A Method for the Definition of Metrics over *i** Models*

Xavier Franch

Universitat Politècnica de Catalunya (UPC)
c/Jordi Girona, 1-3, E-08034 Barcelona, Spain
franch@lsi.upc.edu

Abstract. The *i** framework has been widely adopted by the information systems community for goal- and agent-oriented modeling and analysis. One of its potential benefits is the assessment of the properties of the modeled socio-technical system. In this respect, the definition and evaluation of metrics may play a fundamental role. We are interested in porting to the *i** framework metrics that have been already defined and validated in other domains. After some experimentation with *i** metrics in this context, the complexity inherent to their definition has driven us to build a method for defining them. In this paper, we present the resulting method, *i*MDF$_M$, which is structured into 4 steps: domain analysis, domain metrics analysis, metrics formulation and framework update. We apply our approach to an existing suite of metrics for measuring business processes performance and drive some observations from this experiment.

Keywords: goal-oriented models, *i**, metrics, business process performance.

1 Introduction

Goal-oriented modelling [1] is widely used in Information Systems (IS) development as a way to establish high-level goals and decompose them until obtaining measurable requirements. High-level goals capture the overall organizational objectives and key constraints; therefore they represent stable needs that are less sensitive to changes.

The *i** framework [2] is currently one of the most widespread goal- and agent-oriented modelling and reasoning frameworks. It has been applied for modelling organizations, business processes, system requirements, software architectures, etc. As a modelling framework, one of its required applications is the ability to evaluate properties of the model that may help to detect some flaws in the modelled system, or to compare different alternatives with respect to some criteria.

As a result, some authors have explored techniques for driving the analysis of *i** models. Qualitative-predominant techniques were already formulated in [2] and later other techniques were proposed [3, 4]. Quantitative-predominant proposals aim at formulating metrics for measuring some criteria (see Section 2.1). Having good suites of metrics allows not only analysing the quality of an individual model, but also comparing different alternative models with respect to some properties in order to select the most appropriate alternative.

* This work has been partially supported by the Spanish project TIN2007-64753.

P. van Eck, J. Gordijn, and R. Wieringa (Eds.): CAiSE 2009, LNCS 5565, pp. 201–215, 2009.

Having this in mind, we proposed *i*MDF, an *i** Metrics Definition Framework [5] (see Section 2.2). The framework maintains a language of patterns in which metric templates are defined by means of OCL expressions expressed over an *i** metamodel. We are especially interested in the case in which the metrics are not defined from scratch, but they already exist in the domain that is being modelled with *i** (e.g., organizations, business processes, software architectures, etc.) and they have been properly validated. Therefore, the problem we face is not metric definition and validation, but mapping metrics from the starting domain to *i**.

As a result of experimentation with *i*MDF in the context described above, we have observed that the process for defining metrics may be quite complex, because it requires a full understanding of the domain that is being modelled using *i**, as well as the suite of metrics itself. Therefore, we have formulated a method, $i\text{MDF}_{\text{M}}$, for driving the process of definition of metrics in *i**. The presentation of this method is the main goal of the paper.

To illustrate the method, we define a suite of *i** metrics for business process design and evaluation based on a proposal from Balasubramanian and Gupta [6] that in its turn consolidates others' proposals. The definition of this suite becomes a second goal of the paper, both for the interest of the result itself (i.e., a representative iteration in the incremental construction of a comprehensive catalogue of metrics in *i**), and for the feedback over the framework (for refining the language of patterns, acquiring some more lessons learned, etc.).

Basic knowledge of *i** is assumed in the paper, see [2] and the *i** wiki (http://istar.rwth-aachen.de) for a thorough presentation.

2 Background and Previous Work

2.1 Quantitative Analysis of *i** Models

In spite of the high dissemination of the i* framework, only a few approaches have been proposed presenting some kind of metrics for measuring *i** models. We are mostly interested in quantitative-dominant proposals because they allow more objective and repeatable analysis of *i** models. Apart from *i*MDF itself, we mention Kaiya et al.'s AGORA method [7] that provides techniques for estimating the quality of requirements specifications with emphasis in the AND/OR decomposition of goals. Sutcliffe and Minocha [8] propose the analysis of dependency coupling for detecting excessive interaction among users and systems; they combine quantitative formulae based in the form of the model with some expert judgment for classifying dependencies into a qualitative scale. Bryl et al. propose structural metrics for measuring the Overall Plan Cost of agent-based systems [9].

