Goal-Equivalent Secure Business Process Re-engineering for E-Health[*]

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 ITs in e-Health 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 notion of equivalence over secure business processes based on the notion of goal-equivalence:

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

To this intent, we present a reasoning method for passing from SI*, a modeling language that captures the functional, security and trust requirements of IT systems and their operational environments, to business processes specifications and vice versa. Both translation processes are complementary, in the sense that SI* models can have multiple business process concretizations, and different business processes can be equivalent in terms of the goals they achieve. We illustrate and motivate the proposed approach using an e-health case study.

1 Introduction

It is common knowledge that when new ITs are introduced into a system, a large restructuring of the business process (BP for short) is needed. 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 central point of BPR is that the old and new processes should

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

not be equivalent. Also from a security perspective the two processes may not be equivalent. New actors or authorization profiles can be introduced in the IT systems, old and effective "The doctor is out, ask the chief nurse, as she is trusted" procedures might be disabled and so on.

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 those goals [15]. In contrast, BPs tell us how, albeit at a high-level, we will perform a number of tasks.

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.

At this point the issue becomes understanding whether or not a defined business process meets the business goals of an organization. This issue has been partially addressed in both Requirements Engineering (RE) and Business Analysis (BA) research areas, but separately [13, 16]. Some requirements engineering methodologies have already addressed the definition of business processes [3, 7, 10], but most of them unfortunately result to be inadequate to describe complex business processes, especially when spanning across different organizations [2, 9]. Albeit of the efforts in this direction in the area, the task is still very challenging [2].

In this paper we present a reasoning method for passing from SI* [12], a modeling language tailored to capture and model functional, security and trust requirements of socio-technical systems, to BPMN specifications and vice versa. Both translation processes are complementary, in the sense that SI* models can have multiple business process concretizations, and different business models can be equivalent in terms of the goals they achieve. We also investigate the connection between business processes and requirements models, introducing the notion of *goal equivalence* that allows one to compare business processes in terms of the goals they achieve. We apply the framework to an e-health case study, finding important insights about their architecture and their 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 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 (§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. The exploitation of smart items infrastructures offers significant benefits in many situations ranging from regular monitoring of patients' conditions after hospitalization to emergency cases, where the life of patients is at risk. One for all, access and integration of all available health care resources offering a continuous, widespread, cooperative health care system and tools for personalized patient monitoring.

We have considered a real-world case study showing how health care monitoring of patients after hospitalization might be managed in the near future through a smart items infrastructure. This scenario has been studied in the context of the SERENITY Project[1] 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 for this purpose 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 heart rate, 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, which is responsible for receiving and handling patient 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 that are 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

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

the pharmacy. 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. The architecture of the system is exemplified in Figure 1.

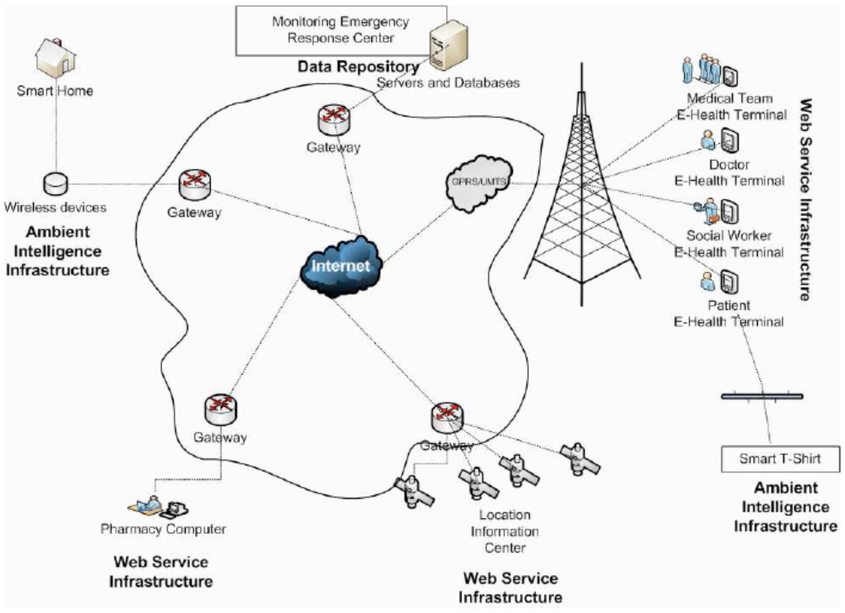
Monitoring Emergency Response Center
Data Repository
Servers and Databases
Smart Home
Wireless devices
Ambient Intelligence Infrastructure
Gateway
Gateway
GPRS/UMTS
Internet
Medical Team E-Health Terminal
Doctor E-Health Terminal
Social Worker E-Health Terminal
Patient E-Health Terminal
Web Service Infrastructure
Gateway
Gateway
Smart T-Shirt
Ambient Intelligence Infrastructure
Pharmacy Computer
Web Service Infrastructure
Location Information Center
Web Service Infrastructure

Fig. 1. Logical Architecture of the Smart Items Scenario [1]

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 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 a different level 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 2 summarizes the different levels of analysis.

