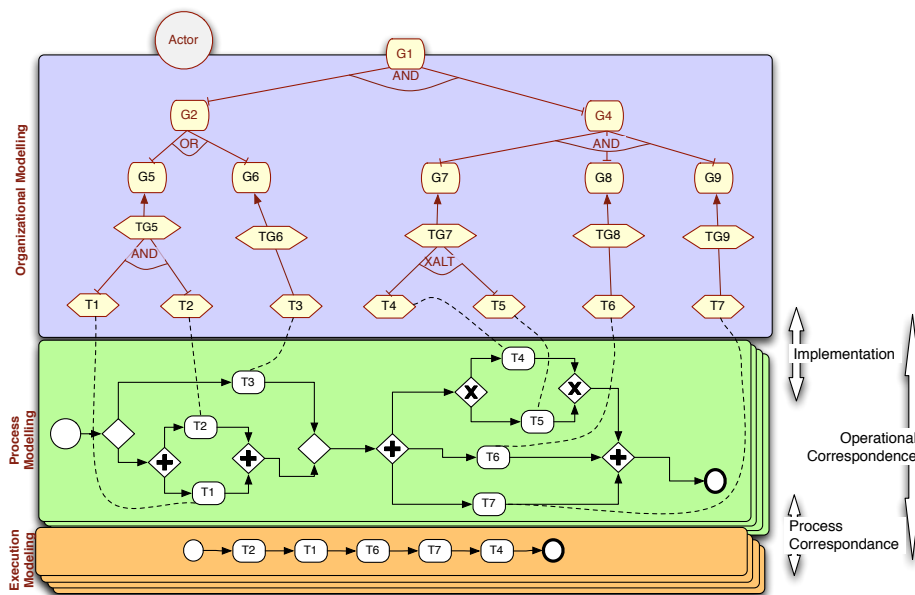


Fig. 1. Modeling at Different levels: Organizational Modeling (Blue), Process Modeling (Green) and execution modeling (Orange). The side arrows represent the model-driven transformations.

As the analysis goes down through the different layers, interesting questions arise about the consistency among the artifacts produced during the different levels of analysis. In particular, we are interested in analyze the consistency of requirements with a

business process as well as their consistency with the actual implementation of the system. To this end, we have considered and analyzed the following relations (Figure 1):

- *Implementation*: Alignment of organizational and process modeling.
- *Process Correspondence*: Alignment of process and execution modeling.
- *Operational Correspondence*: Alignment of organizational and execution modeling.

As the organizational model captures *what* activities must be done to achieve business goals, business process model captures *how* such activities are executed. The *implementation* of organizational models in terms of business process models is done using a design pattern approach that map activities and the relationships among them into business process specifications. This requires to analyze the organizational model in order to understand, for instance, the partial ordering among the activities to be performed [4]. It is worth noting that different business processes can be generated from an organizational model, as in (1) we identify which tasks should be executed, but not their ordering of execution.

The execution of a business process can be described in terms of the traces it generates. We are interested in the *process correspondence*, that is, if the business process specification is able to capture all possible behaviors of the system. To this end, we have analyzed the connections among modeling layers. We see tasks and actions as the common elements between different abstractions: in organizational modeling a task is the lowest level of abstraction that an objective can be refined, normally with an observable evidence present after the execution of the task. We use the same definition for process modeling, mapping the actions evidenced to the actions present in a program trace.

Last but not least, we want to guarantee the *operational correspondence* between the organizational model and the execution model. Essentially, we are interested in verifying that the traces capturing the behavior of the system satisfy the organizational model. This allows us to determine the correctness of the implementation with respect to the requirements model.

In summary, we have all the machinery to determine the traces that provide evidence about the satisfaction of business goals. Let $\llbracket \cdot \rrbracket : GM \rightarrow PM$ a mapping from organizational models to process models, and $\{\cdot\} : PM \rightarrow \mathcal{T}$ a mapping from process models to traces. An organizational model GM can be encoded in all the possible traces that satisfy the business goals, as can be seen below.

$$\frac{GM \rightarrow \llbracket PM \rrbracket \quad PM \rightarrow \{\mathcal{T}\}}{GM \rightarrow \bigcup_{T \in \llbracket OM \rrbracket} \{\mathcal{T}\}}$$

This allow one to identify whether or not agents have been satisfied their objectives.

