Analyzing Basic Problem Frames in i* Context

Maria Lencastre[1], Keldjan Alves[1], Renata Melo[1], Fernanda Alencar[2]
[1]Department of Computer Systems, Polithecnical School, Pernambuco University
Rua Benfica, 455 Madalena, 50750-410, Recife – PE – Brazil
{mlpm@dsc.upe.br, kao, rpllfm}
[2]Department of Electronics and Systems, Federal University of Pernambuco
Rua Acadêmico Hélio Ramos s/n, 50740-530, Recife – PE – Brazil
{fmra@ufpe.br}

Abstract

Problem Frames and i are two broadly used approaches that require a good background from the requirements engineering. Both approaches have typical traits that fit in equivalent. On the other hand, there are features that do not fit. Thereby, there are some points which need improvements on both of them. This work analyzes the five basic problem frames, defined by Michael Jackson, in the i* context. Our aim is to identify how can they be represented in i* and what gains are achieved if problem frames concept can be applied in i* modeling (like for example simplicity, completeness and flexibility of the built models).*

1. Introduction

The need to address problem structure and classification, in a more sharply focused and explicit way, is analyzed by Jackson in [3]. In his works he regards particular problem classes as characterized by problem frames. Problem Frames [4] is an approach for requirements analysis and problem domain specifications which groups problems by type, and call them problem frames; they are problem oriented rather than solution oriented. Problem Frames approach represents the real problem through several descriptions about the application domain. There is a repertoire of recognized problem classes which includes the associated characteristics, difficulties and solution methods. With these structures specialization, the developer can emerge and reach incremental advances in software development. In this paper we use Problem Frames, in uppercase, to reference to the approach and problem frames, in lowercase, to reference problem classes.

Considering Problem Frames potentiality to describe problems, in this work we investigate the modeling of the existing repertoire of basic problem frames [4], also called elementary frames, through i* diagrams [9]. This will make possible to evaluate the real improvements in i* approach, if such classes of problems are available during i* modeling. Our strategy was to model only these basic problem frames, because they gather some important features, such as: they are a representative variety of different kinds of problems; they have the advantage of being limited to 5 diagrams; and they introduce the concept of patterns in the problem space.

This paper is organized in the following way. Section 2 describes problem frames concepts. Section 3 shows the i* approach. Section 4 describes the five basic problem frames, from the existing repertoire, and presents the corresponding i* models; this section also illustrates the use of both approaches using part of the car parking example taken from [6]. After, in section 5 a discussion about the obtained models is done. Finally, section 6 draws some conclusion and points out directions for future work.

2. Problem Frames Concepts

A problem frame is a kind of pattern that captures and defines commonly found classes of simple sub-problems. It is just a generic problem, but instead of showing specific domains and requirements, it shows types of domains and types of phenomena. A problem frame distinguishes some broad types of domains - used as domain marks - according to their principal characteristics. Each type of domain demand different kinds of description and raise different development concerns. Domains can be: (i) Causal domain: its properties include predictable causal relationship among its causal phenomena, they are referenced as a C in the right border corner of the domain, see Fig 1, the Controlled Domain. (ii) Biddable domain: usually consists of people; it lacks positive predictable characteristic of a biddable internal causality; they are referenced by a B, see Fig 5, the Operator Domain. (iii) Lexical domain: this is a physical representation of data, that is, of symbolic phenomena. It combines causal and symbolic phenomena in a special way; they are referenced by an X, see Fig 13, the Workpieces Domain. The phenomena, at the interface between domains, can be classified as: Causal phenomena (C), which are directly caused or controlled by a domain (events, roles, or states relating entities), events can also be identified by (E); and Symbolic phenomena (Y), which symbolize other phenomena and relationships among them (values, truths, states relating values).

Each problem frame has a concern that must be addressed in any problem of the class. The concern identifies the descriptions one must fit together properly in a correctness argument: requirement, specification and domain description. If one tries to fit a problem into an inappropriate class, the resulting development will certainly be unsuccessful. A problem of a composite class is first decomposed into sub-problems characterized by elementary frames.

Composite frames are essentially parallel compositions of elementary frames. For some composite frames it is necessary to introduce additional created domains that mediate between sub-problems. In general, the creation of such an additional domain becomes a sub-problem in its own right, with its own elementary problem frame. Such an elementary frame is called a partial elementary frame, because the problems it characterizes can never be independent problems in their own right but occur only as sub-problems.

