Business Processes Modelling and Diagnosis

Abstract: The computerization of relevant information in organizations is increasingly becoming a necessary reality in companies that want to be present in a market that is characterized by innovation, adaptability and where bigger amounts of information are increasingly available and accessible to everyone, the use of Social Collaboration tools in organizations become increasingly crucial to keep a business running (Brocke et al. 2018). In (Alter 2013), work systems are described as systems "in which human participants and machines perform work (processes and activities) using information, technology and other resources to produce specific products/services for specific internal and external customers". The computerization of processes is not, however, so complete because all the formal and informal relationships among employees, which underlie each organization and have a high impact on the correct definition of processes, are not correctly considered.

In order to mitigate this problem, this article presents a proposal for representing, in computer systems, formal and informal relations between employees and the consequent integration in organizational processes in order to provide automatic diagnosis of highlighted processes.

1. Introduction

As SC (Social Collaboration) tools are becoming a crucial part of nowadays organizations, three reasons for the rise of its popularity are defined in (Brocke et al. 2018):

- SC tools exhibit the characteristics of "malleable software", while tradition systems like CRMs or ERPs are developed to meet well-defined purposes;
- The impact in the organizations is achieved individually by each employee or by a group of employees;
- Creating in organizations the need to develop elaborated collaboration strategies;

SC tools are becoming the most relevant vehicle for any organization's success. These types of work systems rely on human and machine interaction along with information processing to create useful outputs to be used by organizations. The present paper's scope focuses on business processes automatic diagnosis and how formal and informal relations can be an important assessment factor in said diagnosis.

Current modelling languages, although widely used for process modelling in organizations, also have some limitations when it comes to precisely define and represent the reality of these. In (Lara et al. 2017), the authors identify three main limitations in the existing process modelling languages:

- The existing disconnection between operational technology and information technology which led to the creation of operational and IT teams with completely different skills and barely any kind of communication between them,
- The functionality limitations and lack of flexibility on modelling languages and tools, which target are mainly business and IT domains,
- Inadequate approach of modelling languages to address specific industry verticals.

One limitation we're focusing in this paper is the lack of an explicit and inambiguous representation of the formal and informal relations between employees, as they represent an important part of a company's process alignment with its corresponding strategy (Sungur et al. 2014). Since BPMN is the most commonly used modelling language for representing all kinds of processes, an attempt to extend this language's specifications was performed in order to accommodate the mentioned limitation.

Although extending BPMN's specifications allows the graphical representation of formal and informal relations in an organization, processing this information in order to diagnose a process, requires the use of a different language that must be, at the same time, sustained by OMG. Thus, the best solution to complement BPMN visual representation of processes, is OCL based rules that will allow for the process diagnosis.

In this paper we propose a BPMN language extension (class diagram metamodel) in order to cope explicit representation of formal and informal relations. Several rules taken from literature, based on formal and informal relations principles, are used to test our model. We adopted the USE tool to execute and check whether the given OCL version of rules are satisfied or not when the class diagram is instantiated and, thus, a diagnosis is performed.

2. An organization's formal and informal structure

Before presenting the methodology behind this study, it is important to clearly understand what is an organization, how it is formally structured and how informal structures may play a major role in a company's success.

According to (Pugh 1990), an organizational structure "consists of activities such as task allocation, coordination and supervision that are geared toward organizational goals". This maybe a definition that, in a more simplified and unanimous way, better describes the meaning of an organizational structure.

It is necessary for an organization, in order to put its predefined structure into practice, to make it inambiguous and clear to what actually makes up this structure: its employees. An organizational chart represents, in a compact and graphic way, relationships between different parts of the business and allows for a quick analysis at the company's structure in a more efficient way than any other descriptive method (Alexander Hamilton Institute 1923). Visually speaking, an organizational chart will represent the authority, responsibility and information flow in the formal structure of an organization, depicting each one of its divisions in a box shape, which will relate to each other by lines connecting them. These lines will also define, based on its direction, the decision making and reporting power (downward and upward respectively) as well as work/communication relationships.

As there are different types of organizations that require different structural needs, there also are different types of charts to represent each one of them:

- Hierarchical charts are the most commonly used type of organizational chart. They represent all the employees or a group of them, which have at least one superior (excluding one, usually the CEO);
- Horizontal or flat charts are defined by the low or inexistent number of hierarchical levels. The influence of each employee is levelled and, therefore, the authority flows horizontally instead of vertically.
- Matrix charts group employees according to their relevant skills and projects with which they are associated.

It is clear that formal relationships are well addressed when it comes to its graphical representation, with the use of organizational charts. Unfortunately, this diagram lacks in the representation of informal relations. The informal structure covers all the ‘soft’ components of the organizational structure from the relationship between their elements, the perception and opinion of those involved as well as their needs and emotions (Wang et al. 2002). The informal network within an organization and the knowledge management practices are usually related, since this network’s correct management allows for a better and more efficient knowledge management, which has been considered in the past years a critical element to produce sustainable competitive advantages (Wang et al. 2002).

Just like we’re able to graphically represent the formal relations within an organization, the same type of representation for the informal relations is needed but inexistent. As a matter of fact, different studies regarding this subject have already been conducted by several researchers, of which (Castelfranchi et al. 1992) stands out. The author proposes a way to operationalize social dependencies between resources and agents. Other authors made further developments on *Castelfranchi’s* theory, namely in (Odell et al. 2003) where, using UML modelling, the authors operationalize *Castelfranchi’s* agent’s dependence theory.

3. Graphic modelling approach

3.1. Formal relations representation

Representing the formal relations within an organization using the existing UML languages poses a problem given that its specifications (at the light of OMG) do not postulate a way to explicitly make this representation possible.

Using actors from *Use Cases* to represent the organization hierarchy is not a viable solution. The organization hierarchy of Figure 1 is not valid by UML standards. As defined by OMG “an actor models a type of role played by an entity that interacts with the subject (…). Actors may represent roles played by human users, external hardware or other subjects” (Object Management Group 2011). Although an actor could represent a certain employee or role, the only situation when two actors are, somehow, associated is when a generalization occurs (an actor inherits the properties of another). As it is defined in UML superstructure specification: “An actor can only have associations to use cases, components and classes” (Object Management Group 2011), eliminating the possibility of creating direct associations between these.

Another possible approach to represent the formal relations in an organization would be by using BPMN specifications. Just like in use cases, we are able to represent participants in a process through the *Pool* element. A *Pool* is the graphical representation of a participant of an interaction between two or more entities (Object Management Group 2013).

The use of BPMN to represent formal relations was sustained on two premises: each employee/role would be represented by a *Pool* and a link to represent a formal relationship between both entities. Representing a participant as a *Pool* didn't pose any problem however, the link between them only exists in the form of a message flow. Also, the only way a message flow can link two *Pools* is when these *Pools* represent black boxes (blank pools) and since an entity in the form of a *Pool* will have tasks associated, the use of BPMN becomes quite limited for the purpose of operationalizing formal relations.

Finally, the use of objects diagram stands as the most viable option for this representation. IBM defines the object diagram as "a tool that provides a snapshot of the instances in a system and the relationship between the instances" (IBM). The correct instantiation of a classes diagram that contains a *Role* class and recursive association will allow the representation of hierarchical levels between roles/employees, which will be shown further in this paper.

3.2.Informal relations representation

Representing the informal relations between employees is also imperative in order to perform a complete diagnosis to a business process since, as was previously mentioned, the informal relations are increasingly influencing the way the decisions are made in a business.