2.2 *i*MDF: A Framework for *i** Metrics

The *i*MDF framework is the result of our work on *i** metrics over time:

– Phase 1. Preliminary work. Several metrics were defined ad-hoc for comparing alternative *i** models with respect to some properties [10]. This phase revealed the convenience of having some foundations for defining these metrics and provided the necessary expertise for formulating a framework with this goal.

– Phase 2. Formulation of the *i*MDF framework. From this experience and the study of other work done in the field, the *i*MDF framework was formulated embracing:
  - A metamodel [11] including the most relevant concepts in *i**. It is conceived as extensible, since this is a crucial characteristic of the *i** framework.
  - General forms of *i** metrics [12] and patterns for producing them [5].
  - A preliminary (formative) validation based on experimentation over individual metrics coming from several sources (e.g., [13]). The result formed a first catalogue of *i** metrics in *i*MDF.
– Phase 3. Validation of the framework. As mentioned in the introduction, we are currently porting some existing measurement proposals over *i*MDF. Also, some work is planned for formulating and validating metrics about the structural quality of *i** models (complexity, cohesion, ...). As a result, we are enlarging the *i*MDF catalogue whilst learning some insights about limitations of this approach.

Precisely, one of the identified limitations, the lack of clear guidance to define the metrics, has motivated the present work.

3 *i*MDF$_M$: A Method for Defining Metrics over *i** Models

In this section we describe the *i*MDF$_M$ method for developing metrics over *i** models in the *i*MDF framework. It consists of 4 steps (see Fig. 1) described below.

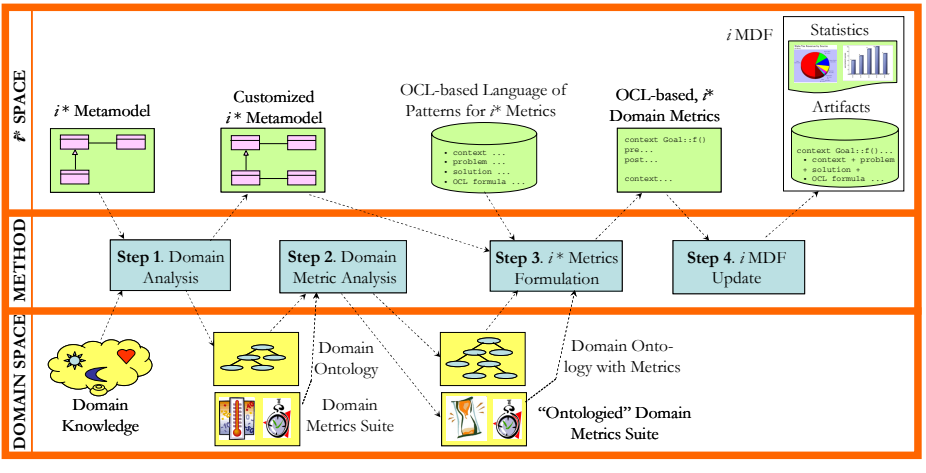

Fig. 1. The *i*MDF$_M$ method: steps and artifacts

3.1 Step 1: Domain Analysis

The goal of this step is to gain understanding about the domain whilst establishing the mapping from concepts in that domain onto the *i** framework. Therefore, Domain Analysis comprises two different activities:

- *Activity 1.1: Create a Domain Ontology*. From the knowledge about the domain (in the form of domain model semantics, related ontologies, tacit knowledge, etc.), an ontology is created or eventually, reused.
- *Activity 1.2: Map the Domain Ontology onto the i* Metamodel*. The correspondence between domain concepts and *i** constructs is established here. Concepts like e.g. business process, stakeholder, software component, etc., are therefore mapped onto *i** constructs like goal, task, agent, etc.
    - *Activity 1.2.1: Customize the i* metamodel*. The *i** metamodel as defined in [11] is refined into a specialization for the domain. This refinement step may involve adding some new attributes or even classes, or more usually integrity constraints that impose restrictions on the way *i** constructs are used to support the domain ontology.

