*i** in Practice: Identifying Frequent Problems in its Application

Karina Abad
Universidad de Cuenca
Av. 12 de Abril s/n
Cuenca, Ecuador
+593984210041
karina.abadr@ucuenca.edu.ec

Wilson Pérez
Universidad de Cuenca
Av. 12 de Abril s/n
Cuenca, Ecuador
+593984329344
wilson.perez@ucuenca.edu.ec

Juan Pablo Carvallo
Universidad del Azuay
Av. 24 de Mayo 7-77
y Hernán Malo,
Cuenca, Ecuador
+593990447559
jpcarvallo@uazuay.edu.ec

Xavier Franch
Universitat Politécnica de Catalunya
Jordi Girona Salgado 1-3 E-08034
Barcelona, Spain
+ 3493413 7891
franch@essi.upc.edu

ABSTRACT
Several notations have been proposed in the last decades to support information system architecting, design and implementation. Although some of them have been widely adopted, their practical application remains cumbersome. Reasons are manifold: ambiguous semantics, confusing graphical representation, lack of safe guidelines, etc. In this paper, we explored the use of the *i** framework in industry for modeling organizational context. We review the models resulting from 36 industrial collaborations conducted in the last five years, where *i** has been intensively used by novice modellers, without previous exposure to *i**, acting as junior consultants in the organizations. We identify and categorize the main problems that they faced and as a result, we propose a set of guidelines to improve the adoption and practical application of the framework.

CCS Concepts
• **Information systems**

Keywords
Goal-oriented Modeling, iStar, Context Models, Enterprise Architecture, Social Dependencies, Industrial Validation.

1. INTRODUCTION
Modern enterprises organize their business processes upon information systems designed to manage the increasing complexity of their interactions with the environment.

Enterprise Architecture [1] is a widely adopted approach for architecting such systems, involving several levels of design which, starting from the business strategy, allow to identify the information system architecture. Early phases of enterprise architecture design are usually oriented to model the enterprise context [2], aiming to understand the purpose of enterprises in their context (e.g., what is needed from them), to help decision makers to design and refine their business strategies, and to support the enterprise architect to understand what is required from the resulting system.

In order to support these early phases, several approaches have been proposed both in industry and in academy. One line of research makes use of goal-orientation [3]. Goal-oriented modeling is a widely used approach in requirements engineering [4][5], business engineering [6][7] and also architecture engineering, both from a software oriented perspective [8][9] and an enterprise oriented perspective, as is our context here [10-13].

In such modeling context, the selection and use of the right notation is crucial for achieving good results, especially thinking of application in industrial contexts, where efficiency and effectiveness are must-be criteria. The authors of this paper have used the notation provided by the *i** framework [14] as the basis of a method for enterprise architecture construction called DHARMA [15]. After this formulation, the method has been used in a total of 36 industrial collaborations with good results. The outcomes of these large number of experiences is the possibility to assess the adequacy of the *i** language as modeling notation for practitioners who have never use it before. The limited level of adoption in industry is well known by the *i** community (with the remarkable exception of the GRL dialect [16] becoming part of the URN telecommunications standard) and has been regularly pointed out as one of the main challenges to overcome for *i** ramping up as a modeling asset towards the current academic circle [17]. In fact, it has been one of the motivations behind the formulation of a standard core notation recently issued [18].

This paper presents an assessment of the *i** language as a modeling language for novices without experience in goal-oriented modeling based on the 36 industrial cases conducted with the DHARMA method. The emphasis is on the context modeling phase of the

enterprise architecture design, which can be considered similar enough to usual early requirements engineering activities, therefore the results of this study shall be of interest for the requirements engineering community. The *i** context models were produced by junior consultants, and reviewed afterwards by the authors, acting as modeling experts, with the object of identifying typical mistakes, apply proper corrections and propose guidelines to improve both training and adoption of the *i** notation in future industrial cases. As an additional outcome of this study, we propose a set of guidelines helping novice practitioners in building such context models for their organizations.

The rest of the paper is structured as follows. Section 2 provides the relevant background. Section 3 sets the research questions and Section 4 explains how the study has been instrumented. Section 5 shows the results of the study and conducts their analysis, reporting typical mistakes done by the junior consultants in the construction of the models. Section 6 exploits these results to enumerate some recommendations in order to avoid these errors. Section 7 reflects on threats to validity. Section 8 presents the conclusions and some future work.

2. BACKGROUND

In this section, a brief description to the *i** language and the DHARMA method is provided.

2.1 The *i** language

The *i** framework [14] 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).

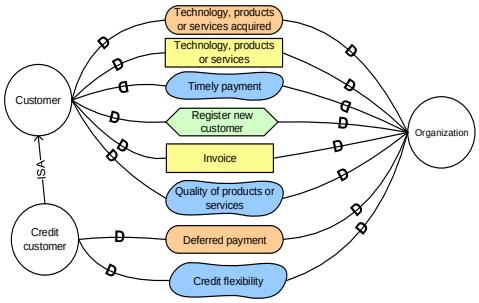

Figure 1: Excerpt of an SD model representing the intentionality between actors *Organization* and *Customer*

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 *depender*, 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; *goal*, represented by an oval (e.g., Invoice Purchased) and *softgoal*, represented by a shrunken 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.