There are two papers that stand out when it comes to this subject: Informal Process Essentials (Sungur et al. 2014) and Supporting Informal Processes (Sungur et al. 2014). In these complementary articles, the authors reiterate the importance that informal relations have in any organization and recognize the lack of tools to enable its modelling. For starters, the authors outline the properties of an informal process and its corresponding requirements (four in total) in order to make it operationally feasible. Afterwards, they commence a "best-fit" process in order to find the tool that better complies with the previously inferred requirements, which resulted in an assessment to the possible use of BPEL, petri-nets, BPEL4People, BPMN, among others. As none of these activity-oriented approaches complied with all the requirements previously established by the authors, they propose an approach to "describe a set of interrelated resources which work together to achieve a collective goal" (Sungur et al. 2014). In figure 2, a conceptual meta-model of this proposal is depicted.

As it is illustrated, the authors resort to the *Relationship* recursive association to represent informal relations between resources (in this case, human performers). This implementation will also be used, later in this paper, in the proposal of the BPMN language extension,

Besides the ones mentioned, two articles that also contributed to developing the methodology of this work were (Brambilla et al. 2011) and (Awad et al. 2009). On both papers the authors, although with different objectives, propose an extension of BPMN notation in order to operationalize the needs of their proposal.

In the first paper, the authors affirm that social networking is increasingly an important tool to help organizations harness the value of informal relationships and weak ties within the organization. This interest generates the growth of Social BPM which focuses on increasing the organization's performance through adding, in a controlled way, external stakeholders to the execution of business processes. The increase in performance may be obtained through exploiting weak ties within the organization, increasing transparency and participation to the decision procedures and involving communities in activity execution. To cover these aspects, business process notations require new features whereas the authors propose an extension to the existing BPMN notation for capturing social requirements, which will enable the description of social behaviours within BPMN diagrams.

On the second paper, the authors focus on the correct allocation of tasks of a process to resources (in this case, human resources) using BPMN and, similar to the methodology that will be presented in this document, they also extended the current specification of BPMN and used OCL to allow for the corresponding restrictions to be processed and applied to processes. The allocation rules that were used in (Awad et al. 2009) were adapted from (Russel et al. 2005) and originated: *Direct Allocation*, *Role-based Allocation*, *Capability-based Allocation* and *History-based Allocation*. Figure 3 shows the extension made to BPMN meta-model in order to incorporate this resource-based allocation feature.

In Figure 3, a model shows the classes added in (Awad et al. 2009) to the BPMN meta-model besides the existing ones (represented with «Core» prefix).

The operationalization of the abovementioned task allocation constraints, as well as the BPMN extension in Figure 3 are illustrated in Figure 4.

The authors used an open-source process modelling tool (https://www.openhub.net/p/oryx-editor) where we are able to see a simple process in which each task has a constraint associated. For instance, regarding a task *Enter Leave Request*, the allocation constraint used is the *Role-based allocation* and the corresponding value is *Employee*, meaning that this task may only be allocated to an employee.

An approach on how to graphically represent the informal relations between employees has already been addressed as well as the extension of BPMN language to accommodate new functionalities for the corresponding purposes. Although these approaches are suitable for the purposes in view, they do not fully comply with the objectives of this paper due to existential differences in the goals of said papers. However, using a BPMN extension as a vehicle to represent the informal poses a solution for the problem in hand.

4. BPMN metamodel extension proposal

The diagram shown in figure 5 illustrates our proposal to extend the current BPMN's specification. It is represented as a class diagram and contains already existing relevant classes (represented with «Core») to provide context to the added classes.

The provided diagram represents part of the solution that will allow us to operationalize the formal and informal relations in an organization and, consequently, allowing for the process diagnosis. It should be noted that this model only represents one organization (not a set of organizations).

The «Core» classes represented in the model were added due to its relevance in the context of the extension proposed. Other elements of the BPMN structure were not considered and, therefore, represented in this model since they do not have any direct influence (or are directly influenced) in the added elements. The added classes to the BPMN meta-model (all classes without the «Core» indication) are: *Role*, *Competence, Employee* and *Control Task.* Although all of these classes pose an important role in the process diagnosis, only the classes *Role* and *Employee* will allow for the representation of formal and informal relations, respectively.

The *Role* class will be graphically represented by a *Lane*, hence the direct association between them. Whenever a *Role* is represented as a *Lane* in a process, it must not appear in duplicate. Also associated with this class, is the class *Employee* with a one-to-many association since, in a given organization, an employee must necessarily perform a role even though he may have responsibilities other than the ones imposed by its formal role. Also represented in the class *Role* is the recursive association *Hierarchy*, which will define the formal relations within an organization. Given that each employee plays a role in an organization, this association will define the hierarchy level of the given employee and since each employee may only have one direct hierarchical superior and none or many hierarchical inferiors, this is a one-to-many association. An example on how this relationship is operationalized can be seen in figure 6 (instantiation created with software USE).

The *Competence* class is directly related to the *Employee* class and the *Core* class *Task.* The first relationship refers to the soft/hard skills that a given employee has and that are relevant to the organization. This association will create constraints when assigning tasks to employees that are not skilled enough to be responsible for such tasks.

Competence and *Task* are related to each other by the associative class *Skill.* Whenever determined task exists and has one or more required skills associated, the *mandatory* attribute will define the compulsory level of such skill for that given task and it consists of a *Boolean* value – true for a mandatory skill and false for a not mandatory skill. The specification of mandatory and non-mandatory skills is indispensable to correctly assess and diagnose processes and its task allocations, as we shall see later on the class instantiation example.

As shown in figure 7, we can see two skills as requirements for concluding a specified task where, for this one in particular, one is mandatory and the other is not (defined by the *mandatory* variable). Later in this paper, the process implications of possessing a mandatory and/or a non-mandatory competence will be explained.

The *Employee* class is one of the main added features to the presented BPMN extension and has a recursive association, *Dependency,* as well as other associations with the classes *Task, Role* and *Competence.*

The *Dependency* association will represent the operationalization of the informal relations between employees in an organization. Since it represents merely relations between individual employees, it gets the shape of a recursive association in this class where an employee maybe influenced by– *influenced* – or have the power to influence another – *influencer*.

Whenever determined task is defined, it is necessarily associated with at least one employee, whom will be responsible for its completion meaning that no task can remain without a responsible employee. Also, an employee that is added to the system cannot remain with no tasks associated, it's responsible for, at least, one task.

In figure 8 diagram, the instantiation of the formal and informal relations is presented between three employees, each one being the detainer of a specific competence that is necessary to successfully complete a task. In this example, the employee is responsible for the task although he only detains one of the three skills needed. This is possible because this employee is also *Influencer* to an employee (informal) that detains the second competence and hierarchically superior (formal) to the third employee that detains the last one. These exceptions are only conceivable through the use of OCL and its implementation will be explained further in this paper.

Finally, the extension to the core class Task may also be defined as a *Control Task*. This class was added in order to allow for the execution of a process diagnosis that assesses the allocation of control tasks to employees. This scenario, to be presented in the next chapter, is represented as a rule for process diagnosis based on (Lee 1988).

At this point, it is clear how the previously mentioned authors and their work was important to develop the theory presented in this paper. On one hand, the contribution of different authors regarding the dependency relationships between agents and on the other hand, the extension to the existing specification of BPMN.