The repertoire of Jackson's basic problem frames includes the following: Required Behavior, Commanded Behavior, Information Display, Simple Workpieces and Transformation. All of these will be used and described in our work. Note that, the repertoire of problem frames is not restricted to elementary and partial elementary frames it also includes composite frames, but here we will only focus on the elementary ones.

3. The i* approach

The i* approach [10] can be used for: (i) obtaining a better understanding of the organizational relationships among the various organizational agents; (ii) understanding the rationale of the decisions taken; and (iii) illustrating the various characteristics found in the early phases of requirements specification. The participants of the organizational setting are actors with intentional properties, such as, goals, beliefs, abilities and compromises. These actors depend upon each other in order to fulfill their objectives and have their tasks performed. The i* approach consists of two models: the Strategic Dependency Model (SD) and the Strategic Rationale Model (SR).

The SD model focuses on the intentional relationships among organizational actors. It includes a set of nodes and links connecting them, where nodes represent actors (depender and dependee) and each link indicates a dependency between two actors (dependum). An actor can depend on another one to satisfy a goal, execute a task, provide a resource or satisfy a softgoal.

The SR model is used to: (i) describe the interests, concerns and motivations of participants' processes; (ii) enable the assessment of the possible alternatives in the definition of the process; and (iii) provide the rationale behind the dependencies between the various actors. In this model, two new types of relationships (links) are incorporated: (i) means-end links, which suggest that one model element can be offered as a means to achieve another model element; (ii) task-decomposition links that describe what should be done in order to perform a certain task. In order to guarantee consistency among models, all SD dependencies are preserved in the SR model.

4. Representing Jackson's Basic Problem Frames Using i* Models

In this section we use the i* for modeling each class of problem introduced in Jackson's basic repertoire. The idea is to make available the existing classes of basic problems for the i* modelers, in a clear and familiar way, allowing them to apply these classes in future works.

This section is organized in the following way. Each basic problem frame in presented in an individually subsection, and each subsection: first presents the idea of the problem frame under discussion and the correspondent diagram, using Problem Frame notation (taken from [3]); then an application of the problem frame is illustrated; after each problem frame is represented using i* model; and, finally, to make it more understandable, the i* model is illustrated through the same example.

The example, which will be used here, is the car parking example taken from [6]: "*To use a car parking system, a client gets a ticket from an entry machine after pressing a button. Afterwards, the car is allowed to enter. When s/he wants to leave the parking place, s/he has to pay the ticket in a paying machine. After paying, the client can leave by inserting the ticket into an exit machine. The system has to control car parking information, validate car entries and exits*". Due to simplification purpose, in this paper we use only part of the example, focusing on the "Car Exit Authorization". This part represents a system that requires functionalities such as to: validate user, verify correct payment, register car exit in the database, and command the car liberation or not. To help in the modeling of the basic problem frames in i* we use the relation presented in [7].

4.1 Required behavior problem frame

The idea of this problem frames is: "There is some part of the physical world whose behavior is to be controlled so that it satisfies certain conditions. The problem is to build a machine that will impose that control".

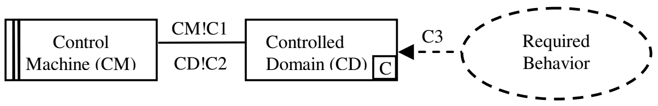


Fig. 1. Required behavior problem frame

The control machine (CM) is the machine to be built, Fig. 1. The controlled domain (CD) is the part of the world to be controlled. The requirement, giving the condition to be satisfied by the behavior of the controlled domain, is called the Required Behavior. The interface of shared phenomena with the machine consists of: C1, which is controlled by the machine (CM), and C2, which is controlled by the controlled domain (CD). The machine affects the behavior of the controlled domain through the phenomena C1; the phenomena C2 provides feedback. The requirement is expressed in terms of C3 phenomena of the controlled domain. These are the requirement phenomena. In general, C3 will be different from C1 and C2. This gap must be bridged by indicative domain properties by the controlled domain.