4 Execution Modeling

This modeling phase intends to capture the system behavior. We use trace languages to capture the possible execution of the system. We can see a trace as a sequence of messages (actions) expressed by a given actor after the execution of an activity, as

```

Patient.Send(Request for Attention, MERC)
Patient.Send(Request for Data, Tshirt)
Tshirt.Receive(Request for Data, Patient)
Tshirt.Send(Data, Patient)
Patient.Receive(TshirtData, Tshirt)
MERC.Receive(Request for Attention, Patient)
MERC.ContactDoctor(Patient)
Doctor.Receive(Request, MERC)
Doctor.Send(⟨Acceptance, Patient⟩, MERC)
MERC.Send(Request for Data, Patient)
Patient.Send(PatientData, MERC)
MERC.Send(PatientData, Doctor)
Doctor.Receive(PatientData, MERC)
Doctor.DecideTreatment(Patient, PatientData)
Doctor.Send(Prescription, Patient)
Patient.Receive(Prescription, Doctor)

```

Fig. 2. A possible trace for completion of the faintness alert process

introduced by Hoare in CSP [3]. They provide the lowest level of abstraction possible, analyzing systems in terms of the partial-ordered traces they generate. Traces have been widely recognized as a comfortable model in which all the possible interactions of the models are captured [8]. In our models, an action can represent a message generated by an agent to the environment and a trace is simply a sequence of such actions. This approach has been widely used in the analysis of distributed systems, specially in security protocol analysis [2, 1, 5]. Actions can represent the execution of an activity, a failed execution, a delay or a message passing through a given channel.

In our e-health scenario, a possible trace can be a sequence of actions in which a faintness alert has been released. The patient releases a request for attention message, and the MERC contacts patient doctor who gives the appropriate prescription. The trace is illustrated in figure 2, where an action is denoted as *Agent.Action(Parameters)*, differencing the agents who produce the activities as well as the actions that produces the agent, the parameters will vary from the actions executed, being data private to the agent who executes it or information available on the environment.

5 Process Modeling

This modeling phase aims to describe the set of activities intended to reach business goals, that is, the business process. We represent BPs using Business Process Modeling Notation (BPMN) [9]. BPMN adopts the concepts of process, which is composed of a set of partial ordered activities, and participants (to the business process) and represent them in a Business Process Diagram (BPD). Essentially, a BPD is a collection of *agents*, *objects*, *sequence flows* and *message flows*. The sequence flow determines the order of execution between two different objects. Objects in BPD are decomposed by *tasks*, *events* or *gateways*. An event may signal the starting point of a process, its termination, arriving messages or a time-date being reached during the execution of

a process (intermediate event). A task stands for an atomic activity to be performed within the process. Finally, the set of gateways denote how a sequence of objects can converge/diverge into different sequences. In particular, the *parallel fork* gateway allows one to create concurrent sequence flows, and the *parallel join* gateway is used to synchronize concurrent flows. Gateways can also filter possible executions, for instance the behavior expressed with *XOR decision gateway* and its corresponding *XOR merge gateway*, that permits the execution of only one of the sequences received. Finally, *OR decision/OR merge* gateways controls the execution of at least one of the sequences in the flow. A *message flow* is used to show message exchanges between two participants. Figure 3 presents a core of BPMN notation and we refer to [9] for the full specification.