5. Operationalization of process diagnosis

Although the formal and informal relations between employees can now be operationalized, the business process diagnosis still needs to be implemented with recourse to OCL. Just like the previously mentioned work *Enabling Resource Assignment Constraints in BPMN* (Awad et al. 2009), a metamodel extension was devised in order to support the added elements and the OCL language will provide with the mechanism to create the business process diagnosis.

This diagnosis will occur based on inferred rules that define each organization's policies, culture and vision. Four rules, mainly based on (Castelfranchi et al. 1992) and (Lee 1988), were created in natural language and transformed to OCL for exemplification purposes:

1. *An employee may only be associated with a task if he has all the required skills. If he only has the mandatory ones, he must have the power of influence (either formal or informal) over an employee who detains the non-mandatory qualifications.*
2. *A task and its correspondent control task (if any) cannot be associated to the same employee.*
3. *No employee should be assigned a control task if the task to be controlled is assigned to another employee who is his superior or if there is any kind of dependency relation between them.*
4. *If an employee A is hierarchically superior to B, and B is, in turn, hierarchically superior to C, then A is hierarchically superior to B.*

The use of these rules in the software USE implies a transformation to OCL language and a posterior adaptation to USE, given that this software doesn't support all the existing OCL functions. As defined by the developers, "*USE is a system for the specification of information systems. (...) Contains a textual description of a model using features found in UML class diagrams. Expressions written in the Object Constraint Language (OCL) are used to specify* *additional integrity constraints on the model*" (The UML-based Specification Environment 2015). In Figure 9 an overview of the USE system is provided.

When validating a system, a modeller specifies its model and observes the states generated by the given system. The simulation of a change in a system state is provided by USE and its tools based on UML and OCL.

As was mentioned, USE relies completely on the textual description and its constraints as input for the conception of a model. The following images contain part of the class's description used in the construction of the metamodel previously presented.

As we can see, both *Employee* and *Task* classes contain functions that will later be used to create the constraints that will provide with the process diagnosis. Each one of these functions returns a *Set* of objects that respects the conditions defined in the function itself:

- *subordinates* return a set of hierarchically inferior employees to a certain employee.
- *influencedEmps* return a set of employees that are influenced (informally) by another employee.
- *influencial* return a set of employees that are either formally or informally influenced by a certain employee.
- *mandatory_skills* return a bag (set with duplicates) of skills that are mandatory requirements of a given task.
- *optional_skills* return a bag of skills that are optional requirements of a given task.

As soon as all the classes are defined, the associations between them must also be created as shown in Figure 10.

With all these components textually specified, USE automatically generates a class diagram based on the description created. This diagram is shown in Figure 11 along with all the names of the associations created.

Lastly, the invariants (constraints) must be defined. Each invariant, now in OCL, is based on the previously presented rules in natural language (Figure 10).

Note that the *skillAssign* invariant and the other invariants that will be shown, are defined in order to be integrated with the USE system. These same constraints, if created under a software which integrates all of OCL functions would have a different definition.

In Figure 12 is shown an object diagram representative of the system in a given moment in time and the validation of all the invariants.

As it shown, the *skillAssign* invariant is not satisfied since the employee responsible for the task – Joao - doesn't have all the required skills to perform it. However, if the employee Jose, who is hierarchically inferior to Joao, has the missing required skill, the invariant is satisfied. Alternatively, this invariant would also be satisfied if Joao could be able to informally influence Jose. Both cases are shown in Figure 13.

This invariant is not satisfied whenever the same employee is responsible for a certain task and its correspondent control task, as shown in Figure 12.

The invariant *controlInfluenceAssign* is not satisfied when an employee responsible for a control task is hierarchically inferior to the employee responsible for the task to be controlled or the first employee is informally influenced by the second.

In Figure 14 is shown a condition where the invariant is not satisfied. This happens because the employee Joao is responsible for a task that controls another, which is allocated to an employee that has influence (informal) over Joao. This incorrect task allocation may happen, not only because of the existence of informal dependence between employees, but also when there is formal dependence (Figures 15 and 16).

The last invariant follows the concept of the hierarchical chain, making sure that it is respected when represented in the system. For instance, when three employees are represented and each one of them is hierarchically inferior to the other, this event must be correctly defined, otherwise the invariant will not be satisfied (Figures 17 and 18).

6. Conclusion

There are several existing tools that allow the computerization of business processes. Such tools allow us to create and customize processes through the use of known modelling languages, which implies dealing with the constraints these languages have. It is important for the modeller that the system provides with a process diagnosis while it is being created given that a simple task allocation error may compromise the given process with loss of effectiveness.

This work shows that it is possible to develop a business process modelling system, based on existing tools/languages, that will perform an automatic diagnosis of such processes using previously added information regarding the organization.

This added information may be a set of rules, like the ones that were used as example in this paper. They resort to previous studies regarding not only formal and informal relations between employees, but also good practices in business management. The correct operationalization of said rules incur in the extension of the existing BPMN metamodel, which will allow to accommodate the concepts underlying this subject. The transformation of these rules to an OMG compliant language was necessary in order to enable a process diagnosis over BPMN, hence its transformation to OCL. The role of the USE system in the development of this methodology relied on the assessment of the successful diagnosis made on the meta-model's instantiation.

In addition to the rules that were used in this paper to make a process diagnosis (based on formal and informal relations and good practices in business management), other factors such as professional history, socioeconomic status and previous life experiences could be addressed to enrich and consolidate the business processes diagnosis approach.

Besides the use of different factors influencing the allocation of tasks to employees, the development of systems that put into practice the concepts addressed in this paper would also be a valuable asset for organizations to better manage their business's inherent processes.

References

Alexander Hamilton Institute (1923) *Organization Charts.* Astor Place, New York : Alexander Hamilton Institute.

Alter S (2013) Work system theory: overview of core concepts, extensions and challenges for the future. *Journal of the Associationfor Information Systems* 14(2):72-121.

Awad A et al (2009) Enabling Resource Assignment Constraints in BPMN.

Brambilla M, Fraternali P and Vaca C (2011) A Notation for Supporting Social Business Process Modeling. In: Dijkman R, Koehler J and Hofstetter J (eds) *Lecture Notes in Business Information Processing: Preface.* Milano : Politecnico di Milano, pp. 88-102.

Brocke J, Maaß W, Buxmann P et al (2018) Future Work and Enterprise Systems. *Business & Information Systems Engineering* 60(4):357-366.

Castelfranchi C, Miceli M and Cesta A (1992) Dependence relations among autonomous agents. *Decentralized A.I.* 13(3):14**.**

IBM. Object diagrams. *IBM Knowledge Center.* Available at: https://www.ibm.com/support/knowledgecenter/SS8PJ7_9.1.1/com.ibm.xtools.modeler.doc/topics/cobjdiags.html (accessed 01 Oct 2018).

Katzenbach J and Khan Z (2010) *Leading Outside the Lines: How to Mobilize the Informal Organization, Energize Your Team, and Get Better Results.* San Francisco : Jossey-Bass.

Lara P, Sánchez M and Villalobos J (2017) OT Modelling: The Enterprise Beyond IT. *Business & Information Systems Engineering* OnlineFirst:1-13.

Lee R (1988) Bureaucracies as Deontic Systems. *ACM Transactions on Information Systems* 6(2):87-108.

Object Management Group (2013) Business Process Model and Notation (BPMN). [Online] 2013. http://www.omg.org/spec/BPMN/2.0.2/PDF.

Object Management Group (2011) Unified modelling language (OMG UML) superstructure. [Online] 2011. https://www.omg.org/spec/UML/2.4.1/Superstructure/PDF.