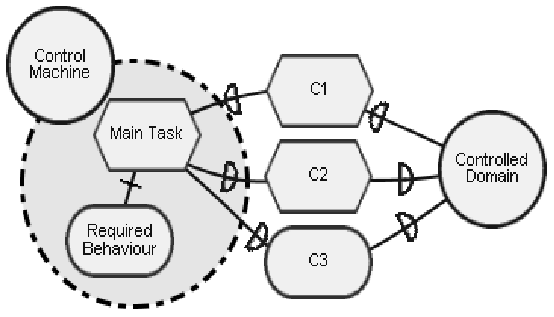


Fig. 2. Required behavior problem frame modeled using i*

In Fig. 2, the corresponding model for the basic problem frame is modeled. There we can see the Behavior problem frame modeled using i*. Both of its domains (CM and CD) where transformed into actors, following [7]. The machine was represented as an expanded actor (using the i* SR model), which has a main task as default. The machine's requirement is transformed to a goal, which is a decomposition of the main task. Note that C1, C2, and C3 are all casual phenomena, which means events, roles, or states (in Problem Frames). So the transformation of each of them is not trivial, because we do not have the same meaning in i*. Each one of these phenomena is transformed in a different way in Fig 2. The reason is that: C1 is a machine command and so it is transformed to a task; C3 is at requirement level (due to its position as a requirement phenomena), so it is transformed to a goal; and finally, C2 is a feed back supplied by the Controlled Domain, and does not necessarily has a correspondence to a task, this only occurs if C2 is an event (as presented in Fig. 2). In the model we choose to represent it as a task, but this is not part of the pattern.

An example of the application of the Required Behavior frame is presented in Fig. 3. The car Exit Authorizer represents the machine CM, and the Exit Machine the controlled domain CD. The required behavior is the liberation of car exit or not, following the car parking payment rules. The C1 phenomena is now represented by the {OpenGate[id,result]}, and the phenomenon C2 by the phenomenon {InformOperationResult} and the C3 phenomena which is at requirement level is represented by the phenomena {ExitAllowed or ExitNotAllowed}.

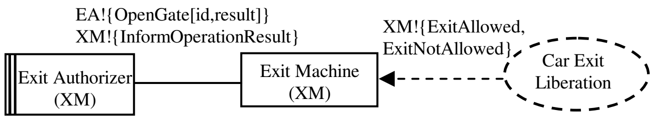


Fig. 3. Example of applying required behavior problem frame

In Fig.4, the application of i* in the car parking example is illustrated, following the Fig. 3.

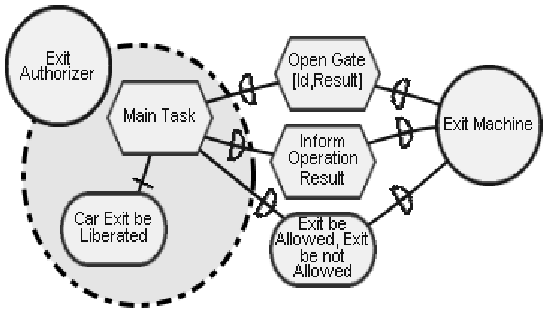


Fig. 4. Example of applying required behavior problem frame modeled using i*

4.2. Commanded behavior problem frame

The idea behind this frame is: "There is some part of the physical world that has to be controlled, in accordance with commands issued by an operator. The problem is to build a machine that will accept the operator's commands and impose the control accordingly". The control machine and controlled domain, and their phenomena C1, C2 and C3, are the same as in the Required Behavior frame (see Fig. 5). But now, there is also an operator, assumed to be a biddable domain, as shown by B marking in the lower right of the rectangle. The operator issues commands (events E4) which are shared with the machine and controlled by the operator. The requirement is named Commanded Behavior. It constraints the behavior of the controlled domain (CD), as represented by the dashed arrow, by describing general rules for its behavior and specific rules for how it must be controlled in response to the operator's command E4. The requirement phenomena of the operator domain are the events E4, which are also specification phenomena that are shared with the machine. The operator is autonomously active (no external stimulus is needed for it to cause the events).

