

A Catalogue of Reusable Context Model Elements based on the *i** Framework

Karina Abad¹, Wilson Pérez¹, Juan Pablo Carvallo² and Xavier Franch³

¹ University of Cuenca, Cuenca, Ecuador

² University of Azuay, Cuenca, Ecuador

³ Universitat Politècnica de Catalunya, Barcelona, Spain

{karina.abadr;wilson.perez}@ucuenca.edu.ec,
jpcarvallo@uazuay.edu.ec, franch@essi.upc.edu

Abstract. The definition of system contexts is one of the most relevant activities in early phases of requirements engineering. It allows system engineers to narrow the system scope, by defining well established system boundaries. In practice, outlining a system context is cumbersome. In order to support this process, in this paper we propose a catalogue of context models elements expressed in *i**, which can be reused as building blocks in the construction of context models for new systems. We describe the process used for the identification of a set of actors and dependencies recurrently appearing in several academic and industrial cases, and the process to store them into a catalogue of reusable *i** context dependencies. The paper also illustrates possible cases of instantiation, to reuse them as patterns or context models chunks, which can be parametrized in relation to a given domain, with the support of a tool specifically designed for this purpose.

Keywords: Goal-Oriented Models, Model Reuse, iStar Framework, iStar Catalogue, Context Models

1 Introduction

The definition of system contexts is one of the most relevant activities in the early phases of requirements engineering [1]. It allows system engineers to narrow the system scope, by defining well established boundaries in relation to the actors placed in its context (organizations, people, cyber-technical systems, etc.) and the interactions (processes) that it must support to communicate with them. The definition of system context requires at least four facets to be considered [1]:

- **Use:** the kind of users that will interact with the system and their abilities and limitations (physical or mental).
- **Object:** the system-to-be and its functional coverage.
- **System:** the platform, protocols and technologies required to run and support system operation.

adfa, p. 1, 2011.

© Springer-Verlag Berlin Heidelberg 2011

Abad, K., Pérez, W., Carvallo, J.P., Franch, X. A catalogue of reusable context model elements based on the *i** framework. A: International Conference on Conceptual Modeling. "Conceptual Modeling: 36th International Conference, ER 2017: Valencia, Spain, November 6-9, 2017: proceedings". Berlin: Springer, 2017, p. 36-49.

The final authenticated version is available online at https://doi.org/10.1007/978-3-319-69904-2_3

- **Development:** internal and external standards and tools used to drive systems construction.

In practice, outlining a system context is cumbersome. It requires continuous and fluid communication among system designers, stakeholders and managers visualizing and defining business strategy, but also notations and tools required to support and document their interaction and agreements in the form of a Context Model (CM).

In order to support this process, in the last few years we have been intensively using the i^* framework [2] to model system contexts, and proposed the DHARMA [3] method to discover the system architecture departing from these models. The application of this method in a good dozen of cases led us to propose some patterns aiming at improving CM construction [4], by reusing some elements that repeatedly appeared in several of these cases. Although these patterns proved to be useful in practice [4], after conducting over 36 industrial experiences we concluded that due to the creative nature of both, managers and organizations, even in the same industry, tend to structure and behave in very dissimilar ways, making the application of large patterns highly difficult. These cases also proved that that reuse of more atomic sets of elements was not only feasible, but also a way to more efficiently construct CM [5].

In this paper, we present a catalogue of reusable CM elements expressed in i^* to be used as building blocks in the construction of CM for new systems. We describe the process used to identify a set actors and related dependencies, which frequently occurred in these cases, and the process to store them into a catalogue of reusable i^* context dependencies. Dependencies can be independently reused as atomic patterns or together, in subsets of dependencies selected in relation to labels assigned to actors. These subsets of dependencies structure larger “dynamically constructed” patterns or CM chunks, which can be parametrized in relation to the specific domain.

The rest of the paper is structured as follows. Section 2 presents the background, including the i^* framework and the DHARMA method, and some related work on pattern-based reuse. Section 3 shows the process followed for the catalogue construction. Section 4 summarizes the contents of the final catalogue. Section 5 shows how the catalogue can be used to automatize the construction of CM, starting from the identification of patterns, until their parametrization. Finally, Section 6 presents some conclusions and future work.

2 Background and Related Work

2.1 The i^* language

The i^* framework [2] was formulated for representing, modeling and reasoning about socio-technical systems. Its modeling language, which we call i^* language hereafter, is constituted by a set of graphic constructs that can be used in two models: the Strategic Dependency (SD) model, which allows representing organizational actors and their dependencies, and the Strategic Rationale (SR) model, which represents the internal actor’s rationale. Since this work makes intensive use of SD models, we focus on the explanation of their constructs (see Figure 1).

Actors in SD models represent entities with some degree of autonomy and are graphically represented by a circle. They can be related by *is-a* (sub-typing) relationships and may exhibit social dependencies. A *dependency* is a relationship between two actors, one of them, named *dependor*, who depends for the accomplishment of some internal intention from a second actor, named *dependee*. The dependency is characterized by an intentional element (*dependum*). There are four types of intentional elements (see Figure 1): resource, represented by a rectangle (e.g., Invoice); task, represented by a hexagon (e.g., Invoice Purchased) and *softgoal*, represented by a shrunk oval (e.g., Timely Payment). Goals stand for services or functional requirements, whilst softgoals represent goals whose fulfilment requires additional agreement about how they are satisfied. Softgoals are usually introduced to represent non-functional requirements and quality concerns. Resources, on the other hand, represent physical or logical elements required to satisfy a goal whilst tasks represent specific ways to achieve goals.

Very recently, a standard core for *i** named iStar2.0 has been published [6]. Differences with the notation used in this paper are minor and do not affect our results.