Odell J, Parunak H and Fleischer M (2003) Modeling Agent Organizations using Roles. *Software & Systems Modeling* 2(2):76-81.

Pugh S (1990) *Organization Theory: Selected Readings.* Harmondsworth : Penguin.

Ramos P and Fiadeiro J (1999) Diagnosing Process Design: A Formal Approach. In*: Reengineering in Action.* s.l. : Imperial College Press, pp. 131-155.

Russel N et al (2005) Workflow Resource Patterns: Identification, Representation and Tool Support. In: O Pastor and J Falcão e Cunha (eds) *Advanced Information Systems Engineering.* Berlin : Springer, pp. 216-232.

Sungur T, Kopp O and Leyman F (2014) Supporting Informal Processes. In: *The 6th Central European Workshop on Services and their Composition (ZEUS 2014).* Stuttgart, Germany : CEUR-WS.org.

Sungur T et al (2014) Informal Process Essentials. In: *Proceedings of the 18th IEEE Enterprise Distributed Object Conference (EDOC 2014).* Stuttgart : IEEE.

The UML-based Specification Environment (2015) The UML-based Specification Environment. *USE OCL.* 1 7, 2015. Available at: http://useocl.sourceforge.net/w/index.php/Main_Page (accessed 15 September 2018).

Wang C and Ahmed P (2002) *The Informal Structure: hidden energies within the organisation.* s.l. : Management Research Centre Wolverhampton Business School.

Figure 1 - Org Chart built with Actors

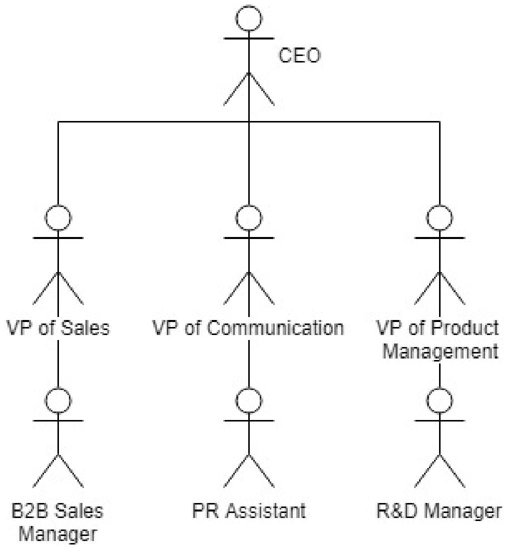

Figure 2 - Conceptual meta-model (Sungur et al. 2014)

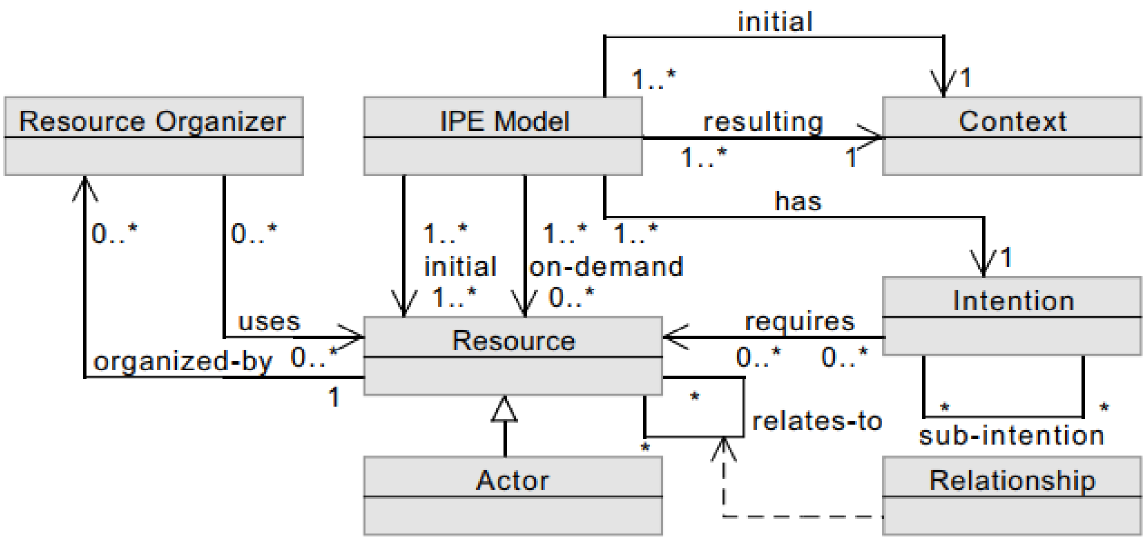

Figure 3 - Extend BPMN metamodel (Awad et al. 2009)

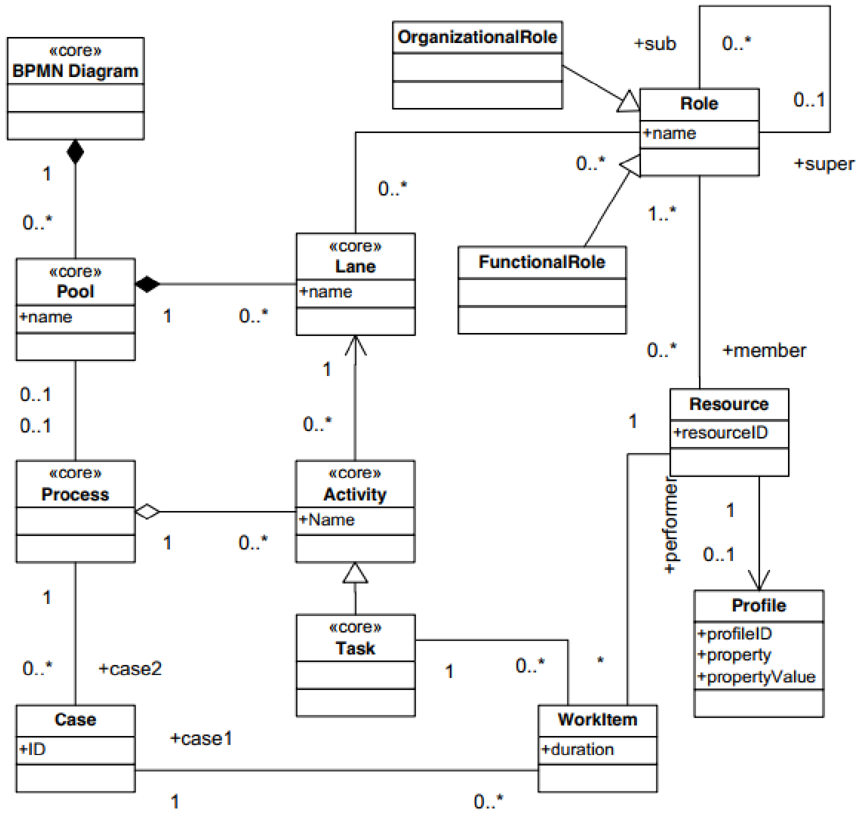

Figure 4 - Task allocation constraints (Awad et al. 2009)