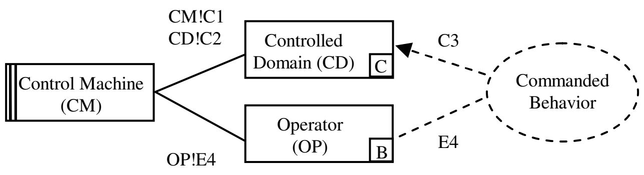


Fig. 5. Commanded behavior problem frame

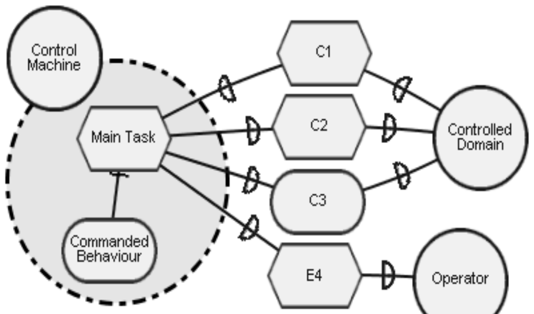


Fig. 6. Commanded behavior problem frame modeled using i*

In Fig. 6, the corresponding model for the basic problem frame is modeled using the i*. As it is very similar to the previous problem frame (the required behavior) we will comment only the transformation of the Operator (which became also an actor) and its interaction: E4 is converted to a task, as it is an event in problem frames at specification level; the OP has the responsibility of executing a task to attend this goal, that is, of emitting the commands. However, there is no E4 goal, since the E4 phenomenon, represented at requirement level, is only reference, not a constraint, to the OP domain. Note that there is no arrow in E4 at this level.

In Fig. 7 the corresponding model, of the basic problem frame, is illustrated, but at this time applying it in the car parking example. The E4 commands are, here, the commands the driver emits through the machine, and are represented as {RequestCarLiberation}. The machine EA will then emit a command to OpenGate sending a parameter which will guide the appropriate Exit Machine behavior. The required phenomena, corresponding to C3 are {exitAllowed or ExitNotAllowed}.

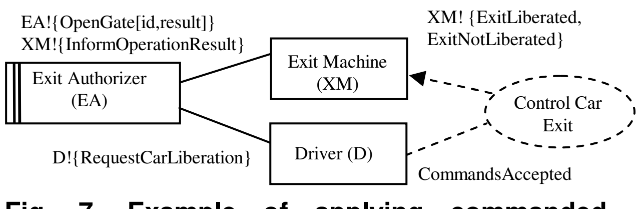


Fig. 7. Example of applying commanded behavior problem frame

In Fig. 8 the application of i* model to the example is presented.

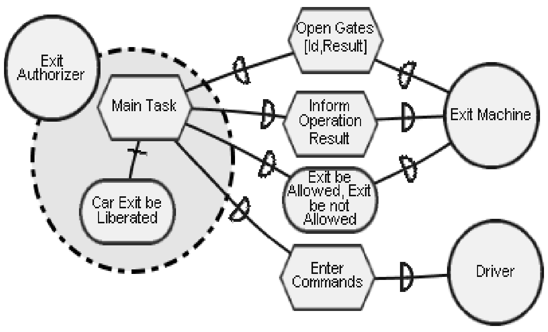


Fig. 8. Example of Commanded Behavior problem frame modeled using i*

4.3. Information display problem frame

The idea behind this frame is: "There is some part of the physical world about whose states and behavior certain information is continually needed. The problem is to build a machine that will obtain this information from the world and present it at the required place in the required form". The part of the world about which information is required is called the real world (see Fig. 9). The display is the part of the world where the information is to be presented. The machine to be built is called the information machine. The requirement is called Display ~ RealWorld, suggesting that it stipulates a correspondence between the symbolic requirement phenomena Y4 of the display domain and the causal requirement phenomena C3 – events or states – of the real world. What shows on the display, interpreted as information about the real world, must be true. The real world is active and entirely autonomous. It causes spontaneous events and state changes, it controls all the shared phenomena at its interface with the machine, and the requirement places no constraint on it.

The machine must satisfy the requirement constraint by diagnosing real world requirement phenomena C3 from C1 phenomenon at its interface. The gap between C1 and C3 must be bridged by causal domain properties of the real world. To produce information the machine must cause changes in the symbolic values and states Y4 of the display domain by causing events E2 at its interface with the display. In Fig. 9 the corresponding model, of the information display problem frame, is illustrated through an example. Here the involved domains include the Real World and the Display Machine. And the requirement is to have a correspondence between the entry sensors of the real world and the park Display.