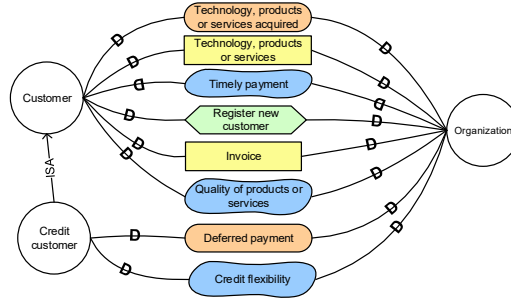


Fig. 1. Excerpt of an SD model representing the intentionality between two organizational actors.

2.2 The DHARMA Method

The DHARMA (Discovering Hybrid ARchitectures by Modelling Actors) method [3] aims at the definition of enterprise architectures using the *i** framework. This method is based on two concepts defined by Porter [7]: 1) model of market forces designed to reason about potential available strategies and how to make them profitable and helpful in the analysis of the influences of market forces; and 2) a value chain that includes primary and support activities. The process consists in four activities. Activity 1 models the enterprise context; the organization and its strategy are carefully analyzed, to identify its role inside the context, making evident the Context Actors (CA) and the Organizational Areas (OA) structuring the organization. *i** SD models are built and used to support reasoning and represent results from this activity. Activity 2 places a system-to-be into the organization and analyzes the impact that it has over the elements in the CM. Strategic dependencies identified in the previous activity (internal and context),

are inspected to determine which of them may be totally or partially satisfied by system. In Activity 3, dependencies included in the CM are analyzed and decomposed into a hierarchy of goals required to satisfy them. The goals represent the services that the system must provide, to support interaction with CA and OA activities. An *i** SR diagram for the system is built. Finally, Activity 4 is used to identify the generic architecture of the system (system actors that structure the system, the services -goals- that must be covered by each of them and the relationships among them).

2.3 Related Work on Reuse through Patterns

The reuse of requirements through patterns has been proposed and used broadly in the field of requirements engineering [8]. Most of them focus on non-functional requirements (NFR) as in [9] and [10], where a set of defined patterns is presented; these patterns pretend to capture and reuse some specific aspects linked to data security. The PABRE framework [11] makes use of patterns in order to define requirements expressed in natural language. The catalogue used by PABRE was grounded on real software requirement documents and applies to both functional requirements and NFRs.

In the *i** community, we find several approaches too. In [12] an evaluation is carried out around the application of patterns over *i** models, trying to find out if those patterns improve CM construction, finding that their application allow to define elements with a broader coverage. Nevertheless, their construction requires a deep understanding and effort; therefore, in this work it was not possible to demonstrate that their application decreases the complexity of the construction of CM. With a similar aim of exploring pattern application, in [4] we have proposed a set of patterns based in the Porter's model and some strategies, specifically the CRM strategy and we have formulated patterns for this strategy, which are formally described and are oriented to industrial applicability.

In this work we extend the works presented in [4] and [13] by adopting an approach similar to PABRE, because we pretend to create a catalogue of common elements, from which a set of patterns can be applied in order to reuse past knowledge. With respect to [13], we present a catalogue of CM elements, we describe the process to construct such catalogue, we introduce and show the use of parametric actors and dependencies, we extend the formalization to include the parameters and we show a case of application through the use of semantic technologies. Also, our work is intended to provide guidance in early phases of the Enterprise Engineering process, providing artifacts to bridge communication gaps among technical Enterprise Engineering staff and administrative staff. With respect to PABRE, the main difference is that our work will be expressed using the *i** notation instead of in natural language (which provides an adequate level of abstraction for modeling CM) and also we want to base our catalogue in semantic technologies, which according to [14] are not very exploited in that field. The literature review presented in [14] describes how ontologies have been used in the different requirements engineering phases and how they are used. The results of the study showed that many approaches exist in the different stages of the requirements engineering, but only a few of them related to the model type, specifically the *i** model.

3 Catalogue Construction

In this section we present the process performed to construct the catalogue of CM elements. The process starts with the data collection, then we analyze separately actors and dependencies. Due to the large number of CM elements, each analysis was performed in many steps, as explained below.

3.1 Data collection

The models upon which we based this analysis were constructed by university students in their final grade project, acting as junior consultants in 36 companies. These students were trained in the construction of CM, specifically in the i^* notation and the DHARMA method, according to the scope, objectives and activities proposed in such method. The models were created for the organizations through formal agreements among them and the University of Cuenca. In the study, 27 of the enterprises were small sized, 6 medium sized and 3 large sized. This distribution largely corresponds with the reality of the country where the studies were conducted, whose industrial net-work is composed by small companies as majority (97,94%) [15]. The organizations were classified according to NACE Rev2's domains [16] identifying: 12 manufacturing, 16 wholesales, 2 human health, 4 education, 1 transportation and 1 financial.

In each organization, the modelling process started with the construction of CM and finished with the identification of the system architecture required to support its operation. The junior consultants worked in pairs in order to complete each DHARMA activity. To perform Activity 1, an interview with the interlocutor assigned by each organization was conducted; its objective was to get information enough as to allow the identification of actors inside and outside the organization, and to discover the relations among them, in other words, their social dependencies. The junior consultants performed the process manually and the final product of this activity was a set of i^* diagrams and its tabular representation according to DHARMA. Each group was able to find around 31 actors and 58 dependencies per organization in average, with a maximum of 50 actors and 113 dependencies and a minimum of 17 actors and 20 dependencies.

3.2 Actor Identification

The actor analysis started by considering two generic groups as defined by Porter [7], namely: 8 external context actors (*Suppliers, Consumers, Strategic Partners, Distributors, Financial Institutions, Regulatory Agencies, Control Agencies and Competitors*) and 9 internal context actors (*Inbound Logistics, Operations, Outbound Logistics, Marketing and Sales, Services, Infrastructure, Human Resources Management, Technology Development and Procurement*). To define the catalogue of actors extending this initial group, we followed a three-step process, described below.