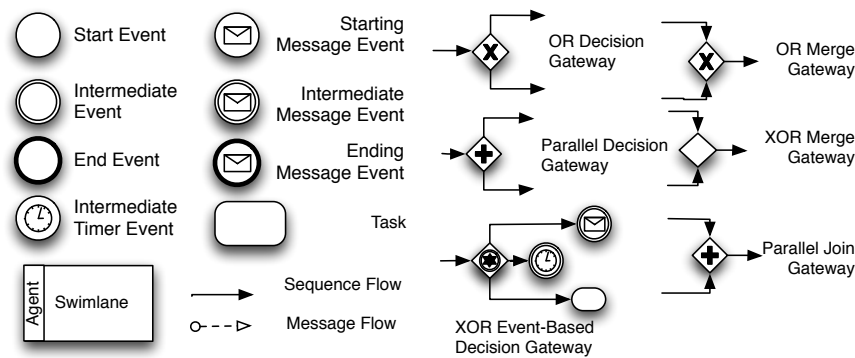


Fig. 3. A core subset of BPMN elements

A business process that captures a medicine delivery process to patient's house is presented in Figure 4. The specification can be seen as the interaction of five Agents (Patient, HCC, MERC, Social Worker and Pharmacist). Every agent acts in his own behalf, synchronizing their actions by means of message passing. Essentially, the patient starts his process by requesting help in the delivery of the medicine. This request is processed by the Health Care Center, selecting the closest MERC assigned to the ill patient and transmitting his request. Once the MERC receive the request for delivery, he starts a search in order to identify an available social worker. This activity will be executed until a social worker acknowledge the MERC with a message denoting his availability. After that the MERC transmits his credentials to the patient for further verification. The social worker will receive the medicine from the Pharmacist prior verification of his identity and validity of the prescription. Then, he will drive to patient's location, delivering the medicine only if he is identified by the patient.

Once the business process is defined, we are interested in verifying if such a process has been correctly implemented. To this end, we introduce the notion of *trace satisfaction* to determine if a particular execution trace can be generated by the business process. A trace T will satisfy a business process BP ($T \models_B BP$) if T describe the behavior of BP . We can analyze \models_B in an inductive manner. The case base is represented by single tasks for which the occurrence of the correspondent action in the

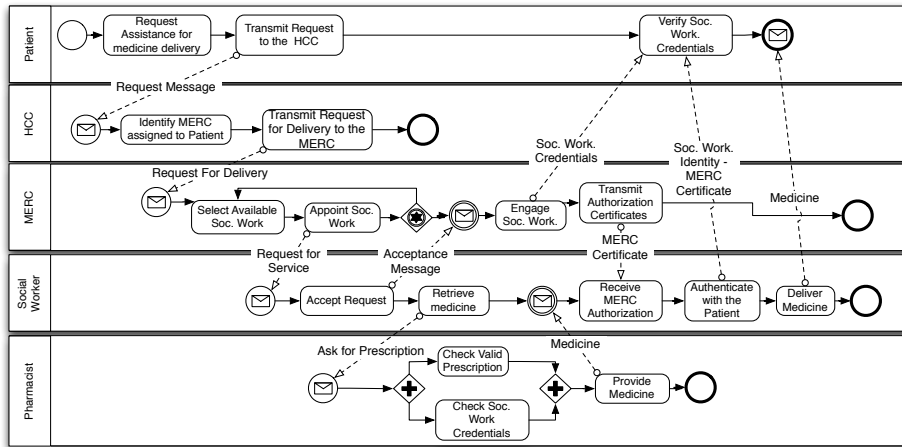


Fig. 4. Business Processes for the Delivery of Medicine

trace is verified. Inductively, composed business processes are split into sub-processes, achieving satisfaction if their sub-processes are satisfied and their composition respect the constraints imposed by the business process.

Other important aspect at this level is the ability of comparing two different business processes. We will use the trace satisfaction to state a similarity relation between business process, saying that two business processes BP_a and BP_b are similar if the set of traces that satisfy BP_a are the same traces that satisfy BP_b . A relaxed version of the property, named *weak B-similarity* allow us to identify business processes that contain the same set of execution traces, but where one can cover more cases than the other, as in the case where a business process is implemented with optional activities that can reduce to the same set of actions. Finally, we can borrow the concept of simulation [7] to compare whenever two different business process behave in the same way, capturing cases such as the implementation of multiple design choices with respect to single (and economical) process sequences, as can be seen on Figure 5.