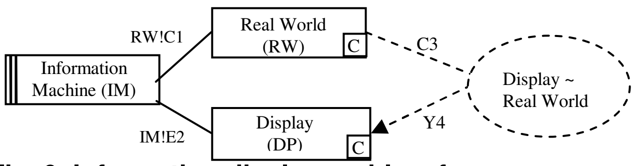


Fig. 9. Information display problem frame

In Fig. 10, the corresponding i* model for example in Fig. 9 is presented. The C3 phenomenon does not appear because it represents only a reference, not a constraint, to the involved domain. Note that the Real World domain was mapped as an actor, although its intrinsic properties are lost in the transformation.

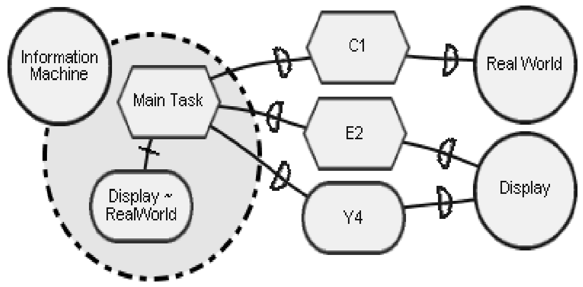
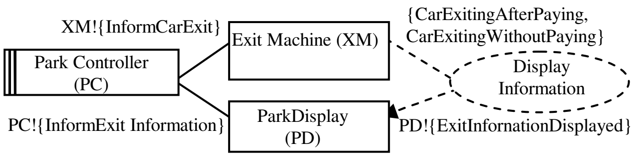


Fig. 10. Information display problem frame modeled in i*

Fig. 11 illustrates the application of this basic problem frame. At this time, the requirement is Display Information according to the correct user payment.



XM!{InformCarExit}
Park Controller (PC)
Exit Machine (XM)
{CarExitingAfterPaying, CarExitingWithoutPaying}
Display Information
PC!{InformExit Information}
ParkDisplay (PD)
PD!{ExitInfornationDisplayed}

Fig. 11. Example of applying information display problem frame

In Fig. 12 the corresponding model (of this basic problem frame) is illustrated, but at this time applying it in the car parking example, following the correspondent application in Fig. 11 (using Problem Frames notation).

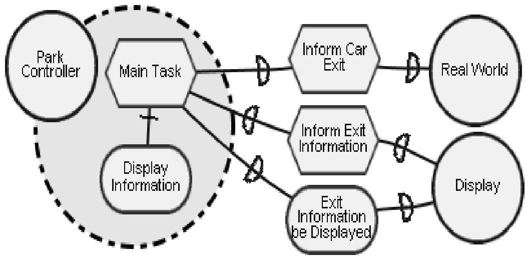


Fig. 12. Example of applying information display problem frame modeled in i*

4.4. Simple workpieces problem frame

The idea behind this problem frame is: "A tool is needed to allow a user to create and edit a certain class of computable processable text or graphic objects or similar structures, so that they can be used subsequently copied, printed, analyzed or used in other ways. The problem is to build a machine that can act as this tool". This problem frame includes the machine domain Editing Tool and the problem domains User, which is biddable, and the Workpieces, which is lexical. At the interface of shared phenomena we have phenomena indicating event commands (E1, E3), and other indicating symbolic phenomena (Y2 and Y4).

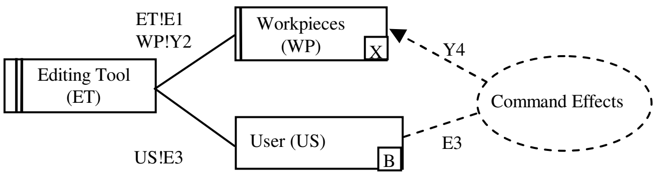


Fig. 13. Simple workpieces problem frame

The workpieces is inert domain: it may change its state in response to an externally controlled event, but initiates no state changes and no events. The events E3 are controlled by the user: they are the commands issued by the user to the tool. Some of these commands will not be obeyed. The requirement is called Command Effects: it stipulates what effects the commands E3 issued by the user to the editing tool should have on the symbolic values and state Y4 of the workpieces. The set of phenomena Y4 may have nothing in common with the set Y2, or may overlap it in any way at all. But of course both Y2 and Y4 are symbolic phenomenon of the Workpieces domain.

In Fig. 14 the corresponding model, of the basic problem frame above, is modeled using the i*.