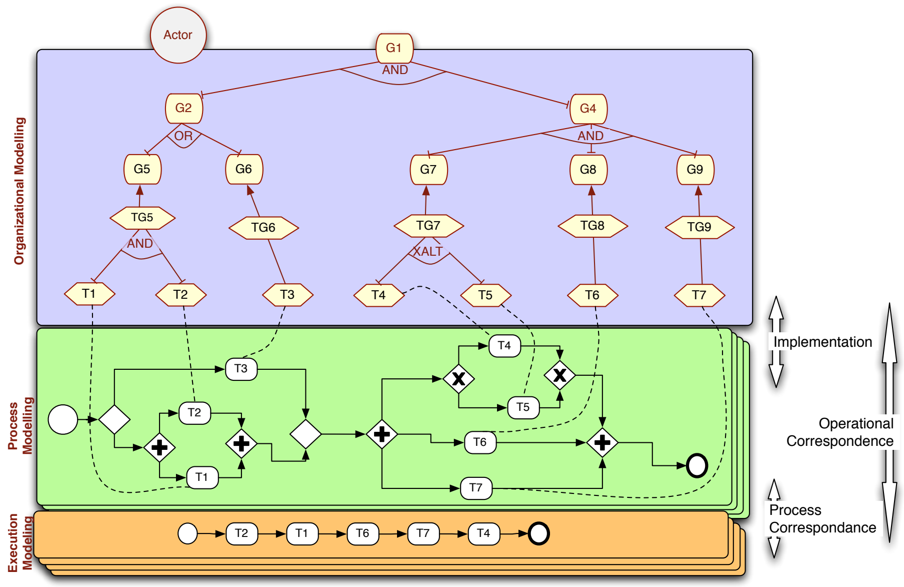

Fig. 2. Modeling at Different levels: Organizational Modeling (Blue), Process Modeling (Green) and execution modeling (Orange). The side arrows represent the model-driven transformations.

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 analyzing 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 2):

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

As the organizational model captures *what* activities must be done to achieve business goals, a business process captures *how* such activities are executed. The *implemen-*

tation of organizational models in terms of business processes 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 [8]. 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. This allows us to verify the *process correspondence*, that is, if the business process specification is able to capture all possible behaviors of the system.

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 whether or not the traces describing the behavior of the system satisfy the organizational model. This allows us to determine the correctness of the system 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 $[\![\cdot]\!] : OM \to PM$ a mapping from organizational models to process models, and $\{[\cdot]\} : PM \to \mathcal{T}$ a mapping from process models to traces. An organizational model OM can be encoded in all the possible traces that satisfy the business goals, as shown below.

$$\frac{OM \to [\![PM]\!] \qquad PM \to \{[\mathcal{T}]\}}{OM \to \bigcup_{\mathcal{T} \in [\![OM]\!]} \{[\mathcal{T}]\}}$$

This correspondence allows one to establish whether or not an organization can satisfy its business 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 introduced by Hoare in CSP [6]. 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 [17].In our models, an action can represent a message generated by an agent to the environment and a trace is simple a sequence of such actions, such as the log messages of a process. This approach has been widely used in the analysis of distributed systems, especially in security protocol analysis [5, 4, 11]. Actions can represent the execution of an activity, a failed execution, a delay or a message passing through a given channel.

As an example in the 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 3, 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.

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

Fig. 3. A possible trace for completion of the faintness alert process

5 Process Modeling

This modeling phase aims to describe the set of partially ordered activities intended to reach business goals, that is, the business process. We represents business processes using Business Process Modeling Notation (BPMN) [18], which is emerging as the defacto standard notation for modeling executable business processes. BPMN adopts the concepts of process, which is composed of a set of partial ordered activities, and participant (to the business process), and represents 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. On the contrary, *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 that are prepared to send and receive them. These messages can involve tasks or events, each of them on different agents. These primitives, available in BPMN specification standard [18] faithfully represents the *behavioral* aspects of a business process, as seen in Figure 4. Interested readers can refer to [18] for the full BPMN specification.