First, the CM of the 36 organizations were integrated into a single data space, and the actors of each organization were classified according to the 8 external context actors and the 9 internal context actors enumerated above (see results in Table 1).

Table 1. Actors identified per organization. Size organization (S – Small, M – Medium and L - Large). Domain classification (M - Manufacturing, E - Education, T - Transportation, H - Human Health, W - Wholesale and F - Financial activities)

| Organization | | External Context Actor | | | | | | | | | | Internal Context Actor | | | | | | | | | |
|--------------|------|------------------------|-----------|-----------|------------------|---------------------|--------------|--------------------|-------------|------------------------|-------|------------------------|--------------------|------------|----------|---------------------|---------------------------|-------------|----------------|---------------------|-------|
| Id | Size | Domain | Suppliers | Consumers | Control Agencies | Regulatory Agencies | Distributors | Strategic Partners | Competitors | Financial Institutions | Total | Inbound logistics | Outbound logistics | Operations | Services | Firm Infrastructure | Technological development | Procurement | Human Resource | Marketing and sales | Total |
| | | | | | | | | | | | | | | | | | | | | | |
| Org1 | S | M | 9 | 3 | 7 | 3 | 0 | 0 | 0 | 0 | 22 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 7 |
| Org2 | S | M | 5 | 3 | 7 | 3 | 1 | 2 | 3 | 0 | 24 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 2 | 6 |
| Org3 | S | M | 3 | 5 | 7 | 5 | 2 | 3 | 0 | 1 | 26 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 4 |
| Org4 | S | M | 8 | 5 | 6 | 3 | 2 | 1 | 0 | 1 | 26 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 2 | 8 |
| Org5 | S | W | 6 | 7 | 2 | 3 | 1 | 2 | 1 | 1 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Org6 | S | W | 5 | 6 | 2 | 2 | 1 | 1 | 0 | 1 | 18 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 5 |
| Org7 | S | W | 4 | 7 | 3 | 2 | 1 | 0 | 1 | 0 | 18 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 5 |
| Org8 | S | E | 4 | 4 | 3 | 2 | 1 | 2 | 0 | 1 | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Org9 | S | W | 6 | 7 | 4 | 3 | 2 | 1 | 2 | 2 | 27 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2 | 7 |
| Org10 | S | W | 3 | 5 | 4 | 4 | 0 | 0 | 0 | 0 | 16 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 4 |
| Org11 | S | W | 10 | 12 | 7 | 3 | 4 | 3 | 2 | 3 | 44 | 1 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 6 |
| Org12 | S | W | 4 | 3 | 2 | 1 | 2 | 0 | 0 | 1 | 13 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 2 | 7 |
| Org13 | S | M | 3 | 6 | 5 | 2 | 3 | 0 | 3 | 2 | 24 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 4 |
| Org14 | S | W | 8 | 5 | 6 | 3 | 2 | 1 | 0 | 1 | 26 | 1 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 1 | 5 |
| Org15 | M | W | 8 | 9 | 7 | 3 | 3 | 4 | 2 | 2 | 38 | 1 | 1 | 0 | 0 | 2 | 0 | 1 | 1 | 3 | 9 |
| Org16 | M | W | 7 | 5 | 6 | 3 | 2 | 1 | 0 | 0 | 24 | 2 | 1 | 0 | 0 | 2 | 0 | 2 | 0 | 3 | 10 |
| Org17 | S | W | 7 | 5 | 3 | 4 | 2 | 2 | 1 | 1 | 25 | 0 | 0 | 1 | 0 | 2 | 0 | 2 | 1 | 1 | 7 |
| Org18 | S | M | 5 | 7 | 2 | 2 | 1 | 1 | 1 | 1 | 20 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 6 |
| Org19 | S | W | 9 | 3 | 7 | 3 | 0 | 0 | 0 | 0 | 22 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 3 |
| Org20 | S | W | 8 | 5 | 6 | 3 | 2 | 1 | 0 | 1 | 26 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 2 | 7 |
| Org21 | M | E | 9 | 8 | 6 | 3 | 4 | 4 | 2 | 2 | 38 | 0 | 0 | 0 | 0 | 2 | 3 | 0 | 0 | 0 | 5 |
| Org22 | S | F | 8 | 3 | 7 | 3 | 0 | 0 | 0 | 1 | 22 | 0 | 0 | 1 | 0 | 3 | 1 | 0 | 1 | 2 | 8 |
| Org23 | S | W | 6 | 5 | 5 | 2 | 3 | 0 | 3 | 1 | 25 | 1 | 0 | 2 | 0 | 1 | 1 | 0 | 1 | 1 | 7 |
| Org24 | S | W | 8 | 3 | 7 | 3 | 0 | 0 | 0 | 0 | 21 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 3 |
| Org25 | S | M | 7 | 5 | 6 | 3 | 3 | 4 | 2 | 2 | 32 | 1 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 6 |
| Org26 | S | E | 5 | 6 | 2 | 3 | 1 | 2 | 1 | 1 | 21 | 0 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 2 | 6 |
| Org27 | L | M | 7 | 7 | 5 | 3 | 4 | 4 | 2 | 1 | 33 | 1 | 1 | 1 | 0 | 4 | 0 | 1 | 1 | 1 | 10 |
| Org28 | S | E | 9 | 8 | 6 | 3 | 4 | 4 | 2 | 2 | 38 | 0 | 0 | 3 | 0 | 1 | 0 | 0 | 1 | 0 | 5 |
| Org29 | S | M | 6 | 6 | 2 | 2 | 1 | 0 | 1 | 0 | 18 | 1 | 1 | 8 | 0 | 2 | 0 | 0 | 1 | 1 | 14 |
| Org30 | S | M | 8 | 5 | 6 | 3 | 2 | 1 | 0 | 1 | 26 | 1 | 1 | 2 | 0 | 1 | 0 | 0 | 1 | 2 | 8 |
| Org31 | S | W | 5 | 4 | 2 | 1 | 2 | 0 | 1 | 1 | 16 | 1 | 0 | 0 | 0 | 4 | 1 | 0 | 1 | 0 | 7 |
| Org32 | M | M | 8 | 3 | 7 | 3 | 0 | 0 | 0 | 0 | 21 | 1 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 4 |
| Org33 | M | H | 5 | 4 | 2 | 1 | 2 | 0 | 0 | 0 | 14 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 5 |
| Org34 | M | H | 7 | 7 | 5 | 3 | 4 | 4 | 2 | 1 | 33 | 1 | 0 | 2 | 1 | 2 | 1 | 1 | 1 | 4 | 13 |
| Org35 | L | M | 6 | 7 | 6 | 3 | 2 | 1 | 1 | 0 | 26 | 2 | 0 | 2 | 0 | 3 | 1 | 1 | 1 | 2 | 12 |
| Org36 | L | T | 8 | 5 | 6 | 3 | 0 | 0 | 1 | 0 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total | | | 234 | 198 | 176 | 99 | 64 | 49 | 34 | 32 | 886 | 28 | 14 | 42 | 5 | 46 | 10 | 10 | 21 | 47 | 223 |