2.2 The DHARMA Method

The DHARMA method [15], allows the definition of enterprise architectures using the *i** notation. This method is based on two concepts defined by Porter [18]: 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) value chain that includes primary and support activities. The DHARMA method is structured into four main activities as shown in Figure 2.

Activity 1: Modelling the enterprise context. The organization and its strategy are carefully analysed, to identify its role inside the context. This analysis uncovers the *Context Actors* (CA) that surround the organization, and the *Organizational Areas* (OA) that structure it. CA are identified in relation to Porter's market forces and examined in relation to each OA in Porter's value chain, to identify strategic needs among them (*Context Dependencies*). Also OAs are analysed in relation to each other to identify their strategic interactions (*Internal Dependencies)*. *i** SD models are built and used to support reasoning and represent results from this activity. Various Context Model (CM) are constructed from the perspective of each OA, including their related CA and OAs as well as their contextual and internal dependencies. Resulting models are eventually combined into a single enterprise Context Model (CM; see Figure 2, Activity 1).

Activity 2: Modelling the environment of the system. A system-to-be is placed into the organization (it can be a pure information system or a hybrid system including hardware, software or hardware with embedded software components) and the impact that it has over the elements in the CM is analysed. Strategic dependencies identified in the previous activity (internal and context), are examined to determine which of them may be totally or partially satisfied by information system. These dependencies are redirected inside the *i** SD diagram to the information system. The model includes the organization itself as an actor in the system environment, its needs are modelled as strategic dependencies over the system (see Figure 2, Activity 2).

Activity 3: Decomposition of system goals and identification of system actors. Dependencies included in the CM are analysed and decomposed into a hierarchy of goals required to satisfy them. The goals represent the services that information system must provide, to support interaction with CA and OA activities. An *i** SR diagram for the system is built, using means-end links of type goal-goal (representing then a decomposition of objectives into sub-objectives) (see Figure 2, Activity 3).

Activity 4: Identification of system architecture. Finally, goals included in the SR model are analysed and systematically grouped into *System Actors* (SA). Objectives are clustered into services, according to an analysis of the strategic dependencies with the

environment and an exploration of software components marketplace. Relationships between SA that form the system architecture are described according to the direction of the means-end links that exist among the objectives included inside them. SA are not software components; instead, they represent atomic software domains for which several situations may occur: 1) there can be a software component covering the functionality of several SA (e.g., ERP system); 2) the functionality of a single SA is covered by several software components for ubiquity reasons (e.g. mobile and local applications); 3) or there can be cases for which no software components exist, leading to the need of bespoke software (see Figure 2, Activity 4).

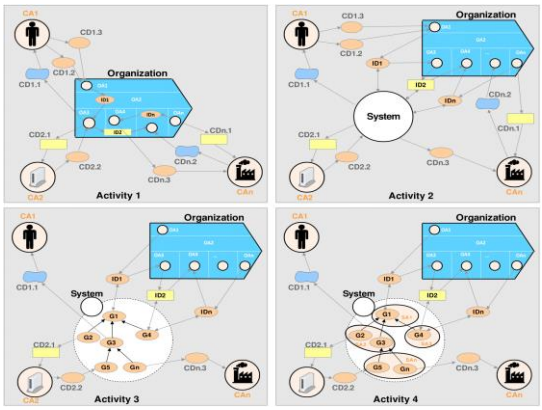

Figure 2: The DHARMA method

It is worth mentioning that the DHARMA method, due to its practitioner-oriented nature, does not use the fully-fledged *i** SD language constructs. The elements that are left out are: 1) actors can only be generic actors, i.e. no further classification into agents, roles and positions is supported; 2) the only relationship among actors is specialization, there are not part-of or other relationships that exist in some language versions; 3) the concept of dependency strength does not exist. Therefore, these aspects of the language are not assessed in the study reported in this paper. On the contrary, DHARMA adds a classification of actors into four categories: human, organization, software and hardware.

3. RESEARCH QUESTIONS

To formalize the study, we formulate two research questions to be answered based on the information obtained from the CMs built by the junior consultants.

RQ1: Is the concept of i actor, its types (according to DHARMA) and the is-a (sub-typing) relation understood by the junior consultants who participated in the CM construction?* The identification of actors is one of the most important activities when building SD models, and they are the starting point to address a consultancy based in the DHARMA method and are key to define social dependencies. In other words, if one actor is omitted, inappropriately classified or with ambiguous semantics, the model will suffer from inconsistencies and lack of information.

RQ2: Is the concept of dependency, as well as the four types of dependency proposed in the i language, understood by the junior consultants who participated in the CM construction?* This research question addresses one of the most distinctive concepts of the *i** notation, which in turn is one of the basis of any approach that delivers SD models, as DHARMA does. We aim to determine if the concept of social dependency is easily understood by newcomers in the use of *i** language; if not, it could be argued that the *i** language is not adequate for our purposes. In addition, in order to assess the correct use of dependencies, the objective was to determine if the semantics of the four types of *i** dependencies are easily understood by the consultants, or conversely, its boundaries are somehow blurred. The direction of the dependency and the description of the dependum are also addressed by this RQ.