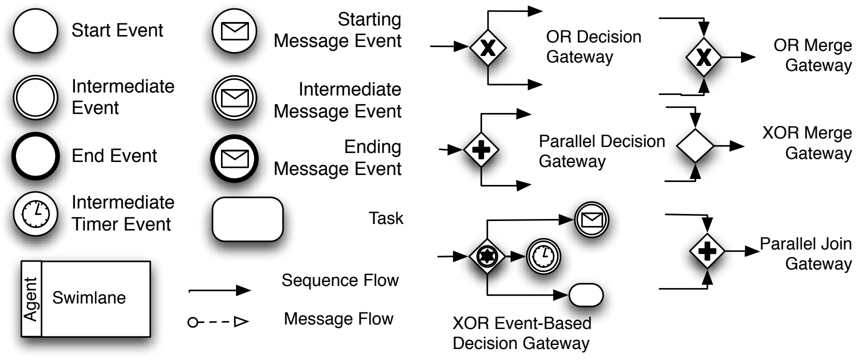

Fig. 4. A core subset of BPMN elements

A business process that captures a medicine delivery process to patient's house is presented in Figure 5. The specification can be seen as the interaction of five agents (i.e., 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 acknowledges 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 $\mathcal{T}$ will satisfy a business process BP ($\mathcal{T} \models_B BP$) if $\mathcal{T}$ contains all the necessary actions that requires a BP. We can analyze $\models_B$ in an inductive manner. The case base is represented by single tasks for which the occurrence of the correspon-

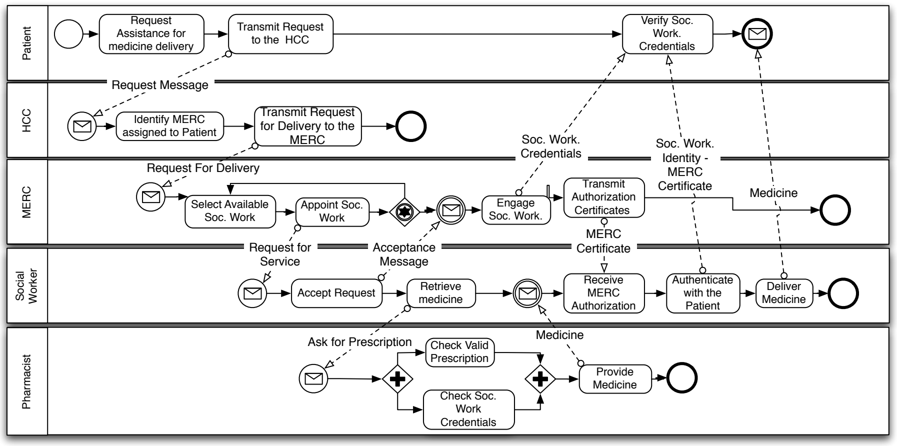
Patient
Request Assistance for medicine delivery
Transmit Request to the HCC
Verify Soc. Work. Credentials
Request Message
HCC
Identify MERC assigned to Patient
Transmit Request for Delivery to the MERC
Soc. Work. Credentials
Soc. Work. Identity - MERC Certificate
Request For Delivery
MERC
Select Available Soc. Work
Appoint Soc. Work
Engage Soc. Work.
Transmit Authorization Certificates
Medicine
Request for Service
Acceptance Message
MERC Certificate
Social Worker
Accept Request
Retrieve medicine
Receive MERC Authorization
Authenticate with the Patient
Deliver Medicine
Ask for Prescription
Medicine
Pharmacist
Check Valid Prescription
Check Soc. Work Credentials
Provide Medicine

Fig. 5. Business Processes for the Delivery of Medicine

dent action in the trace is verified. Inductively, composed business processes (i.e., they are structured using the information flow, such as parallel composition, exclusive alternatives, design choices, etc.) are split into sub-processes, achieving satisfaction if their sub-processes are satisfied and their composition respect the constraints imposed by the business process.

Another 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* $B\text{-}similarity$ 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 [14] to verify if two different business processes 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 6.

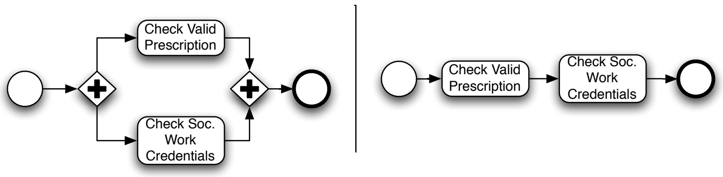

Fig. 6. Simulation between Business Processes: the business process on the right can be simulated by the business process on the left on the basis on the traces they generate, but not vice versa

6 Organizational Modeling

The organizational modeling addresses the capture of the requirements of the system. The basic idea of this phase is to understand *what* are the system functionalities and *why* they are necessary.