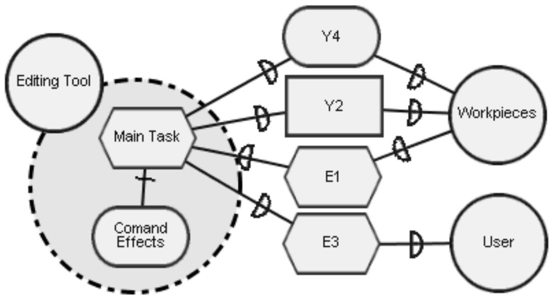


Fig. 14. Simple workpieces problem frame modeled in i*

Fig. 15 illustrates and example of the application of this basic problem frame. The requirement is Commanded Effects be controlled.

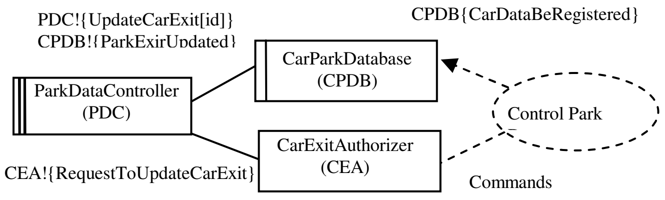


Fig. 15. Example of applying simple workpieces problem frame

In Fig. 16 the corresponding model (of this basic problem frame) is illustrated using i*.

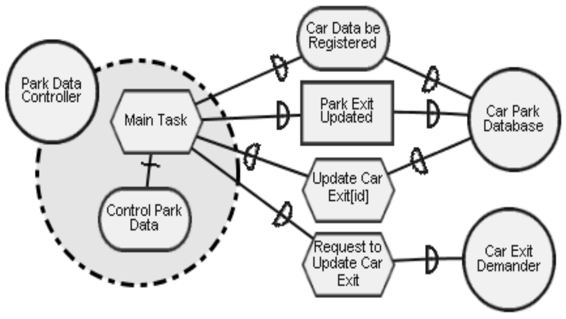


Fig. 16. Example of applying simple workpieces problem frame in i*

4.5. Transformation problem frame

In the Transformation problem frame the idea is: "There are some given computable readable input files whose data must be transformed to give a certain required output files. The output data must be in particular format and it must be derived from the input data according to certain rules. The problem is to build a machine that will produce the required outputs". The inputs and the output domains are lexical. The inputs are given; the outputs are to be made by the machine. The requirement is called de IO Relation. It stipulates a relationship between the symbolic phenomenon Y3 – values and truths and states that relate them – of the inputs domain and the symbolic phenomenon Y4 of the output domain. The relationship must be established by making the outputs appropriately. The inputs are given and cannot be changed, The machine has access to the phenomena Y1 of the inputs domain, an can determine the symbolic phenomena Y2 of the outputs domain. Y1 may or may not be the same as Y3, and Y2 may or may not be the same as Y4. The machine must lead with more elementary phenomena (such as characters), while the requirement refers to larger phenomena such as the records and fields.

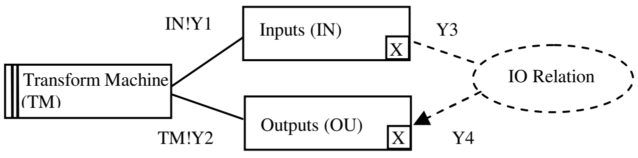


Fig. 17. Transformation problem frame

In Fig. 18 the corresponding model of this basic problem frame is presented in i*. Note that, in this case, the interaction between domains was through symbolic phenomena (Y1, Y2, Y3, Y4). So there was no event controlled by the machine. So in the transformation to i*, the main task is created and one resource is created (the INPUT), as a decomposition link, meaning that the main task requires it in order to fulfill its work, and an output resource is also created as a means-end, representing that it must be generated (as an end) in order to consider the task concluded.