A total of 1,109 actors were found, from which 886 are external and 223 are internal. The most common types of actors are *Suppliers* and *Customers* with a 38.89% appearance rate. On the other hand, the least common actor is *Services*, included only in 5 organizational models.

Second, we proceeded to unify all the actors that appeared duplicated in the models, so that we obtained a set of instances for each internal and external actor. Table 2 shows an excerpt of actors identified as *Supplier* instances and their occurrences in the 36

organizations. We can see instances like *Basic services supplier*, *Technology supplier*, etc. Summarizing, from the 886 external actors, only 203 are unique instances and from the 223 internal actors, only 77 are unique instances.

Third, after getting the instances, we realized that some of them had common characteristics, being able to group them into categories, obtaining a hierarchical structure; the categories are called dimensions and are composed by the specific instances found in the previous step. As an example, consider actors categorized under the *Supplier* generic actor, which defines three dimensions (see Table 3): *Type* (Goods, Hardware, Basic services, Technology, etc.); *Volume* (Wholesaler and Retailer); and *Location* (Local, International and National).

Table 2. Excerpt of actors identified and their occurrence in the 36 cases conducted

| Generic Actor | Actor | Org1 | Org2 | Org3 | Org4 | Org5 | Org6 | Org7 | Org8 | Org9 | Org10 | Org11 | Org12 | Org13 | Org14 | Org15 | Org16 | Org17 | Org18 | Org19 | Org20 | Org21 | Org22 | Org23 | Org24 | Org25 | Org26 | Org27 | Org28 | Org29 | Org30 | Org31 | Org32 | Org33 | Org34 | Org35 | Org36 | Total | Percentage | |
|---------------|----------------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------------|-----|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Supplier | Hardware | | | | | | | | | | | | | | | | | | | X | | | | | | | | | X | | | | | | | | | 3 | 8% | |
| | Basic services | | | | | | | | | X | X | X | X | | X | | X | X | | | | X | X | X | | | | | | | | | | | | | X | 14 | 39% | |
| | Technology | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 3% |
| | Transport | | | | | | | | X | | | X | | | | X | | | | | | | | | | | | | | | | | | | | | | | 3 | 8% |
| | Wholesale | | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | | 1 | 3% |
| | Local | X | | | X | X | | | | X | | X | | X | X | | | | | | X | | | | | | | | | | | | | | | | | | 7 | 19% |
| | National | | | | X | X | | X | | X | | | | | X | X | | | | | X | | | | | | | | | | | X | | | | X | X | 9 | 25% | |
| | International | | | | | | | X | X | | | | | | | X | | X | | | X | | | | | | | | | | | | | | | X | | | 7 | 19% |

Table 3. Dimensions found for the *Supplier* generic actor.

| Generic actor | Dimension | Actor Instances |
|---------------|-----------|-----------------|
| Supplier | Type | Goods |
| | | Hardware |
| | | Basic services |
| | | Technology |
| | | |
| | | Transport |
| | Volume | Wholesaler |
| | | Retailer |
| | Location | Local |
| | | International |
| | | National |

3.3 Dependency Identification

The dependency analysis was focused on their description and type in order to define a catalogue of dependencies. This process was defined in two steps, as described below.

First, similarly to actors' identification, the CM corresponding to the 36 organizations were integrated into a single data space and the dependencies of each organization

were analyzed based on their type. Results are shown in Table 4. A total of 2,095 dependencies were found (1351 for dependencies related with external actors and 744 for dependencies related with internal actors); from them, 862 dependencies are goals, 537 are softgoals, 619 are resources and 77 are tasks. The scarce use of task dependencies is probably due to their high level of prescriptiveness, which is something that does not match well with the activity of context modeling.

Second, we proceeded to group the 2,095 dependencies with respect to the generic actors that they were connecting. In this step, the duplicated dependencies were omitted, finding 994 dependencies (408 dependencies corresponding to external actors and 586 to internal actors). Summing up, from the 862 goal dependencies identified in the previous step (table 4), 449 are unique instances; from the 537 softgoal dependencies, 249 are unique instances; from the 619 resource dependencies, 254 are unique instances; and from the 77 task dependencies, 42 are unique instances.