To this intent, we have used SI* [12], 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, which 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. In particular, AND-decomposition represents the high-level process for achieving a goal, OR decomposition identifies the alternative ways in which a goal can be achieved. We have also extended the language by including XALT 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 7. Figure 8 shows the SI* model of our e-health scenario.

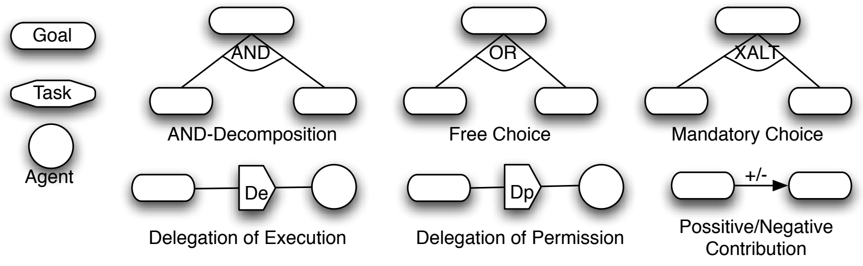

Fig. 7. Elements of Organizational Models in the SI* language

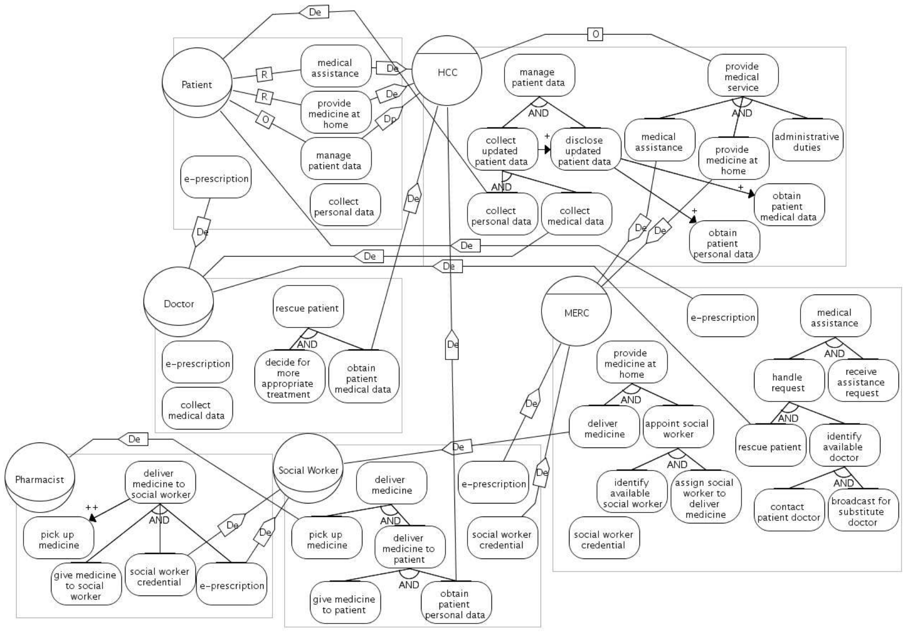

Fig. 8. Organizational Model of the E-Health Scenario

The translation from the organizational model to the execution model will capture the "Domain of the system", that is the set of possible traces that satisfy an organizational model.

We see the satisfaction of an organizational model in two different layers. Firstly, as an organizational model consists 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 $\mathcal{T}_a$ $G\text{-}Satisfy$ an organizational model OM ($\mathcal{T}_a \models_G OM$) if $\mathcal{T}_a$ contains evidence of the actions executed by the lowest level tasks of the organizational model in an order that respects the decomposition operators of OM.

Secondly, we need to identify whenever an organizational model is satisfied by a set of traces. Let us remind that an organizational model is a diagram that refines the objectives of an actor in lower levels of abstraction. As objectives are decomposed the evidence of achievement is only captured by the lowest level tasks, so it is necessary to establish a criterion to analyze if a decomposed goal have been or not achieved. The satisfaction of a decomposition is presented with the aim of capturing this notion. Basically, an organizational model OM_j will satisfy by decomposition OM_i ($OM_j \models_D OM_i$) if OM_i have been decomposed into OM_j and exists at least a trace $\mathcal{T}$ such that $\mathcal{T} \models_G OM_j$ satisfy the constraints imposed by the sequence of decompositions between OM_i and OM_j.