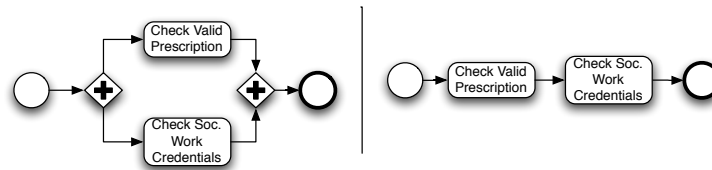


Fig. 5. Simulation between Business Processes: the BP on the right can be simulated with the BP on the left based on the traces they generate, but the case is impossible in the opposite direction as it accept more traces due to the interleaving concurrency of the traces.

6 Organizational Modeling

Organizational modeling addresses the understanding of *what* are the system functionalities and *why* they are necessary. To this intent, we have used SI* [6], a modeling language tailored to model secure socio-technical systems. In a nutshell, SI* adopts the concepts of actor, goal, task, and resource. An *actor* is an intentional entity that performs actions to achieve goals. A *goal* is a strategic interest of an actor. A *task* specifies a sequence of actions that can be executed to achieve a goal. Every actor is defined along with a set of *objectives*, *entitlements*, and *capabilities*, which represent what the actor wants, what he has the permission to do, and what he is able to do. SI* also adopts the notions of *execution dependency*, *permission delegation*, *trust of execution*, and *trust of permission* to model assignments of responsibilities and permission between two actors and the expectations of one actor about the performance and fair behavior of another actor, respectively.

From a methodological perspective, SI* rests on the idea of building a model of the system that is incrementally refined and extended. Specifically, goal analysis consists of refining goals and eliciting new social relationships among actors. It is conducted from the perspective of single actors using means-end analysis, AND/OR decomposition, and contribution analysis. Means-end analysis identifies the tasks that can be used to achieve a goal. AND/OR decomposition is used to define a finer goal structure. We have also extended the language by including X-ALT decomposition. This relation identifies refinements with mutually-exclusive alternatives. Essentially, the difference between OR and XALT operators lies at the compulsory character of the choice: while OR allows a free choice of multiple alternatives, XALT defines an obligatory selection of a choice. Finally, contribution analysis represents the impact that achievement of a goal has on other goals, in positive or negative ways. The set of elements of the diagrams are illustrated in Figure 6. Figure 7 shows the SI* specification of the organizational model corresponding to our e-health scenario.

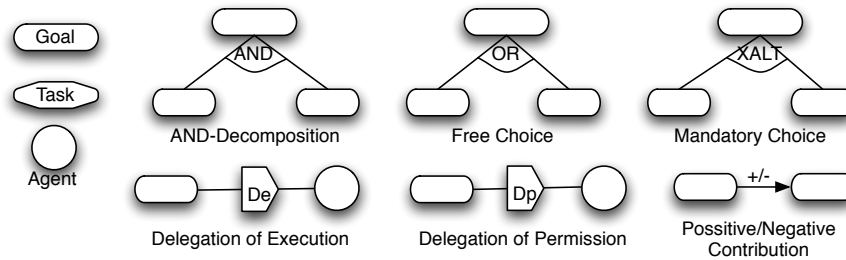


Fig. 6. Elements of Goal and Task Diagrams in the SI* language

We can see the satisfaction of an organizational model in two different layers. Firstly, as organizational model consists of diagrams of goals that are decomposed, we need to identify which tasks satisfy a given goal. This idea is captured by defining a notion of trace-satisfaction over organizational models. Essentially, a trace \mathcal{T}_a

We say that a BP *implements* the goals of an organizational model if the set of traces that satisfy the BP are included in the set of traces captured by the organizational model. Note that at this level, it is not feasible to state the inverse relation, as the implementation of an organizational model with a BP implies the selection of design alternatives, still complying with the fulfillment of the business goals. In our health care scenario, this case can be exemplified in an arbitrary implementation of one of the sequences accepted by the parallel specification of tasks on the pharmacist work.