4. STUDY INSTRUMENTATION

The base models used for this analysis were constructed by university students in their final grade project, acting as junior consultants in companies, who were trained in the construction of CM, specifically 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 paper authors' university. 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 network is composed by small companies as majority (97,94%) [20]. According to NACE Rev 2. [21], the organizations of the study were categorized using domains such as Manufacturing, Wholesale and retail trade, and Services.

For each organization was assigned a pairs of junior consultants, created a single model. The modelling process started with the construction of CMs and finished with the identification of the IS architecture required to support its operation. The junior consultants worked in pairs in order to complete each activity, by interviewing the responsible of each organization.

Resulting *i** CMs were next represented in tabular format, following the guidelines proposed in the DHARMA method. In order to make the analysis easier, additional columns were added to the CM table to allow the authors (acting as modeling experts) providing proper corrections (type, direction, description, etc.), without overriding or modifying the original dependencies containing errors written by the junior consultants. Table 1 shows an excerpt of one of such tables for dependencies analysis and corrections.

5. ANALYSIS OF THE MODELS

Once all the CM models for the 36 organizations were consolidated in a single data space, they were analyzed by the authors playing the role of modeling experts. Therefore, the actors and dependencies analysis tables mentioned in Section 4 were filled. The result of this consolidation is compiled in Table 2. The table is divided into three big areas: the leftmost and central parts for actors and is-a links, to answer RQ1; the rightmost part for dependencies, to answer RQ2.

The study shows that the total number of actors is 1,111, and up to 204 errors were detected, which represents 18.36%, with a minimum of 10.87% and a maximum of 32.00% errors. Errors belong to two categories.

- Type errors: they represent 58.82% of all errors related to actors. See row 4 in Table 1.

- Name errors: less frequent than the former ones (41.18% of the total), we have classified names as errors only when they were misleading or simply wrong. See row 5 in Table 1.

Table 1: Excerpt of the dependencies analysis table

Dependencies identified in the CMs						Corrections performed			
N°	**Actor1**	**Dependum**	**Actor2**	**Type**	**Dir**	**Dependum**	**Actor2**	**Type**	**Dir**
1	Wholesale customer	Large purchase order	Sales	Goal	←			Resource	←
2	Teacher	Teachers acquired	Customer	Goal	→	Teaching services acquired	Student		←
3	Supplier	Commercial contract made	Sales	Goal	→				←
Actors identified in the CMs						**Corrections performed**			
	Actor		**Type**			**Actor**		**Type**	
4	Public Supplier		Person					Organization	
5	Sales in Pharmacies		Organization			Pharmacies			

For is-a, the error rate is greater. Consultants declared 839 is-a relationships in the model, but we found 217 (25.86%) errors. Hierarchy-related errors were more complex to identify, because it depends on the consultant who is building the CM to decide which level of specification would be the most convenient. To simplify the process, decisions were based on the dimensions identified in [21], which are categorizations of actors that help consultants to identify and organize instances of actors. Thus, it becomes easier to identify the validity of the is-a relations defined in the CMs under analysis. As example, consider the actor identified as "Customer" (see row 2 in Table 1), which has been considered as the main actor in the dependency, but according to the dependum, it could be sub-classified as "Student".

Finally, the models included 2.095 dependencies, in which we detected 318 errors (15.18%). The dependencies' analysis implied a greater effort because in addition to description and type errors, the tuple of depender, dependum, dependee and direction has to be analyzed as one single entity. Dependency-related errors belong to four categories:

- Type errors: they represent 26.73% of all errors related to dependencies. One method to discern this type of error is the categorization by grammar, that is, goals have to be expressed as a verb in past participle, softgoals have to include an adjective or adverb, resources have to be nouns and tasks include more than often an infinitive verb in the name. However, this is not the only criterion and expert judgement was always needed. For instance, the dependency "Large purchase orders" was classified as a goal by the junior consultant, but its correct type should be resource (and not a softgoal) even considering "large" as matter of opinion. See row 1 in Table 1.
- Name errors: they represent 32.08% of all errors related to actors. See row 2, Dependum column in Table 1.
- Direction errors. This type declares depender and dependee playing each other role. It was the most numerous type of error by far, with 131 occurrences (41.19%). As example of direction error, see row 3 in Table 1.

We also searched for correlations among the types of errors and other characteristics. We found only a few explained below. First, Fig. 3(a) correlates actor-related errors with the placement of such actors in its environment. The results point out that errors concerning to internal CA are firstly of name (60%), and a low rate for type errors (40%); for external CA, errors are mainly due to is-a hierarchy identification (52%), and less for type (28%) and name (20%). Second, Fig. 3(b) correlates dependency-related errors with the type of the dependum, where direction errors are the most frequent for goals (60%) and resources (55%); for tasks, all presented errors are of name, for softgoals the majority of errors are related to type (45%) and name (43%). Last, Fig. 3(c) shows the error rate per industry classification, where the error rate is not too different between the different industries, but the most common errors are of is-a hierarchy, followed by actor errors and lest frequent dependency errors.