7 Goal Equivalences

So far we have studied both the relations between requirements expressed in Organizational models and Business Processes in terms of the traces they execute. However, what we want is to analyze how a business process "implements" an organizational model, or in other words, we want to identify which goals are captured by a business process.

We say that a Business Process *implements* the goals of an Organizational Model if the set of traces that satisfy the business process 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 Business Process can include the cancellation of alternatives by decision of the designer, still complying with the fulfillment of the business goals. In our e-Health 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 Business Processes. An Organizational Model *corresponds* to a Business Process if all possible traces captured by the goals in the Organizational Model are also captured in the set of traces that implements a Business Process. This is an 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 the 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 support our analysis over the correspondence property to define equivalences between goals on the business processes. We can say that two business processes 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. This concept allow us to identify correct process of reingeneering, where no objectives have been forgotten in the transformation process.

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 different process views are consistent in terms of the traces they capture.

Figure 9 illustrates our approach in a loop process where `Abstract` $\subseteq OM \times BP$ is a function that translates a business process model into its corresponding organizational structure, and `Concretise` $\subseteq BP \times OM$ is a function that concretizes an organizational model into a business process specification. We need to verify that:

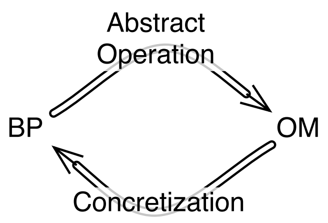

Fig. 9. Model-Driven Verification: Relations between Organizational Models and Business Processes.

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

If both properties hold, we can guarantee soundness and completeness of the transformation process, in terms that all the information analyzed on organizational models are preserved by 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, a 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. A. Benameur, R. Bonato, L. Compagna, V. Di Giacomo, D. Donnan, P. El Khoury, D. Gidoin, L. Gomez, K. Hajy, S. Holtmanns, J. Latanicki, C. Pandolfo, J-C. Pazzaglia, M. Seguran, and N. Zannone. Specification of SERENITY Architecture. Research report A7.D3.1, SERENITY consortium, 2007.
2. J. Bleistein, Karl Cox, June Verner, and T. Phalp. Requirements engineering for e-business advantage. *REJ*, 11(1):4–16, 2005.
3. R. J. A. Buhr. Use Case Maps as Architectural Entities for Complex Systems. *TSE*, 24(12):1131–1155, 1998.
4. Federico Crazzolara. *Language, Semantics, and Methods for Security Protocols*. Doctoral dissertation, BRICS, daimi, May 2003. PhD thesis. xii+160.

5. 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.
6. C. A. R. Hoare. Communicating Sequential Processes. *Commun. ACM*, 26(1):100–106, 1983.
7. Ivar Jacobson, Maria Ericsson, and Agneta Jacobson. *The object advantage: business process reengineering with object technology*. ACM Press/Addison-Wesley Publishing Co., 1994.
8. H. J. Johansson, P. McHugh, A. J. Pendlebury, and W. A. Wheeler. *Business Process Reengineering–Breakpoint Strategies for Market Dominance*. John Wiley & Sons, 1993.
9. Raman Kazhamiakin, Marco Pistore, and Marco Roveri. A Framework for Integrating Business Processes and Business Requirements. In *Proc. of EDOC'04*, pages 9–20. IEEE Press, 2004.
10. Pericles Loucopoulos. The S3 (Strategy-Service-Support) Framework for Business Process Modelling. In *Proc. of REBPS'03*, volume 75 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.
11. Gavin Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(1–2):53–84, 1998.
12. 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.
13. Heinrich C. Mayr, Christian Kop, and Daniela Esberger. Business Process Modeling and Requirements Modeling. In *Proc. of ICDS'07*, page 8. IEEE Computer Society, 2007.
14. 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.
15. Bashar Nuseibeh and Steve Easterbrook. Requirements Engineering: a Roadmap. In *Proc. of ICSE'00*, pages 35–46. ACM Press, 2000.
16. Jason Rubens. Business analysis and requirements engineering: the same, only different? *REJ*, 12(2):121–123, 2007.
17. R. J. van Glabbeek. The linear time-branching time spectrum. In *Proceedings of the Theories of Concurrency: Unification and Extension*, pages 278—297, 1990.
18. S.A. White. Business Process Modeling Notation (BPMN) Version 1.0. *Business Process Management Initiative, BPMI. org, May*, 2004.