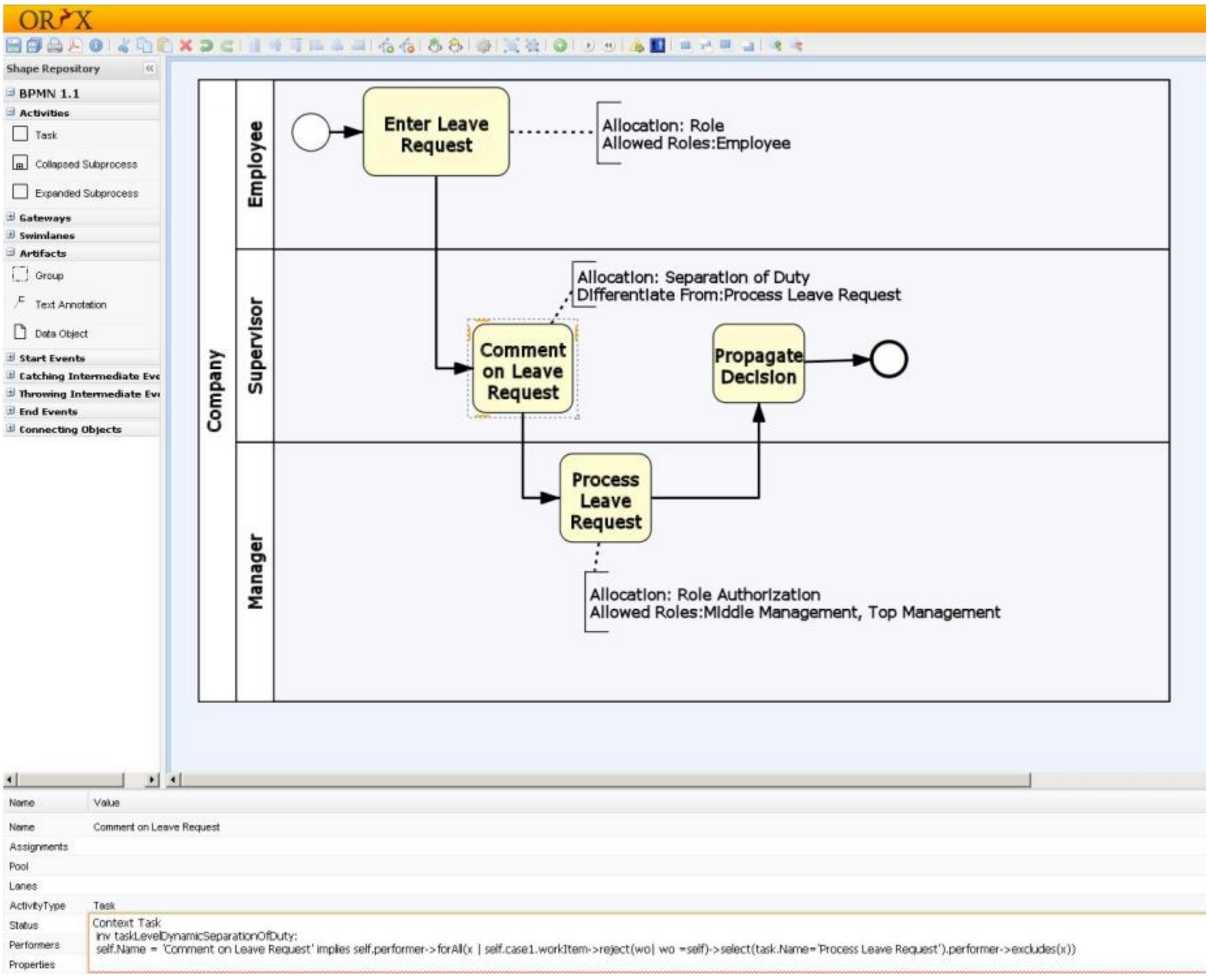

Figure 5 - BPMN extension proposal

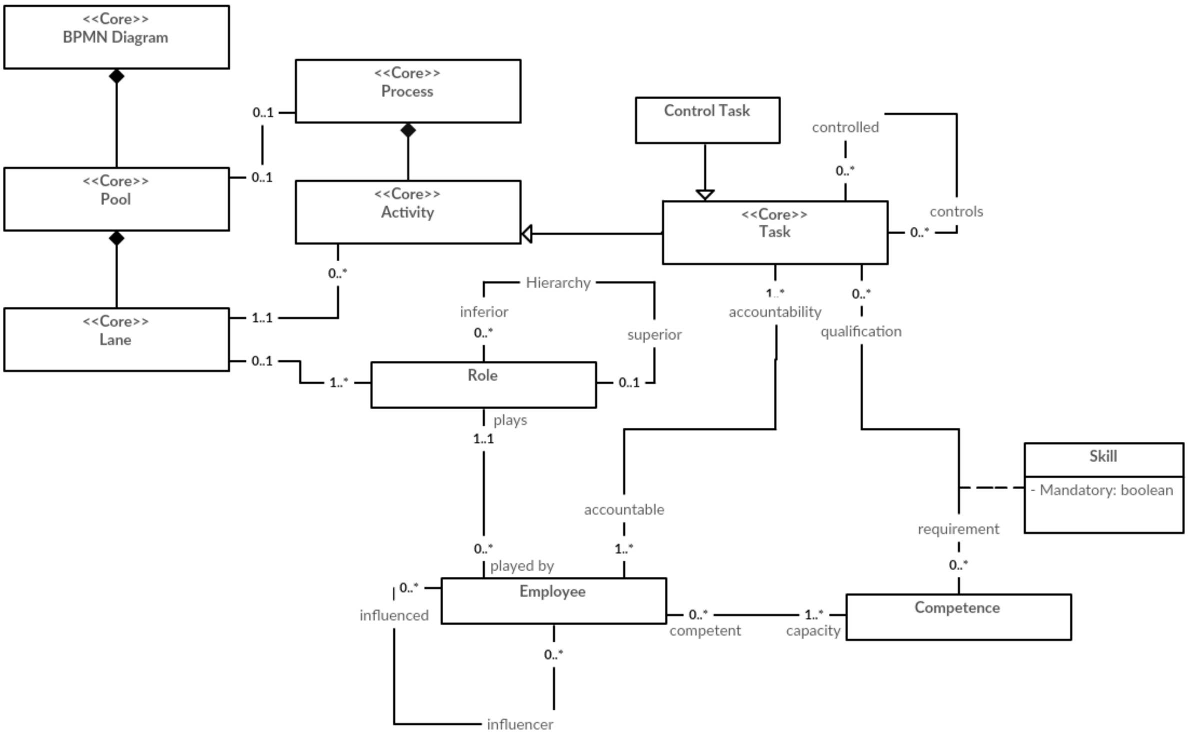

Figure 6 - Hierarchy and Role classes operationalization

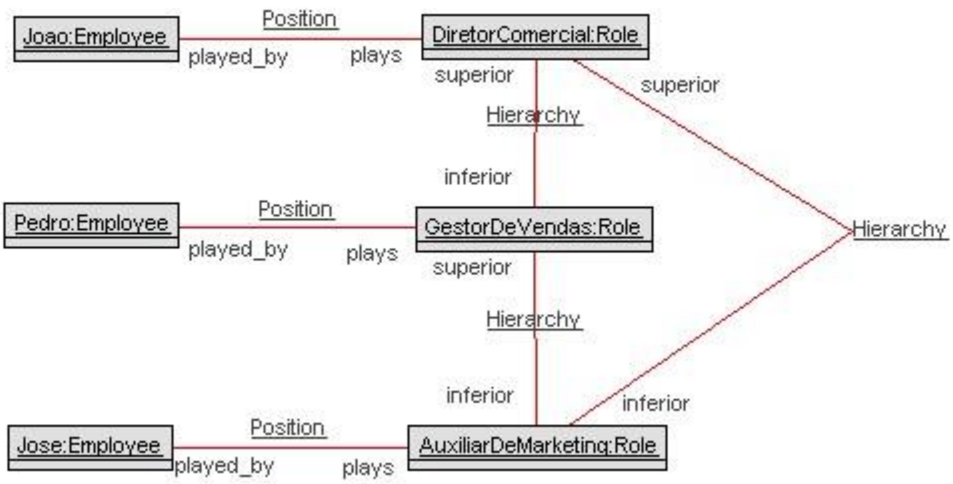

Figure 7 - Competence class operationalization

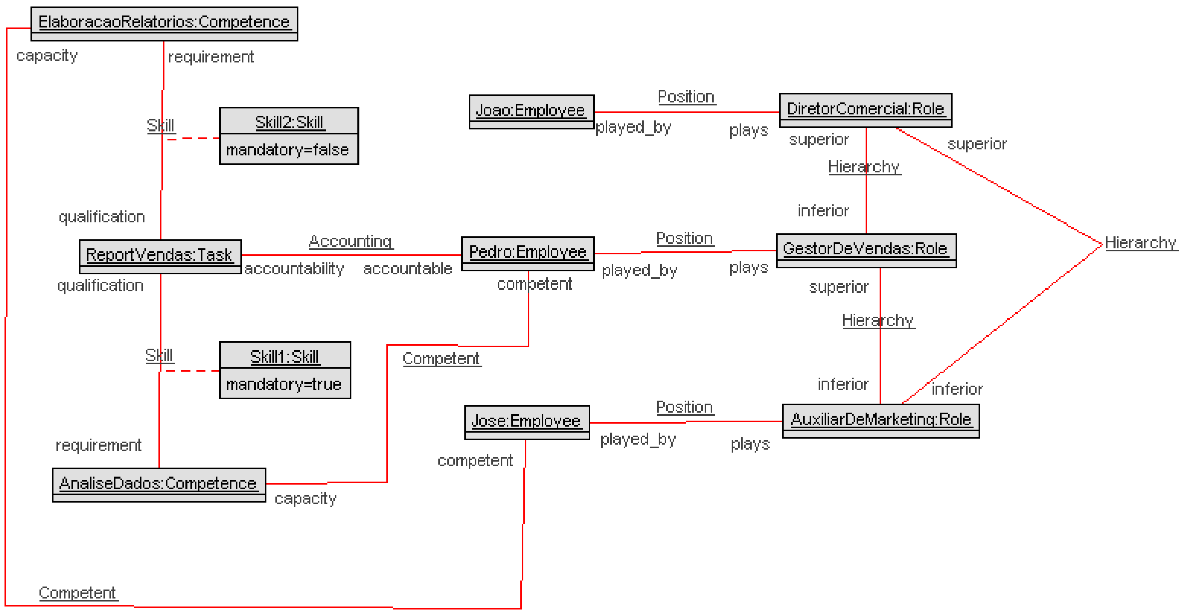

Figure 8 - Employee class operationalization

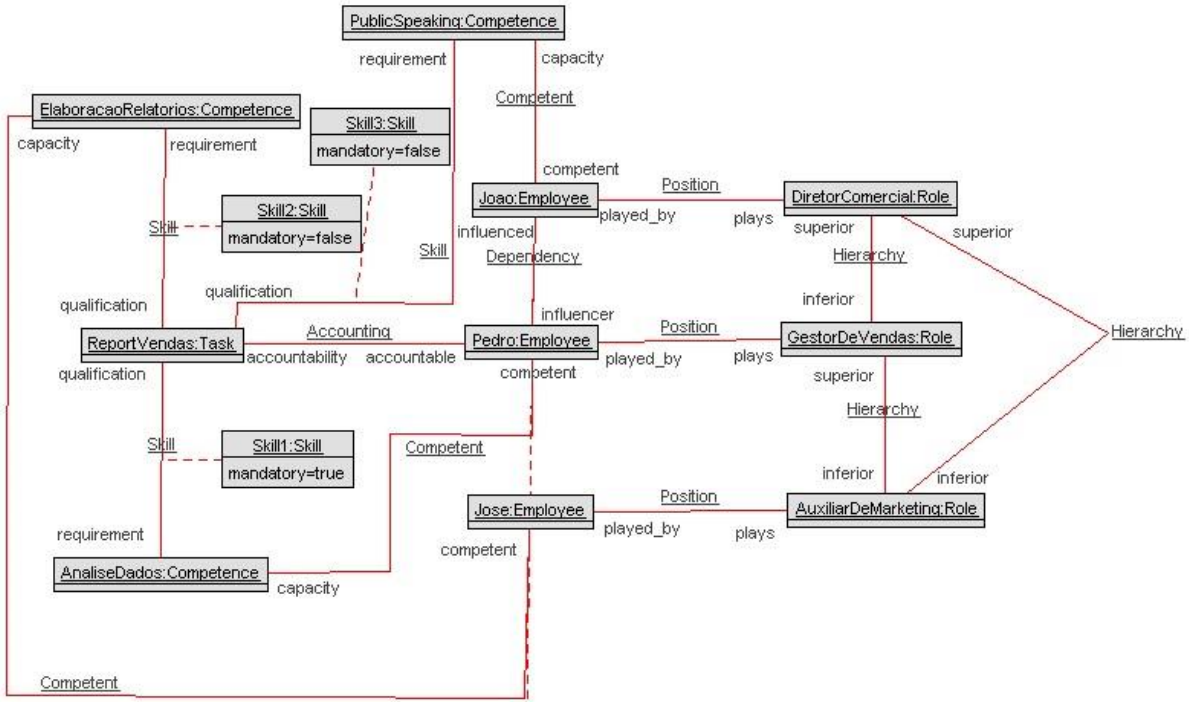

Figure 9 - USE approach - an overview[1]

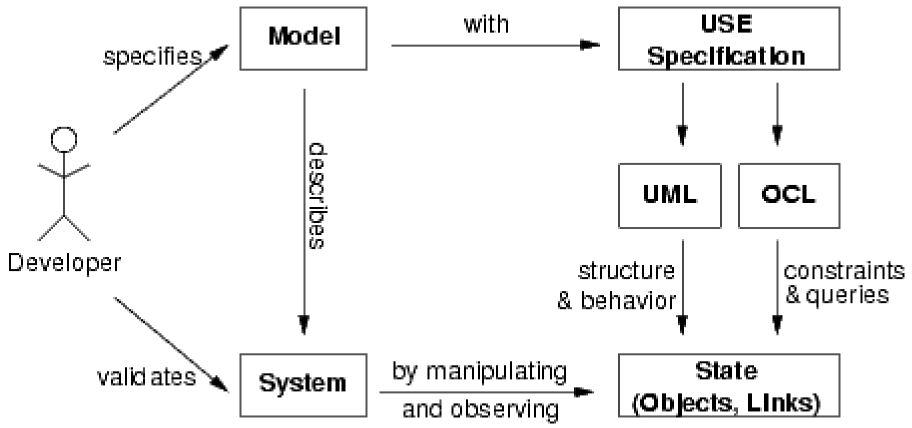

Figure 10 - Associations between classes

```
association Hierarchy between
       Role [0..1] role superior
       Role [0..*] role inferior
end

association Position between
       Role [1..1] role plays
       Employee [0..*] role played_by
end

association Competent between
       Employee [0..*] role competent
       Competence [1..*] role capacity
end

association Accounting between
       Employee [1..*] role accountable
       Task [1..*] role accountability
end
```