In addition, we performed an analysis of the error growth for actors and dependencies, see Fig. 4, where the "X" axis represents the number of elements (actors or dependencies) included in the models, and the "Y" axis contains the number of elements with errors. The results show that it has a linear distribution, for example in Fig. 4a, for each six actors modelled, one has some kind of error (name, type or hierarchy), the same phenomenon occurs with dependencies (fig. 4b) where every 8 dependencies one has almost one error (name, type or direction).

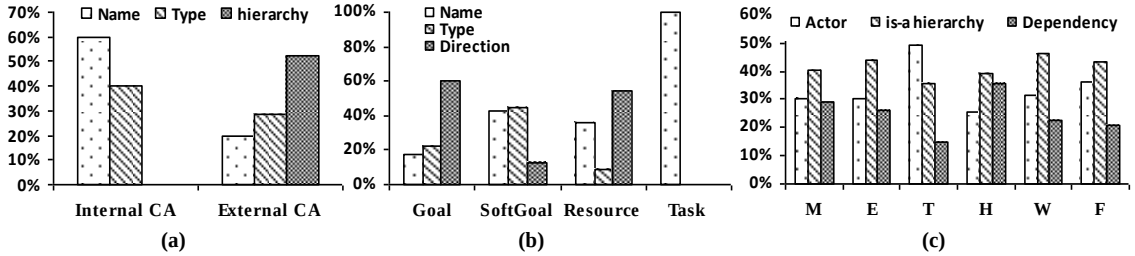

Figure 3: Correlations among the types of errors. a) Type of actor, b) Type of dependency and c) Industry classification (M - Manufacturing, E - Education, T - Transportation, H - Human Health, W - Wholesale and F - Financial activities)