Table 4. Dependencies identified per organization

| Type | Org1 | Org2 | Org3 | Org4 | Org5 | Org6 | Org7 | Org8 | Org9 | Org10 | Org11 | Org12 | Org13 | Org14 | Org15 | Org16 | Org17 | Org18 | Org19 | Org20 | Org21 | Org22 | Org23 | Org24 | Org25 | Org26 | Org27 | Org28 | Org29 | Org30 | Org31 | Org32 | Org33 | Org34 | Org35 | Org36 | Total |
|----------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Goal | 16 | 31 | 22 | 36 | 15 | 39 | 26 | 19 | 39 | 14 | 29 | 25 | 15 | 29 | 29 | 32 | 16 | 14 | 16 | 9 | 40 | 30 | 28 | 29 | 17 | 24 | 46 | 20 | 30 | 28 | 13 | 16 | 10 | 9 | 15 | 36 | 862 |
| Softgoal | 11 | 10 | 9 | 19 | 11 | 12 | 2 | 4 | 18 | 6 | 15 | 22 | 8 | 19 | 23 | 15 | 10 | 21 | 11 | 9 | 19 | 18 | 21 | 17 | 13 | 14 | 32 | 22 | 23 | 33 | 4 | 7 | 7 | 10 | 22 | 20 | 537 |
| Resource | 18 | 12 | 0 | 26 | 10 | 10 | 2 | 13 | 22 | 5 | 15 | 18 | 14 | 25 | 22 | 32 | 16 | 24 | 13 | 11 | 21 | 11 | 24 | 39 | 15 | 27 | 32 | 12 | 21 | 35 | 8 | 5 | 3 | 14 | 26 | 18 | 619 |
| Task | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 7 | 2 | 0 | 2 | 4 | 2 | 4 | 6 | 0 | 0 | 1 | 2 | 1 | 4 | 3 | 1 | 1 | 7 | 3 | 1 | 7 | 8 | 1 | 0 | 0 | 0 | 5 | 2 | 77 |
| Total | 45 | 53 | 31 | 82 | 36 | 63 | 30 | 36 | 86 | 27 | 59 | 67 | 41 | 75 | 78 | 85 | 42 | 59 | 41 | 31 | 81 | 63 | 76 | 86 | 46 | 72 | 113 | 55 | 81 | 104 | 26 | 28 | 20 | 33 | 68 | 76 | 2095 |

3.4 Synonyms in Actors and Dependencies

During the data analysis we found some actors and dependencies that represented the same entity or idea but written differently, that is, synonyms. An example is the occurrence of two actors *Final client* and *Final customer* in two different organizations; for dependencies, consider the dependency *Timely delivery* found in one organization and *On-time delivery* found in another one. The total number of synonyms found were 13 for external actors, 51 for dependencies related with external actors, 32 for internal actors and 86 dependencies related with internal actors. For dependencies, 68 were goals, 48 were softgoals and 21 were resources. To simplify the catalogue, we decided to create a section focused in those findings. The main idea is that in later stages, through the use of semantic technologies, analyze the actors and dependencies entered by the user as part of a CM and inform him that a similar actor or dependency has been previously defined with different words (if that is the case), and let him decide which of them is the best option.

Fig. 2 shows graphically how the total number of actors and dependencies shrank little by little after of each step describe in sections 3.2 and 3.3, including the result of synonyms, obtaining a total of 235 actors, from them 190 are external context actors and 45 internal context actors. Additionally, 857 dependencies were identified, where 381 are goal, 201 are softgoal, 233 are resources and 42 are task dependencies.

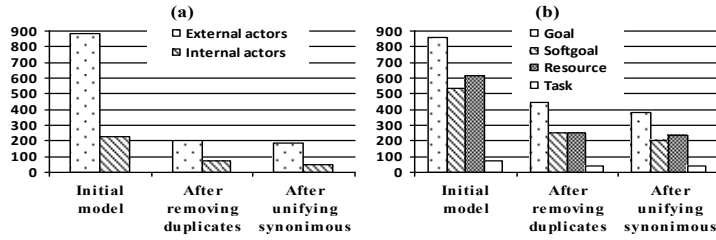


Fig. 2. Number of elements shrunk on each step. (a) Actors, (b) Dependencies

3.5 Parametric Actors and Dependencies

Even after the consolidation of actors and dependencies as explained in the previous subsections, we found groups of actors or dependencies identified in different organizations but all those sharing similar characteristics. To make explicit this similarity and also to make the catalogue more compact, we incorporated parameters to the definition of actors and dependencies. As an example of parametric actors, consider an organization where the actor *Primary Students* has been identified, and another one where the actor *High school Students* emerged, both of them with similar relations in their respective organization. That allowed us to group them in one category, with the possibility to parametrize them, that is, the parametric actor is defined as *<type-of-student> Students*, where the parameter *<type-of-student>* can be instantiated as *Primary* or *High school*. Other possible case of parametrization can be found with actors sharing characteristics as the sector, but differing in the industry, it means, the parametric actor could be *Supplier of <services>* and the tag *<services>* can be parametrized as *basic services*, *telecommunications*, *security*, etc.

The same occurs in dependencies. For instance, let's consider the dependency *<Products> acquired*; the parameter *<Products>* can be replaced by *furniture*, *clothes*, *equipment*, etc., according to the industry of the depender or dependee. Sometimes, parametric dependencies can be associated to parametric actors, for example, the parametric actor *Supplier of <services>* (mentioned in the paragraph above) is associated to the parametric dependency *<Specific documents>*, if the actor is instantiated as *Supplier of transport services*, the dependency has to be instantiated as *Waybill* or if the actor is *Supplier of medical services*, the dependency has to be instantiated as *Medical record*.