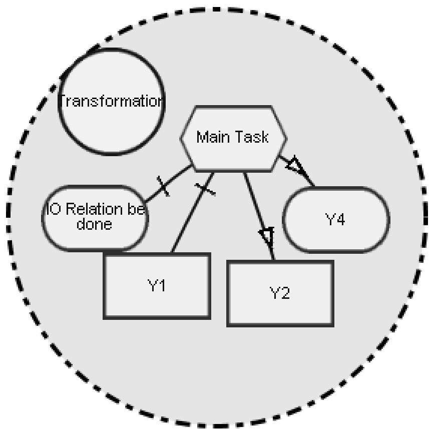


Fig. 18. Transformation problem frame in i*

Fig. 19 illustrates an example of the application of this basic problem frame. There, the machine Generator is the machine that generates information about valid payment.

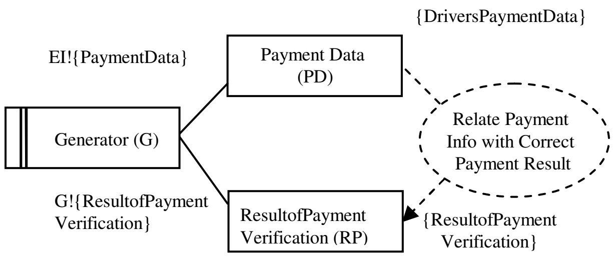

{DriversPaymentData}
EI!{PaymentData}
Payment Data (PD)
Generator (G)
Relate Payment Info with Correct Payment Result
G!{ResultofPayment Verification}
ResultofPayment Verification (RP)
{ResultofPayment Verification}

Fig. 19. Example of applying transformation problem frame

Fig. 20 illustrates the problem frame application using the car parking example, following the correspondent application in Fig. 19, using Problem Frames notation.

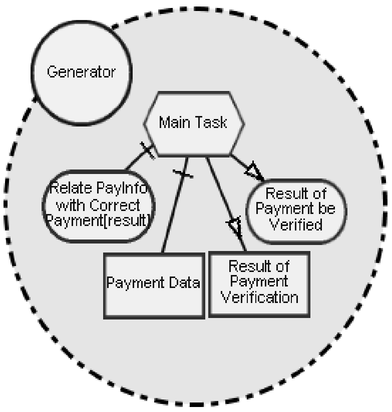

Generator
Main Task
Relate PayInfo with Correct Payment[result]
Result of Payment be Verified
Payment Data
Result of Payment Verification

Fig. 20. Example of applying transformation problem frame in i*

5. Discussions

Considering the transformations between both approaches, presented in the previous section, and the relationship proposed in [7], we can go a bit deeper in our analysis and establish a criteria for modeling a problem frame to a i* model:

1. Each machine is transformed into an expanded actor, which includes a main task, and a main goal (taken from the requirement).
2. All involved domains are transformed into actors. However, attention must be taken to lexical domains. When they include only symbolic phenomena they must be represented as resources, not actors (as exemplified in the transformation frame).
3. Specification phenomena, at the interface level, are transformed into dependencies between actors.
4. Requirement phenomena are transformed to dependencies involving goals, but only the ones which are constrained in the Problem Frame model (the ones that have an arrow).

This criterion makes it possible to follow a simple way for representing all frames. Table 1 describes the elements created in the i* model, in order to represent each of the concepts presented in a problem frame, that is, in a class of problem.

We have maintained the Problem Frames marks for phenomena (C-causal, Y-lexical, E-event), as element names in the i* models, (see Figure 2, 6, 10, 14 and18). This can be seen as a potentiality to enrich the i* semantic. However, domain types could not be expressed in the transformation.

Table 1. Correspondences created in i* to model a problem frame

Elements present in a problem frame	Correspondent element created in i*
Generic Machine	Actor and a Main task (use SR model)
Generic Requirement	Main task goal
Problem Domains (B-bidabble or C-causal)	Actors
Problem Domains (L-lexical)	Actors or resources
Specification Phenomenon	Dependum (Y→ resource, E → task, C→ goal)
Requirement Phenomenon - constrained	Dependum (Goal)
Requirement Phenomenon – not constrained	No correspondence
Domain marks (C-causal, B-bidabble, L-lexical)	No correspondence
Phenomenon marks (C-causal, E-event, L-lexical)	Dependum's name (Y→resource, E→task, C→Goal)

We can say that the representation of the basic repertoire of problem frames was possible using i*, however there are some important points which are still not solved, and so there are gaps. Some modeling steps do not have a so linear correspondence. An example of this is the main task created, in the i* models, for every frame; despite of having a well defined goal (requirement), the creation of this task was only an artifice to make it possible to represent the involved subtask, resources and goals; and so has no correspondence in Problem Frames. The causal phenomena also do not have an appropriate correspondence, so we have chosen a generic way for the conversion. The domain marks are also lost in i* modeling.