Table 2: Errors identified per organization

Organization	Actors		Actor Errors				is-a hierarchy Actor			Dependencies																				
										Goal				Softgoal				Resourse				Task				Name	Type	Direction	T. Error	% Error
	External Context Actor	Internal Context Actor	Type	Name	T. Error	% Error	Total	T. Error	% Error	Total	N	T	D	Total	N	T	D	Total	N	T	D	Total	N	T	D					
Org1	22	7	2	2	**4**	13,79%	23	**7**	30,43%	16	0	1	4	11	1	3	0	18	0	0	2	0	0	0	0	1	4	6	**11**	24,44%
Org2	24	6	3	2	**5**	16,67%	23	**5**	21,74%	31	3	1	5	10	1	2	1	12	0	0	0	0	0	0	0	4	3	6	**13**	24,53%
Org3	26	4	5	2	**7**	23,33%	22	**7**	31,82%	22	2	2	5	9	1	2	0	0	0	0	0	0	0	0	0	3	4	5	**12**	38,71%
Org4	27	8	2	3	**5**	14,29%	27	**8**	29,63%	36	0	1	5	19	3	3	2	26	2	0	1	1	0	0	0	5	4	8	**17**	20,73%
Org5	23	0	3	2	**5**	21,74%	17	**7**	41,18%	15	1	1	2	11	1	2	0	10	0	0	0	0	0	0	0	2	3	2	**7**	19,44%
Org6	19	5	2	1	**3**	12,50%	17	**4**	23,53%	39	0	1	3	12	3	1	1	10	0	0	0	2	0	0	0	3	2	4	**9**	14,29%
Org7	18	5	2	1	**3**	13,04%	17	**5**	29,41%	26	1	0	2	2	0	1	0	2	0	0	0	0	0	0	0	1	1	2	**4**	13,33%
Org8	17	0	2	1	**3**	17,65%	11	**3**	27,27%	19	0	1	1	4	1	0	0	13	0	0	1	0	0	0	0	1	1	2	**4**	11,11%
Org9	27	7	5	2	**7**	20,59%	28	**8**	28,57%	39	1	1	4	18	3	0	0	22	0	0	1	7	1	0	0	5	1	5	**11**	12,79%
Org10	16	4	2	2	**4**	20,00%	15	**5**	33,33%	14	0	1	1	6	0	0	0	5	0	0	0	2	0	0	0	0	1	1	**2**	7,41%
Org11	44	6	4	4	**8**	16,00%	39	**10**	25,64%	29	1	2	3	15	1	1	0	15	0	0	2	0	0	0	0	2	3	5	**10**	16,95%
Org12	13	7	4	1	**5**	25,00%	12	**3**	25,00%	25	1	1	0	22	1	2	0	18	1	0	1	2	1	0	0	4	3	1	**8**	11,94%
Org13	24	4	3	2	**5**	17,86%	22	**7**	31,82%	15	0	2	2	8	0	1	1	14	1	0	1	4	1	0	0	2	3	4	**9**	21,95%
Org14	26	5	6	3	**9**	29,03%	26	**7**	26,92%	29	2	0	1	19	2	1	0	25	2	0	1	2	1	0	0	7	1	2	**10**	13,33%
Org15	38	9	3	3	**6**	12,77%	37	**6**	16,22%	29	0	1	3	23	1	2	0	22	0	0	1	4	1	0	0	2	3	4	**9**	11,54%
Org16	24	10	3	2	**5**	14,71%	26	**6**	23,08%	32	1	1	3	15	3	1	0	32	1	1	1	6	1	0	0	6	3	4	**13**	15,29%
Org17	25	7	5	3	**8**	25,00%	24	**8**	33,33%	16	1	0	2	10	0	1	0	16	1	0	1	0	0	0	0	2	1	3	**6**	14,29%
Org18	20	6	3	2	**5**	19,23%	19	**5**	26,32%	14	0	1	0	21	1	1	0	24	0	0	0	0	0	0	0	1	2	0	**3**	5,08%
Org19	22	3	4	2	**6**	24,00%	20	**7**	35,00%	16	0	1	3	11	0	1	1	13	0	0	0	1	0	0	0	0	2	4	**6**	14,63%
Org20	26	7	3	2	**5**	15,15%	25	**8**	32,00%	9	0	1	3	9	0	1	0	11	0	0	1	2	1	0	0	1	2	4	**7**	22,58%
Org21	38	5	3	3	**6**	13,95%	35	**6**	17,14%	40	1	1	1	19	1	1	0	21	4	1	1	1	0	0	0	6	3	2	**11**	13,58%
Org22	22	8	3	2	**5**	16,67%	20	**4**	20,00%	30	1	1	4	18	0	0	0	11	0	0	0	4	0	0	0	1	1	4	**6**	9,52%
Org23	25	7	6	3	**9**	28,13%	24	**8**	33,33%	28	0	2	2	21	1	3	0	24	1	0	2	3	1	0	0	3	5	4	**12**	15,79%
Org24	21	3	2	2	**4**	16,67%	17	**5**	29,41%	29	1	0	3	17	2	0	0	39	1	0	1	1	0	0	0	4	0	4	**8**	9,30%
Org25	32	6	6	2	**8**	21,05%	30	**8**	26,67%	17	0	1	2	13	0	2	0	15	0	0	0	1	0	0	0	0	3	2	**5**	10,87%
Org26	21	6	3	2	**5**	18,52%	19	**6**	31,58%	24	2	2	2	14	0	1	0	27	0	0	1	7	1	0	0	3	3	3	**9**	12,50%
Org27	33	10	7	4	**11**	25,58%	31	**8**	25,81%	46	0	3	7	32	4	2	1	32	0	0	2	3	0	0	0	4	5	10	**19**	16,81%
Org28	38	5	2	3	**5**	11,63%	34	**6**	17,65%	20	1	0	2	22	1	1	1	12	1	0	0	1	0	0	0	3	1	3	**7**	12,73%
Org29	18	14	2	3	**5**	15,63%	23	**4**	17,39%	30	0	0	3	23	3	3	2	21	2	1	2	7	0	0	0	5	4	7	**16**	19,75%
Org30	26	8	3	4	**7**	20,59%	24	**7**	29,17%	28	1	1	4	33	6	4	1	35	3	1	2	8	0	0	0	10	6	7	**23**	22,12%
Org31	16	7	3	2	**5**	21,74%	14	**6**	42,86%	13	2	0	1	4	0	1	0	8	0	0	0	1	0	0	0	2	1	1	**4**	15,38%
Org32	21	4	5	3	**8**	32,00%	19	**4**	21,05%	16	1	0	1	7	0	0	0	5	0	0	1	0	0	0	0	1	0	2	**3**	10,71%
Org33	14	5	2	1	**3**	15,79%	12	**4**	33,33%	10	0	0	1	7	1	1	0	3	0	0	0	0	0	0	0	1	1	1	**3**	15,00%
Org34	33	13	2	3	**5**	10,87%	36	**5**	13,89%	9	0	0	0	10	1	1	1	14	0	0	3	0	0	0	0	1	1	4	**6**	18,18%
Org35	26	12	2	3	**5**	13,16%	32	**7**	21,88%	15	1	1	1	22	1	2	1	26	0	1	1	5	1	0	0	3	4	3	**10**	14,71%
Org36	23	0	3	2	**5**	21,74%	19	**3**	15,79%	36	1	0	2	20	2	0	0	18	0	0	0	2	0	0	0	3	0	2	**5**	6,58%
Total	**888**	**223**	**120**	**84**	**204**		**839**	**217**		**862**	**26**	**32**	**88**	**537**	**46**	**48**	**13**	**619**	**20**	**5**	**30**	**77**	**10**	**0**	**0**	**102**	**85**	**131**	**318**	

Last, we report a few additional observations emerging from the analysis:

First, although described independently, errors occurred sometime together. For instance, the dependency "Teachers depends on Customers to Evaluate Performance" (see Table 1) contains two different inconsistencies. On the one hand, the Customer actor could be specialized as Student, and on the other, the Teacher is the actor who satisfies the dependency, it means that the Teacher actor is the dependee and the direction of the dependency has to be the other way.

Second, another perspective for the analysis comes if we consider the actors and dependencies not independently at every model, but altogether. When we consolidated the models, we concluded that the 1,111 actors were in fact instances of a basic set of 302 actors

that appeared in various models, many of them stemming from Porter's market forces and value chain [19]. Similarly, different dependencies were only 720, which were instantiated up to the total number of 2.095. This observation is important in connection with the guidelines we propose in Section 6.