[1] Image source: http://useocl.sourceforge.net/w/index.php/Main_Page

Figure 11 - USE Auto-generated class diagram

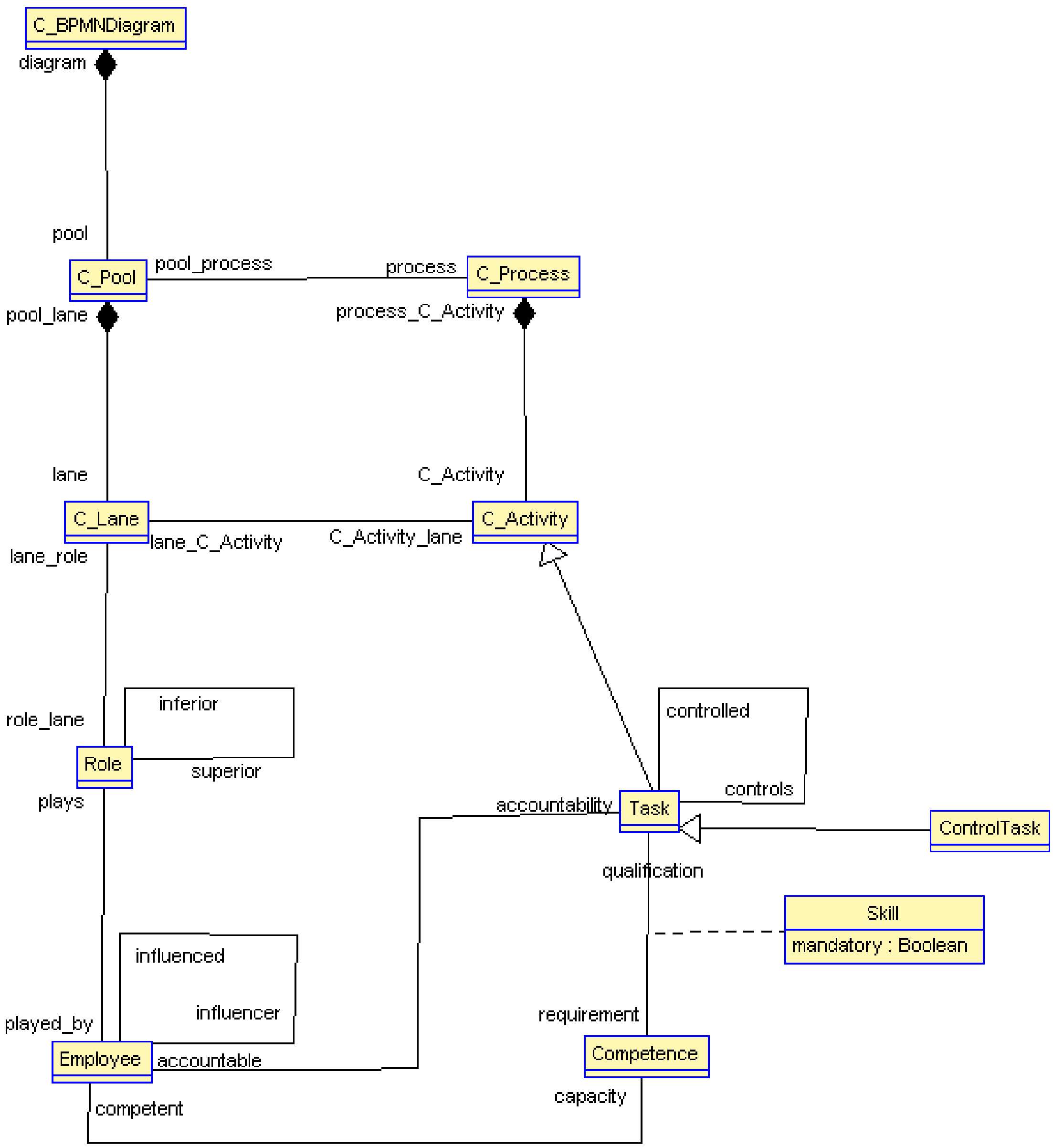

Figure 12 - SkillAssign invariant not satisfied

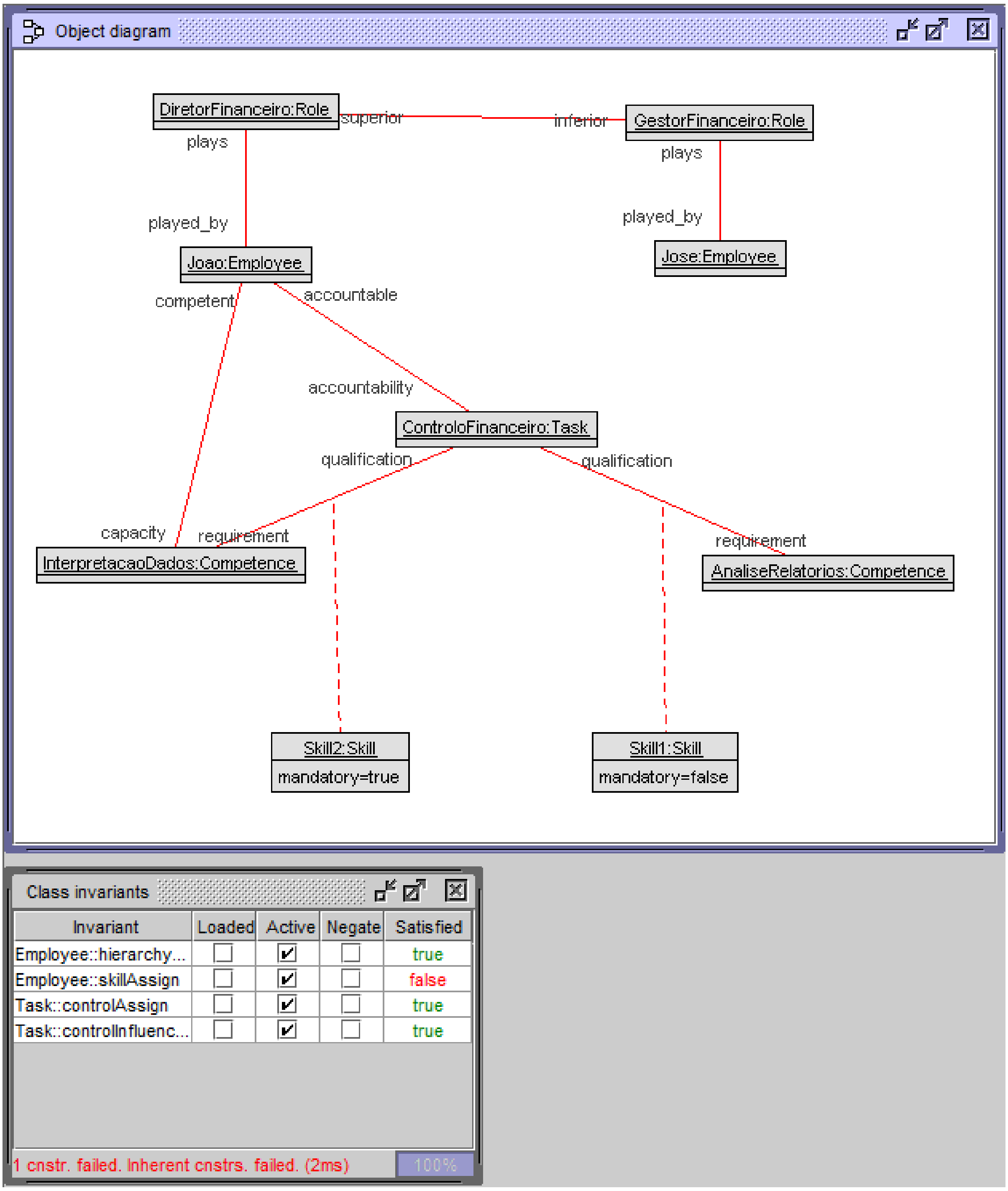

Figure 13 - Rule one: SkillAssign

An employee may only be associated with a task if he has all the required skills. If he only has the mandatory ones, he must have the power of influence (either formal or informal) over an employee who detains the non-mandatory qualifications.

```
constraints

       context Employee

              inv skillAssign:
              self.capacity->includesAll(self.accountability.mandatory_skills())
              and self.influencial().capacity->
              includesAll(self.accountability.optional_skills())
```