We can define a correspondence criteria between organizational models and BPs. An organizational model *corresponds* to a BP if all possible traces captured by the goals in the organizational model are also captured in the set of traces that implements the BP. This is a stronger counterpart of the implementation property, in the sense that correspondence will need an implementation for every possible goal in the diagram that defines the organizational model. Returning to our example, there is no way that an arbitrary implementation of a partial set of sequences generated by the parallel decomposition can correspond to the organizational model of the system, we will need to include the parallel decision gateway to describe the different executions of the system or we will not correspond to the organizational model.

Finally, we can say that two BPs are *Goal - Equivalent* if their correspondence with respect to the goals expressed in an organizational model are the same, even if they are not trace equivalent.

8 Model - Driven Verification

In MDA we aim a transformation of different models retaining consistency between the source model and the target, in terms of the constraints imposed by the operators used. The approach presented in this paper shows how Business Processes and Organizational Models have a clear relation between each other. The verification phase aims to guarantee that the transformations between models actually represents the same cases, in terms of the traces they capture.

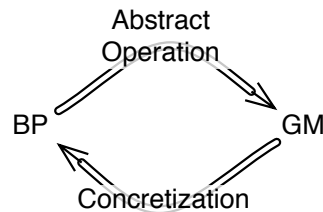


Fig. 8. Model-Driven Verification: Relations between Organizational Models and BPs

Figure 8 illustrates our approach in a loop process where $\text{Abstract} \subseteq GM \times BP$ is a function that translates a business process model into its corresponding organizational structure, and $\text{Concretise} \subseteq BP \times GM$ is a function that concretise an organizational model into a business process specification. We need to verify that:

- For an abstraction operation, for all the traces \mathcal{T} , if $\mathcal{T} \models_B \text{BP}$ then $\mathcal{T} \models_G \text{Abstract}(\text{GM})$.
- For a concretizations operation, for all the traces \mathcal{T} , if $\mathcal{T} \models_G \text{GM}$, then for all business process BP_i derived from $\text{Concretise}(\text{GM})$, $\mathcal{T} \models_G \text{BP}_i$.

If both properties hold, we can guarantee soundness and completeness of the transformation process, in terms that all the information analyzed on an Organizational Model is preserved in Business Processes and vice versa.

9 Conclusions

In this work we have proposed a methodology for re-engineering of business processes in terms of requirements analysis, with a set of verification steps that allow us to identify behavioral relations between business processes and the goals they are aiming to achieve. Moreover, a set of criteria is proposed in order to identify whenever two business processes are equivalent to the requirement models they implement, allowing designers to validate different approaches without restricting themselves to implement the same activities on the business process. Finally, an sketch of the model driven architecture is proposed, defining the validation steps that should be followed in order to find correct business process transformations.

References

1. Federico Crazzolaro. *Language, Semantics, and Methods for Security Protocols*. Doctoral dissertation, BRICS, daimi, May 2003. PhD thesis. xii+160.
2. F. Javier THAYER Fabrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. In *Journal of Computer Security*, volume 7, pages 191–230, 1999.
3. C. A. R. Hoare. Communicating Sequential Processes. *Commun. ACM*, 26(1):100–106, 1983.
4. H. J. Johansson, P. McHugh, A. J. Pendlebury, and W. A. Wheeler. *Business Process Reengineering–Breakpoint Strategies for Market Dominance*. John Wiley & Sons, 1993.
5. Gavin Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1–2):53–84, 1998.
6. Fabio Massacci, John Mylopoulos, and Nicola Zannone. An Ontology for Secure Socio-Technical Systems. In *Handbook of Ontologies for Business Interaction*. The IDEA Group, 2007.
7. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, parts I and II. *Journal of Information and Computation*, 100:1–77, September 1992.
8. R. J. van Glabbeek. The linear time-branching time spectrum. In *Proceedings of the Theories of Concurrency: Unification and Extension*, pages 278–297, 1990.
9. S.A. White. Business Process Modeling Notation (BPMN) Version 1.0. *Business Process Management Initiative, BPML.org, May, 2004*.