A total of 41 actors in the catalogue were identified as parametric (22 external and 19 internal) and also 63 dependencies (35 dependencies related with external actors and 28 dependencies related with internal actors), where 25 are goal dependencies, 6 are softgoal dependencies, 28 are resource dependencies and 4 are task dependencies.

4 The Catalogue of Context Model Elements

After organizing the data and identifying unique instances, parameters and synonyms of actors and dependencies, we provided structure to the catalogue. It was organized into two sections, one for actors and other for dependencies. The actors' section has a total of 235 instances, from them 190 are external context actors and 45 internal context actors, structured in 4 hierarchical levels as explained below:

- **First level:** composed by the 17 Porter generic context actors, 8 external and 9 internal, introduced in Section 3.2.
- **Second level:** Each internal and external actor is decomposed into subactors (defined as "Dimensions" in Table 3). From them, 17 are dimensions of external context actors, and 7 are dimensions of internal context actors.
- **Third level:** Contains a total of 39 instances on external context actors and 9 instances on internal context actors (see column "Actor instances" in Table 3).
- **Fourth level:** This level contains the parameters that can be used as instances of parametric actors defined in the third level and has 190 external context actors and 45 internal context actors.

From the point of view of the dependencies, the catalogue contains a total of 857 dependencies, from which 381 are goals (126 dependencies linked to external actors and 255 to internal actors); 201 are soft goals (107 dependencies linked to external actors and 94 to internal actors); 233 resources (109 dependencies linked to external actors and 124 to internal actors); and 42 tasks (15 dependencies linked to external actors and 27 to internal actors).

The catalogue (currently in Spanish only) can be accessed from a URL address¹. Table 5 shows an excerpt of the catalog, presenting the dimension, actors and dependencies identified for the *Customer* and *Supplier* actors.

5 Catalogue Use

In this section we provide further details related to the application of the catalogue. As argued in this paper, the use of patterns will improve the construction of CM; the idea of its use is to instantiate the required patterns that may appear when creating a CM, using the catalogue of elements constructed before as instances that may or not be added to the model, according to each instantiation case. We start this section introducing some of the patterns found in [4], and defining other new ones (Section 5.1). The manual application of such patterns is kind of difficult, therefore some tool support becomes necessary. In this work, we propose a tool based in a semantic model in order to provide knowledge management capabilities to our catalogue. Section 5.2 explains the semantic model implemented to support the catalogue and Section 5.3 shows the tool constructed to apply the patterns in real cases.

¹https://www.dropbox.com/sh/7jnsv7vqwwhmv8/AADqaiHx_vDj-gi6mk_5pg_Ca?dl=0

Table 5. Catalogue excerpt

| Generic actor | Dimension | Actor Instances | Dependency | Type | Direction |
|---------------|----------------------|-------------------------|---|----------|-----------|
| Customer | Frequency or Volume | Potencial | Widespread promotions | Goal | > |
| | | | Promocional samples | Resource | < |
| | | New | Membership card provided | Goal | > |
| | | | Special introduction prices provided | Softgoal | > |
| | | | Membership card | Resource | > |
| | | | Personal information registered | Goal | < |
| | | Important | VIP benefits granted | Goal | > |
| | | | Personalized attention | Softgoal | > |
| | | | VIP card | Resource | > |
| | | | Important high volume order placed | Goal | < |
| | Distribution channel | Wholesaler | Product availability guaranteed | Goal | < |
| | | | Product distribution agreement signed | Softgoal | < |
| | | | Increase sales through the distribution chain | Softgoal | < |
| | | | Product distribution agreement | Resource | < |
| | | | Product distribution chain achieved | Softgoal | > |
| | | Retailer | Restocking in small quantities provided | Goal | > |
| | | | Approach consumers through an specific location | Softgoal | < |
| | | | Increase sales through individual stores | Softgoal | < |
| | | Specific market segment | Specialized customer service infrastructure | Softgoal | > |
| | | | Trained stuff for specific needs | Softgoal | > |
| | | | Specific documents | Resource | > |
| | | | | | |
| | Payment method | Credit | Deferred payments | Goal | > |
| | | | Credit flexibility | Softgoal | > |
| | | | Acceptance of various credit cards | Softgoal | > |
| | | | Voucher | Resource | > |
| | | | Warranty documents | Resource | < |
| | | Cash | Cash rebates | Goal | > |
| | | | Money | Resource | < |
| | | | | | |
| Supplier | Type | Goods | Health service obtained | Goal | > |
| | | Services | Home service | Goal | > |
| | Volume | Wholesaler | Large purchase orders | Goal | > |
| | | Retailer | Restocking in small quantities provided | Goal | < |
| | Location | Local | Cash payment discounts | Goal | > |
| | | National | Deferred payments | Goal | > |
| | | International | | ... | ... |

5.1 Analysis of patterns

With the aim of analyzing the context of an organization, in [4] we identified 8 different pattern instantiation cases, in relation to actors and dependencies which can be applied to our catalogue. They are described in Fig. 3. The first six cases are in fact two pairs of similar cases for actors and dependencies: one-to-one instantiation (one pattern actor

or dependency is replaced by one element of the same kind in the context of the organization); one-to-many instantiation (one pattern actor or dependency is replaced by several elements of the same kind); null instantiation (one pattern actor or dependency is not instantiated in the organization). In addition, two dependency-only patterns were identified: split instantiation (in which the dependencies associated to one actor in the pattern are split into groups, each one associated to a different actor in the context of the organization) and group instantiation (all the dependencies associated to one actor in the pattern are associated to a single instance of the actor in the context of the organization).