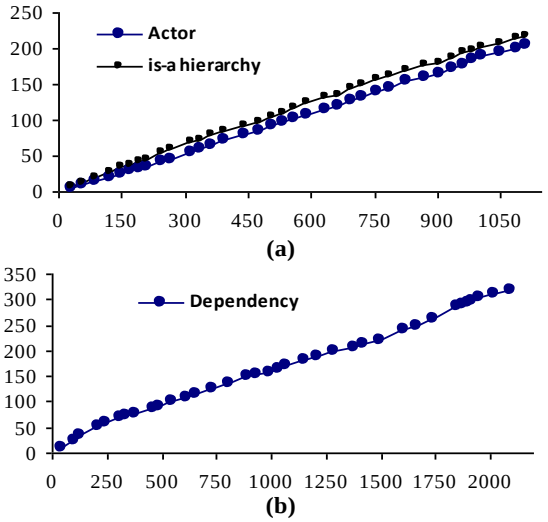

Figure 4: Error growth

Third, a very important thing that appears as result of the dependencies' analysis is the identification of bi-directional dependencies. This appears when both actors involved in the dependency have the same intentionality upon each other, which has to be satisfied (probably) simultaneously. An example of this case can be reflected in the dependency "Customer depends on Sales to get a Signed Invoice", where the invoice has to be signed by both, the Customer and the Seller. During the analysis, 7 bi-directional dependencies were found, which have not been considered as direction errors because the junior consultants weren't trained against those cases.

Last, two common errors were identified during the analysis in relation to the allocation of dependencies in is-a hierarchies:

i) *Generalization of dependencies:* some dependencies did not have an adequate definition of inheritance (is-a relations), that is, an actor has been associated with very specific dependencies. An example is the dependum "Evaluate performance" which has been assigned to the actor Customer instead of the actor Student. The total number of generalization of dependencies discovered is 5 cases.

ii) *Specialization of dependencies:* Contrary to the previous case, some dependencies have been assigned to actors in a very low hierarchical level, such dependencies must be associated to the parent actor instead of generating a new specialization that is increase unnecessarily the number of environment actors, with 7 cases discovered.

6. SOME GUIDELINES TO FACILITATE THE *i** LANGUAGE ADOPTION

After reviewing the typical problems in the use of the *i** language, with focus on SD models, we propose in this section a set of guidelines to improve its adoption by practitioners with no prior experience in the notation as it was the case of the junior consultants involved in our study. These guidelines make use of a catalogue of actors and dependencies that was built from the first 29 cases [20] that we addressed in this study. The catalogue encompasses the 302 actors mentioned in Section 5, structured in a three level *is-a* hierarchy that departs from 17 generic actors, 8 external: *Suppliers*, *Consumers*, *Strategic Partners*, *Distributors*, *Financial Institutions*, *Regulatory Agencies*, *Control Agencies* and *Competitors* and 9 internal: *Inbound Logistics*, *Operations*, *Outbound Logistics*, *Marketing and Sales*, *Services*, *Infrastructure*, *Human Resources Management*, *Technology Development* and *Procurement*. The rest of the hierarchy includes 46 actors in the 2nd. level and 239 in the 3th. level.

The catalogue also includes the 720 generic dependencies also mentioned in Section 5, which can be assigned a set of labels used to categorize CA and OA in relation to several orthogonal dimensions. For instance, the *Suppliers* generic CA can be categorized in relation to three dimensions: *Location* (Local, National, International); *Kind of supply* (Products or Services); and *Volume* (Wholesale or Retail). Each label has a set of generic dependencies attached to it, so each time that a label is assigned to an actor, its related dependencies become eligible to be linked to the actor in a new CM.

Based in the catalogue, we suggest *i** novice practitioners to use the following procedure when building their CM:

1. **Identify actors from generic catalogues:** At the beginning of the process, instead of starting from scratch, select the CA and OA that are suitable for the organizational context model, from the generic actors included in the root level of the catalogue's actor hierarchy.
2. **Specialize actors based on categorization labels:** Once the appropriate generic CA and OA have been selected, identify their is-a specializations by selecting proper categorization labels in the catalogue. In this way, is-a hierarchies can be constructed in a prescriptive way.
3. **Complete actor's identification with proper instances:** Each organizational context has its particularities; therefore, it is a natural fact to be able to identify some specific CA and OA actors for the organization under study. However, after conducting over 36 industrial cases of *i**-based CM constructing processes, there is enough evidence to confirm that all instances will be in relation to more generic actors included in the catalogue. Together with Step 2, this step will guide novice practitioners to avoid mistakes in relation to the identification of actors, their type and/or their specialization.
4. **Populate the CM with generic dependencies:** As explained in the first paragraphs of this section, actor classification labels in the catalogue have attached several generic dependencies. Therefore, each time that a categorization label is selected in relation to a CA or OA, all the dependencies attached to the label should be linked to the CA or OA attached to the label and thus, included in the model.
5. **Refine dependencies in a pairwise way:** Similarly, to what has been explained in Step 3 in relation to actors, it is also a normal situation to be able to identify some dependencies specific for the organization under study, or to delete / modify some of the dependencies added in Step 4, if they are not

proper for the specific case. To conduct this process in a more systematic way, we recommend novice practitioners to refine dependencies in the model considering only two related actors at a time (one CA and one OA or two OA; we omit CA to CA dependencies since they are out of the context of the system, (see Figure 2, activities 2 through 4).