An advantage of having the basic frames in i*, is that this approach is associated with existing tools [8], [10], which help the developers in their works. Problem Frames until now does not have a specific frame for developing models. So, if stakeholders have in hand well defined classes of problems, they have a way of describing them, and so they can also apply them using i*.

Also, the possibility of representing all problem frames, that is, classes of problems in i* through a hybrid notation as presented in the problem frames modeled using i* (see Fig 2, 4 and 6, for example) induces us to argue that now we can use i* to represent other classes of problems. However, the use of patterns, in complex model, must be considered a future work, due to the involved complexity to deal with composite pattern and also many patterns in a model.

Despite of the importance of problem frames, and of the range of possibilities which can be inherited, considering the proposed modeling, problem frames seams to still require further improvements in order to be viable and useful in complex structures. Decomposition and further composition of problems, pattern application and frame variants are still an up today issue to be solved in the Problem Frame Approach.

6. Conclusions

In this paper we investigated the possibility and gains of modeling problem patterns in i*, following Michael Jackson approach. To do so we used the Jackson's repertoire of 5 basic problem frames. For each of them we proposed an equivalent model in i* and then, to consolidate and make clear each of them (basic problem frame and correspondent i* model), we made an illustrative example applying each problem frame in both approaches. This makes possible the analysis the representation of them in i* and also to elaborate some conclusions about the gains of making available such basic repertoire for i* developers.

There are some related works which explore patterns in i* and propose some architectural styles using i* approach. In [2] is offered a set of information system architectural styles which are motivated by organizational theory. Their perspective complements well, but also subsumes, proposals for multi-agent architectures. In [1] the SIRA framework is proposed to identify and map key architectural elements and the dependencies among these elements, based on the stated system requirements and organizational concepts.

In future works, we aim to make more complex and detailed examples in order to obtain a more refined analysis in the real improvements, that is, in terms of quality attributes such as: simplicity, reuse, completeness and flexibility in the built models. Other future work is to propose a more formalized way of leading with patterns in problems space considering the i*. This work is also a substep towards a detailed mapping between i* and Problem Frames and vice-versa, which in turn will contributes for the definition of a hybrid approach [2], [5], [7] which aim is to take the benefit of i* , Problem Frames and other approaches such as early aspects.

12. References

[1] L. Bastos, J. Castro, "Organizational Model to Derive Multi-Agent Architecture from Requirements", CAiSE'05 FORUM, 17th Conference on Advanced Information Systems, 2005.
[2] J. Castro, "Integração de Técnicas de Requisitos com Aspectos: O Caso TROPOS", Projeto CGCI/CAPES/MEC N.129/05, Programs of International Cooperation Brazil/Portugal, 2005.
[3] M. Jackson, Problem Frames: Analyzing and Structuring Software Development Problems, Addison-Wesley, 2001.
[4] M. Jackson, "Problem Analysis Using Small Problem Frames", South African Computer Journal 22, Special Issue on WOFACS'98, 1999, pp 47-60.
[5] M. Lencastre, J. Araújo, A. Moreira, J. Castro, "Analyzing Crosscutting in the Problem Frames Approach", In: 2nd International Workshop on Applications and Advances in Problem Frames, ICSE'06, Shangai, China, 2006.
[6] M. Lencastre, J. Botelho, P. Clericuzzi, J. Araújo, "A Meta-model for the Problem Frames Approach", WiSME: 4th Workshop in Software Model Engineering, Jamaica, 2005.
[7] M. Lencastre, F. Alencar, J. Castro, "Relating i* with the Problem Frames Approach", IDEAS'06, La Plata, Argentina, 2006.
[8] TAOM4E - Tool for Agent Oriented visual Modeling for the Eclipse Platform, http://sra.itc.it/tools/taom4e/.
[9] E. Yu, "Modeling Strategic Relationships for Processing Reengineering.", PhD Thesis in Informatics, Department of Computer Science, University of Toronto, Toronto, 1995.
[10] Yu, E. and Liu, L. (2005) "OME (Organization Modeling Environment)". Project homepage available at: http://www.cs.toronto.edu/km/ome.