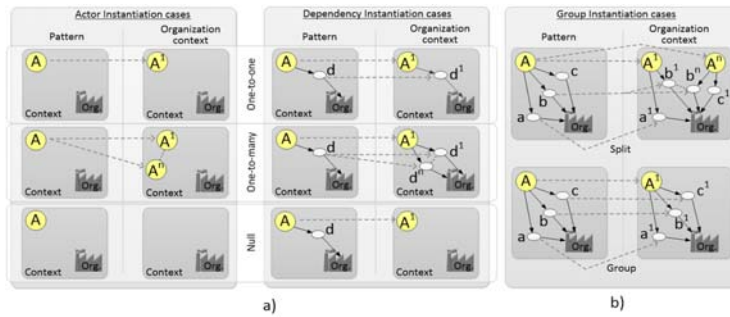


Fig. 3. a) Actor and dependency instantiation cases; b) dependency group instantiation cases

After analyzing these instantiation cases of actor and dependency over our catalogue, based in the definition and patterns mentioned above, we identified one additional case referred to parametric dependencies:

1. *One or many parametric dependency instantiations.* - A parametric dependency associated to one actor in the pattern is instantiated by one or many dependencies associated to an instance of the actor in the context of the organization. In this case, the parametric dependency and the instances have same contribution with the actors involved in the relationship (see Fig. 4).

Although the use of patterns to instantiate and create CM helps in knowledge reuse, it is evident that their application is not an easy task, mainly due to the increasing number of dependencies existing in organizational CM, and second because it is difficult for consultants to identify and familiarize with parametric dependencies, and also control synonymy cases, even more when working in groups. To solve those problems and help in the construction of CM, we found interesting to automatize the process, making use of a semantic model (an ontology) which allows us to identify those synonymy cases. In the next section we present the semantic model in which our work is based.

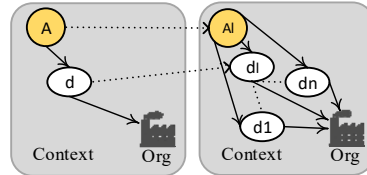


Fig. 4. Parametric dependency instantiation cases

5.2 The DHARMA Ontology

To reuse the catalogue of SD elements, we found necessary to have a semantic knowledge base aiming at making inferences and recommendations about the elements in the catalogue. We created the DHARMA ontology [17], which conceptualizes the knowledge of the DHARMA method in an ontological network (see Fig. 5).

This ontology allows us to store concepts of actors and dependencies, such as 1) its environment (internal or external, according to Porter), 2) features of the parametric elements, such as if the element is a parameter, which are its instances and 3) define other characteristics of DHARMA – type of actor (hardware, software, person, organization), coverage of the dependencies (total, partial), etc.

The DHARMA ontology was created parting from four ontologies of different domains, linked between them to create a network ontology, which includes:

- The *Offer-job* and *Classification* ontologies [18], to cover concepts of size, and domain of an organization.
- The *OntoiStar+* ontology [19], which contains concepts the *i** framework.
- The *ValueChain* [17] ontology, developed by the authors to include organizational areas,

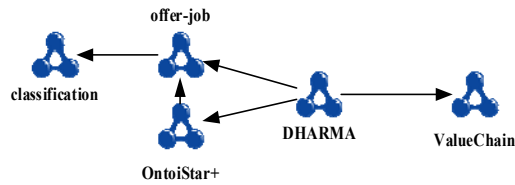


Fig. 5. DHARMA ontology network

In this paper, we have changed our first version of the ontology [17] in order to improve:

- *Inference capabilities*: We have defined some inference rules to improve the recommendation of the dependencies, analyzing the impact that the dependencies associated to an actor in a lower level will have over its hierarchical parent.

- *Synonym management*: This point, still under development, is aimed to extend the ontology and feed it with the synonyms data, and even extend it to include some ontologies that will contribute to that process.

With all the knowledge stored in the ontology and its capabilities of inference, our tool support (see Section 5.3) will be able to recommend the right dependencies, applying the patterns listed in the previous section, and also to resolve duplicity of CM elements which appears when the number of elements in CM increase considerably.

5.3 Tool support for the use of the catalogue

The last step in the application of the catalogue is the construction of CM in an automated way. To this purpose, we created Semantic DHARMA [20] (S-DHARMA), a web application that allows the creation and maintenance of the catalogue elements, as well as the creation of CM applying the DHARMA method and its four activities. S-DHARMA uses the DHARMA ontology as basis to describe the elements of the catalogue, which were migrated to a triplet store, specifically Apache Marmotta², where the resulting repository is a semantic abstraction of catalogue.

From our first version of the system in [20], we have added the application of instantiation patterns as defined in Section 5.1 and we have systematized the process of creating a new CM:

1. *Define organization*. The characteristics of the organization, like name, size, sector and industry, are defined.
2. *Select actors*. The catalogue is rendered to the user in a hierarchical structure, as described in Section 4. The selection of one or more actors in any level implies the use of the one-to-one or one-to-many instantiation patterns, while the actors that have not been selected are part of the null actor pattern.
3. *Select dependencies*. Once the actors are selected, the tool recommends some dependencies retrieved from the repository and related to actors selected in the previous step. The dependency-related patterns one-to-one, one-to-many, split instantiation and null are applied according to the characteristics of the selected dependencies stored in the triple store. Also, if among the selected elements, there exists a group where all of them are linked to an actor in the pattern, then the group instantiation pattern may be used.
4. *Show model*. The tool renders the tabular representation of the CM (see Fig. 6) which looks similar to the data presented in Table 3. This is the starting model that can be refined afterwards.
5. *Refine model*. This initial model can be refined by updating existing elements and even creating new ones. The available properties to create or update an actor are: its position with respect the context (internal or external) and its name. In the same way, to create or update dependencies, the user can choose the dependency type (goal, resource, task or softgoal), and its direction (right, left or bidirectional). When a par-

² <http://marmotta.apache.org/>

ametric actor or dependency is instantiated, its parameter's abstraction level is "specific"; in the same way, if a new actor or dependency is added to the CM, its type will be "specific".