Other important aspect in relation to the catalogue mentioned in this section are the concepts of parametric actor and parametric dependency. Parametric actors include parameters in the actor's name and description, which can take values from an ordinal domain. Graphically, the parameter is written between the symbols <>, and the actors are annotated in the catalogue with the domain of values; for example, an actor in the catalogue is "*< important > customer*", where the *<important>* parameter can be replaced by labels such as *Recurring*, *Trusted* or *Frequent*. Other case of parametric actor documented in the catalogue is the case of generic actors associated to various segments in industry. For instance, the actor *Supplier of <type of service> services*, where the parameter *<type of service>* may adopt values as *Security*, *Telecommunication* or *Transport*.

In the same way, dependency descriptions can be parametric, based on the segment of industry to which the company belongs. To illustrate these cases, let's consider the generic dependency "*<products or services> acquired*" included in the catalogue. If the organization under study is a clothing store, the parameter *<products or services>* can adopt the value *clothing*, whilst in the case of an educational institution, the parameter can adopt the value *education services*. In this way, the final descriptions of the dependencies would be *clothing acquired* and *education services acquired*, respectively.

In the most generic case, a parametric actor may be associated to parametric dependencies depending on the parameter assigned to the actor. For instance, consider the parametric actor *Supplier of <type of service> services*, introduced above, which can be linked to the parametric resource dependency *<specific documents>*. If the actor is instantiated as *Supplier of transport services*, when the resource is instantiated at its turn, it has a binding relationship to the legislation of the country, which requires the resource's parameter to take as value "*Guide of remission*".

Based on this concept of parameter, the proposed construction procedure is completed with the following additional steps:

6. Replace the parameters of parametric actors by the values relevant for the context of the organization under study.
7. Replace the parameters of parametric dependencies by the values relevant for the context of the organization.
8. Replace the parameters of the parametric dependencies associated to each parametric actor, by values relevant to the context of the organization, considering the values assigned to the parameters of the parametric actors associated.

After reviewing the actor-related mistakes in the 36 cases reviewed in this paper, evidence points to the fact that if the catalogue had existed before conducting the experiences and consultants had followed the proposed guidelines, actor-related mistakes would had been significantly reduced. All of the actors included in the models are included in the hierarchy of the catalogue, except for very particular instances specific for some contexts. For instance, specific regulatory agencies in the country, or particular suppliers in the context of the organizations.

We are currently conducting a new set of industrial experiences with different junior consultants. Although we are in a preliminary phase of the study, we can already report that the number of errors in relation to actors and is-a hierarchies, can be reduced to almost zero. In the same way, evidence points not only to a significant reduction in the mistakes related to dependencies, due to the fact that the ones included in the catalogue act as check lists and can be reused in different modelling processes, but also to the fact that they can be used as examples to follow by practitioners, helping them to improve common mistakes. Additionally, the systematic nature of the process helps young practitioners to focus in very specific activities, one at the time, making model construction and refinement straightforward.

7. THREATS TO VALIDITY

As any other empirical study, our work faces some threats to validity that we summarize below [22].

Construct validity. 1) We made sure of performing a rigorous planning of the study and establishing a solid protocol and templates for data collection and data analysis by following guidelines for software engineering [23]. 2) Data collection was mainly made through several forms as the one presented in Table 1. We piloted the structure of such form to make sure that the data collected was the one really needed.

Internal validity. 1) Not all the models were constructed at the same time, but along the last four years. Although consultants were not trained at the same time, their training material included the exact same contents. All the courses used the same training material that were devised long ago and it is even used in an MSc course in the authors' university, therefore it can be considered highly adequate. 2) The identification of wrong constructs was done by expert judgement. This way, some model element might have been misclassified as correct or incorrect. To mitigate this threat, two authors of the paper reviewed all the models and in case of discrepancy or doubts, a third authors was involved.

External validity. 1) All the models were constructed for different organizations, in various segments of industry. As explained earlier, the study includes 27 small sized organizations, 6 medium sized, and 3 large sized which corresponds to the reality of the country where it was performed. Although 25% of the organizations were not small, and this fact may have introduced some noise, we did not observe significant differences among the results obtained in these organizations in relation to smaller ones, other that the time required for the modelling activities which was longer depending on the size of the organization. 2) Consultants were in fact young practitioners with willingness to learn and solid background. Running the same study with other type of practitioners could have yielded different results. Further studies with other type of practitioners would certainly help to obtain a more complete picture.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented the study performed in over 36 industrial projects in which junior consultants use the *i** language to build context models. The models were analyzed in order to determine the level of understanding of *i** constructs (with focus on SD diagrams) by new entrants in the field of software consultancy, as

well as the identification of common errors present when using the notation. Considering research questions RQ1 and RQ2, we believe that the results obtained in this study are relevant: the concepts of actor and dependency are understood, but a deeper explanation is needed since the 18.36% and 15.18% of all elements (actors and dependencies) in the models suffered from some type of error. Generalization and specialization of actors and their dependencies are the most difficult part of the process for junior consultants. Besides, we uncovered some correlations whose analysis may further help to improve results. Nevertheless, even in the current state, we believe that our results show that *i** can be successfully adopted by practitioners in the modeling activities.