Figure 14 - ControlAssign invariant not satisfied

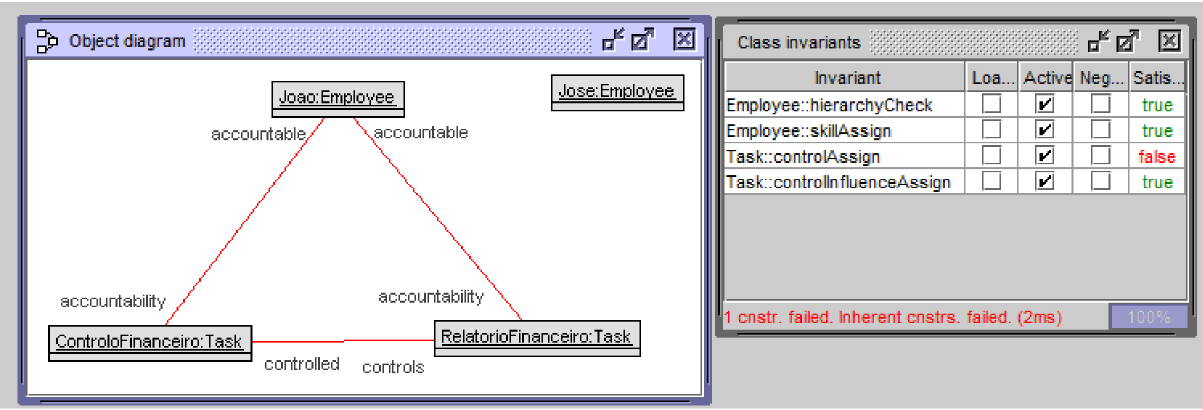

Figure 15 - ControlInfluenceAssign invariant not satisfied (hypothesis 2)

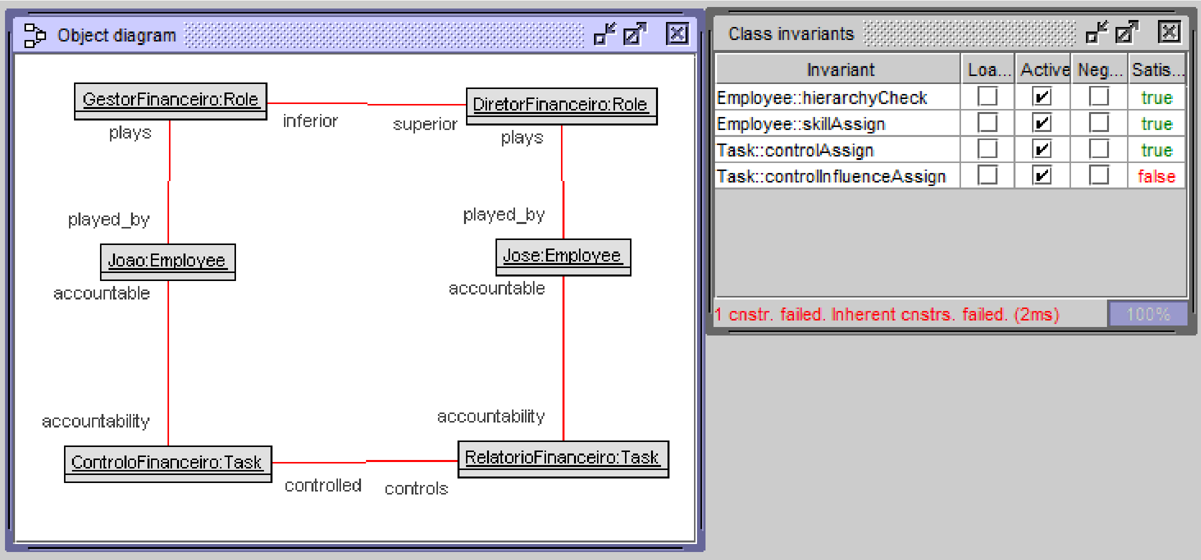

Figure 16 - ControlInfluenceAssign invariant satisfied

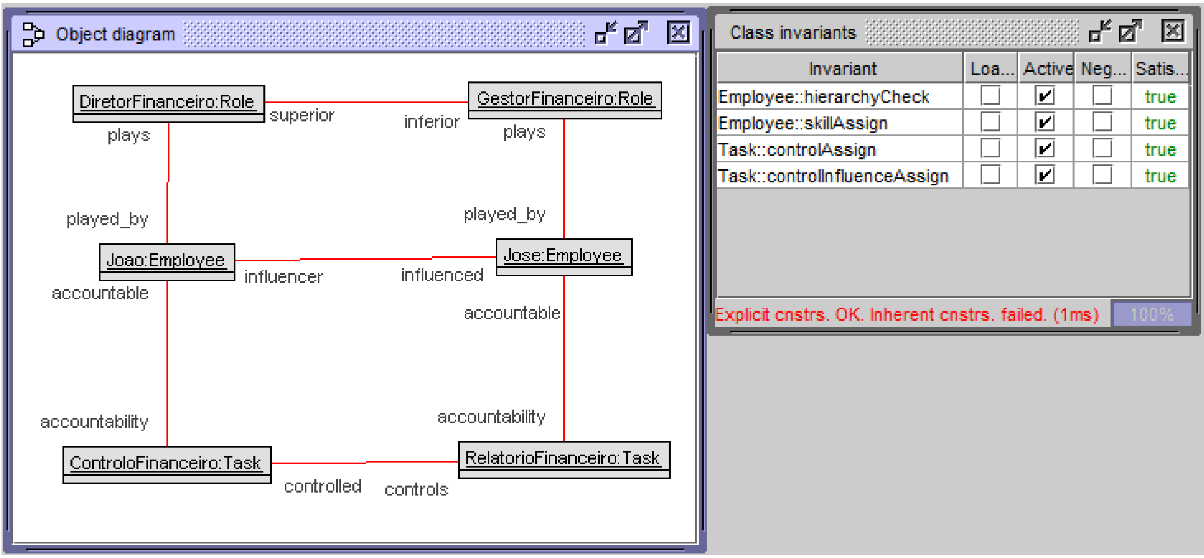

Figure 17 - Rule 4 HierarchyCheck

If an employee A is hierarchically superior to B, and B is, in turn, hierarchically superior to C, then A is hierarchically superior to B.

```
constraints

	context Employee

		inv hierarchyCheck: Employee.allInstances->
		forAll(e1, e2, e3 | e1.subordinates()->
		includes(e2) and e2.subordinates()->
		includes(e3) implies e1.subordinates()->includes(e3))
```

Figure 18 - HierarchyCheck invariant not satisfied