In both steps 2 and 3, the particular case of parameterized elements needs to be considered. When a user needs to parameterize an element of the catalogue, the tool shows a list of possible elements with similar characteristics, allowing the user to select the parametric elements, depending on the parameters assigned to an actor or dependency, this option will correspond to the one or many parametric dependency instantiations case.








| ACTOR 1 | TYPE | DIRECTION | DEPENDUM | ACTOR 2 |
|-----------------------------|---|-----------|---------------------|-------------------|
| CREDIT CUSTOMER |  | → | DEFERRED PAYMENTS | SALES |
| CREDIT CUSTOMER |  | → | VOUCHER | SALES |
| SERVICES CUSTOMER |  | ← | HOME SERVICE | SERVICE |
| TRANSPORT SERVICES SUPPLIER |  | ← | WAYBILL | INBOUND LOGISTICS |
| CUSTOMER |  | → | SERVICES OF QUALITY | SALES |
| CUSTOMER |  | → | INVOICE | SALES |
| CREDIT CUSTOMER |  | → | CREDIT FLEXIBLY | SALES |

Fig. 6. Screenshot of the tabular representation of CM in the Semantic DHARMA application

6 Conclusions

In this paper we have presented the study performed in over 36 industrial IS architectural in which junior consultants use the *i** language, in particular Strategic Diagrams (SD), to build context models. The models were analyzed, classified and organized in order to get a catalogue of reusable context model elements to serve as a basis for the construction of *i** SD-based CM in future case studies. The results obtained in this study are important; the catalogue contains a total of 235 actors (190 external context actors and 45 internal context actor) and 857 dependencies (381 are dependencies of type goal, 201 are dependencies of type softgoal, 233 dependencies of type resource, and 42 dependencies of type task). Additionally, the patterns presented in [4] were validated in this study, and based on the concept of parametric elements we have obtained new definition and the extension of patterns to parametric instantiation. Also, a tool for the application of the catalogue and the parameters was introduced, which uses semantic technologies to store the catalogue; the use of the tool will support the construction of CM avoiding synonyms in the dependencies.

Work in progress includes the further refinement of the existing catalogue. Besides, we aim to improve the S-DHARMA ontology; our goal is the modeling of new context model to validate the catalogue and its refinement with new knowledge to discover. Last, we plan to conduct a similar study focused on *i** SR models, which are used in later phases of the DHARMA method, analyzing goal decomposition, means-end links, etc.

References

1. Pohl, K. Requirements Engineering: Fundamentals, Principles, and Techniques, Springer, Berlin, Heidelberg, 2010;
2. Yu, E. Modelling Strategic Relationships for Process Reengineering. Ph.D. Thesis, University of Toronto, Department of Computer Science, Canada, 1995.
3. Carvallo, J. P., Franch, X. Descubriendo la arquitectura de sistemas de software híbridos: un enfoque basado en Modelos *i**. WER 2009.
4. Carvallo, J. P., Franch, X. Building Strategic Enterprise Context Models with *i**: A Pattern-Based Approach. TEAR 2012.
5. Abad, K., Pérez, W., Carvallo, J. P., Franch, X. *i** in Practice: Identifying Frequent Problems in its Application. ACM SAC 2017.
6. Dalpiaz, F., Franch, X., Horkoff, J.: *iStar 2.0 Language Guide*. CoRR. 2016. <https://arxiv.org/abs/1605.07767>.
7. Porter, M. Competitive Strategy. Free Press, New York, NY, United States, 1980.
8. Withall, S. J., Software Requirement Patterns. Microsoft Press, 2007.
9. Supakkul, S., Hill, T., Chung, L. An NFR Pattern Approach to Dealing with NFR. RE 2010.
10. Ruiz-López, T., Garrido, J., Supakkul, S., Chung, L. A Pattern Approach to Dealing with NFRs in Ubiquitous Systems. CEUR-WS 2013.
11. Renault, S., Méndez, O., Franch, X., Quer, C.: Constructing and Using Software Requirement Patterns. RCIS 2009
12. Strohmaier, M., Horkoff, J., Yu, E., Aranda, J., Easterbrook, S.: Can Patterns improve *i** Modeling? Two Exploratory Studies. REFSQ 2008.
13. Abad, K., Carvallo, J. P., Peña, C. *iStar* in Practice: On the identification of reusable SD Context Models Elements. *iStar* 2015.
14. Dermeval, D., Vilela, J., Bittencourt, I., Castro, J., Isotani, S., Brito, P., Silva, A. Applications of ontologies in requirements engineering: a systematic review of the literature. RE 2016.
15. Instituto Nacional de Estadísticas y Censos. Ecuador en cifras. <http://aplicaciones2.ecuadorencifras.gob.ec/dashboard2/pagina3.php>.
16. Office for Official Publications of the European Communities. NACE Rev 2. Statistical classification of economic activities in the European Community. Luxembourg: Eurostat Methodologies and Working papers. 2008.
17. Abad, K., Carvallo, J. P., Espinoza, M., Saquicela, V., Hacia la Creación de un Repositorio Semántico de Modelos de Contexto Basados en *i** y el método DHARMA. RISTI 2016.
18. Villazón-Terrazas, B.; Ramírez, J.; Gómez-Pérez A., Human Resources Management Ontology, <http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/99-hrmonontology/index.html>
19. Najera, K., An ontology-based approach for integrating *i** variants. Master thesis, National Center of Research and Technological Development, Mexico, 2011.
20. Abad, K., Pérez W., Carvallo, J. P. Managing *i**-based Reusable Context Models Elements through a Semantic Repository. *iStar* 2016.