Although there are many applications of *i** in industry, there are very few studies trying to measure the complexity of the language as we have done. The most remarkable existing work comes from Engelsman and Wieringa who performed some studies with practitioners assessing a goal-oriented extension of Archimate [12]. We think that our study corroborates the perception that the concept of goal is very well understood by practitioners.

Future work spreads along several dimensions. In the domain of context modeling targeted by the paper, we aim at linking the dependencies identified by the consultants with the areas in the value chain, in order to better distribute the responsibilities of the system. With respect to the analysis of SD models, completing the ongoing study mentioned at the end of Section 6 should allow to check if the guidelines provided in that section will really reduce the percentage of errors found in this study. With respect to scope, we plan to conduct a similar study focused on *i** SR models, analysing goal decomposition, means-end links, etc. Last, we are currently developing tool support for building *i** models applying the guidelines presented in Section 6, including a list of validated actors and dependencies, which will be used to support consultants in the modelling activities.

9. ACKNOWLEDGMENTS

This paper is part of the research project called "Towards the identification, validation and semantic enrichment of context models patterns to be used as basis for the definition of enterprise information systems architecture" sponsored by the Research Department of the University of Cuenca, Ecuador.

10. REFERENCES

[1] The Open Group. *The Open Group Architecture Framework (TOGAF)* version 9, 2009.

[2] Nadoveza, D., and Kiritsis, D. Ontology-based approach for context modeling in enterprise applications. *Computers in Industry* 65(9), 2014: 1218-1231.

[3] Guizzardi, R., Guizzardi, G., Perini, A., and Mylopoulos, J. Towards an Ontological Account of Agent-Oriented Goals. SELMAS 2006.

[4] Yu, E. Towards modelling and reasoning support for early-phase requirements engineering. ISRE 1997.

[5] Van Lamsweerde, A. Goal-oriented requirements engineering: A guided tour. ISRE 2001.

[6] Yu, E., and Mylopoulos, J. Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering. *International System in Accounting, Finance and Management* 5(1), 1996: 1-13

[7] Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., and Mylopoulos, J. Strategic business modeling: representation and reasoning. *Software and Systems Modeling* 13(3), 2014: 1015–1041.

[8] Grau, G., and Franch, X. A Goal-Oriented Approach for the Generation and Evaluation of Alternative Architectures. ECSA 2007.

[9] Pimentel, J., Lucena, M., Castro, J., Silva, C., Santos, E., and Alencar, F. Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach. *Req. Engineering Journal* 17(4), 2012: 259-281

[10] Carvallo, J.P., and Franch, X.: On the Use of *i** for Architecting Hybrid Systems: A Method and an Evaluation Report. PoEM 2009.

[11] Yu, E., Deng, S., and Sasmal, D. Enterprise Architecture for the Adaptive Enterprise-A Vision Paper. TEAR/PRET 2012.

[12] Engelsman, W., and Wieringa, R. Goal-Oriented Requirements Engineering and Enterprise Architecture: Two Case Studies and Some Lessons Learned. REFSQ 2012.

[13] Marosin, D., van Zee, M., and Ghanavati, S. Formalizing and Modeling Enterprise Architecture (EA) Principles with Goal-Oriented Requirements Language (GRL). CAiSE 2016.

[14] Yu, E. *Modelling Strategic Relationships for Process Reengineering*. Ph.D. Thesis, University of Toronto, Department of Computer Science, Canada, 1995.

[15] Carvallo, J. P., Franch, X. Descubriendo la arquitectura de sistemas de software híbridos: un enfoque basado en Modelos *i**. Proceedings of RE'09, pp. 45-56.

[16] *User Requirements Notation (URN) – Language definition*, https://www.itu.int/rec/T-REC-Z.151-201210-I/en, 2012.

[17] Franch, X. The *i** Framework: The Way Ahead. RCIS 2012.

[18] Dalpiaz, F., Franch, X., Horkoff, J. iStar 2.0 Language Guide. CoRR abs/1605.07767, 2016.

[19] Porter, M. *Competitive Strategy*. Free Press, New York, NY, United States, 1980.

[20] Abad, K., Carvallo, J. P., and Peña, C. iStar in Practice: On the Identification of Reusable SD Context Models Elements. iStar 2015.

[21] Office for Official Publications of the European Commu-nities. *NACE Rev 2. Statistical classification of economic activities in the European Community*. Luxembourg: Eurostat Methodologies and Working papers, 2008.

[22] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A. *Experimentation in Software Engineering*. Springer, 2012.

[23] Runeson, P., and Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14(2), 2009: 131-16.