3.2 Step 2: Domain Metrics Analysis

This step aims at analysing the departing suite of domain metrics before its formalization is tackled. The analysis moves along two directions, exogenous (seeking an accurate correspondence with the domain ontology), and endogenous (making the metrics uniform and complete). The activities performed are thus:

- *Activity 2.1: Extend the Domain Ontology*. The domain ontology is extended to incorporate those concepts that did not appear in the former analysis of the domain and that become necessary for using the metric suite.
- *Activity 2.2: Consolidate the Domain Metrics Suite*. The suite of metrics is analysed in the search of inconsistencies, lack of uniformity, ambiguities, etc. The domain ontology is extensively used during this activity. As a result, the suite is reformulated: definitions are clarified and eventually some metric may experiment some change (e.g., the subject over which the metric is applied may change).

3.3 Step 3: *i** Metrics Formulation

This step makes operative the metrics in terms of *i** constructs following the mapping from the domain ontology to the *i** metamodel established in Step 2:

- *Activity 3.1: Map the Metrics onto the i* Metamodel*. The formulation of the metric is analysed and the definition rephrased in terms of the *i** metamodel.
- *Activity 3.2: Express the Metrics in OCL*. The rephrased definition is made operative as OCL formulae expressed over the *i** metamodel class diagram taking the integrity constraints into account.
    - *Activity 3.2.1: Apply Language of Patterns*. Patterns from the *i*MDF catalogue are identified and applied wherever possible. In fact, it is expected that patterns cover most of the situations to be faced during the process of metrics definition, making thus this process easier.

3.4 Step 4: *i*MDF Update

Last, the result of the process is analysed to learn more about the method and the whole framework. This step combines three different activities:

- *Activity 4.1: Update Statistics*. Statistics refer specially to applicability of the patterns that form our language and are used in the Activity 4.2 below.
- *Activity 4.2: Update Language of Patterns*. At some moment, as a result of the accumulated statistics, patterns seldom used may be removed, or may be reformulated. Also, most used patterns may be further analysed for possible specializations. Last, some new patterns may be added after the current process.
- *Activity 4.3: Update Metric Catalogue*. Finally the decision to include or not the result of the current process has to be taken. Although the usual case should be to add the obtained metrics to the catalogue, we could eventually discard the suite for some reason (e.g., concerns on the mapping from the domain ontology to the *i** metamodel). Also, it could be the case that some particular metric is removed from the catalogue.

4 Applying *i*MDF$_M$ on a Business Process Modeling Metrics Suite

Balasubramanian and Gupta consolidated a metrics framework composed of eights metrics for business process design and evaluation [6]. These metrics come from others' proposals (remarkably Nissen [14, 15], and Kueng and Kawalek [16]) and address performance aspects such as process cost, cycle time, process throughput and process reliability. In this section we apply *i*MDF$_M$ over this suite of metrics.

4.1 Step 1: Domain Analysis

[6] proposes a 3-view model for business processes. The *workflow view* reveals the sequence of constituent activities and the business participants that execute them. In business processes, an activity may be defined as "work that a company performs" [17]. This is quite similar to the notion of *task* in the *i** framework, so we establish this fundamental equivalence (see Table 1). A sequence of activities may be thought as a particular kind of *routine* in *i**, i.e., a sequence of intentional elements that are inside some actor's boundary in the Strategic Rational (SR) view of the *i** model. To represent a sequence of activities, the routine must fulfill: 1) its components are just tasks; 2) constraints expressing precedence relationships are included; 3) there exists one and only one initial task. In particular, if task T2 goes after task T1, we assume a constraint Follows(T2, T1); if task T branches into T1, ..., T*k*, $k > 1$, we assume k constraints Branches(T, T1), ..., Branches(T, T*k*). We consider also a Join constraint which acts the opposite than Branches. Participants are represented as actors.

Table 1. Mapping among the concepts on [6] and the *i** metamodel constructs

Business Process Ontology according to [6]		***i** Metamodel*
View	Concept	*i** Element
Workflow view	Activity	Task
	Sequence of activities	Routine
	Business participant	Actor
Interaction view	Interaction	Resource dependency
	Business segment	Actor
	Operation	Is-part-of
Stakeholder-state view	Process stakeholder	Actor
	Milestone	Goal
	Visibility need	Goal placement + dependency

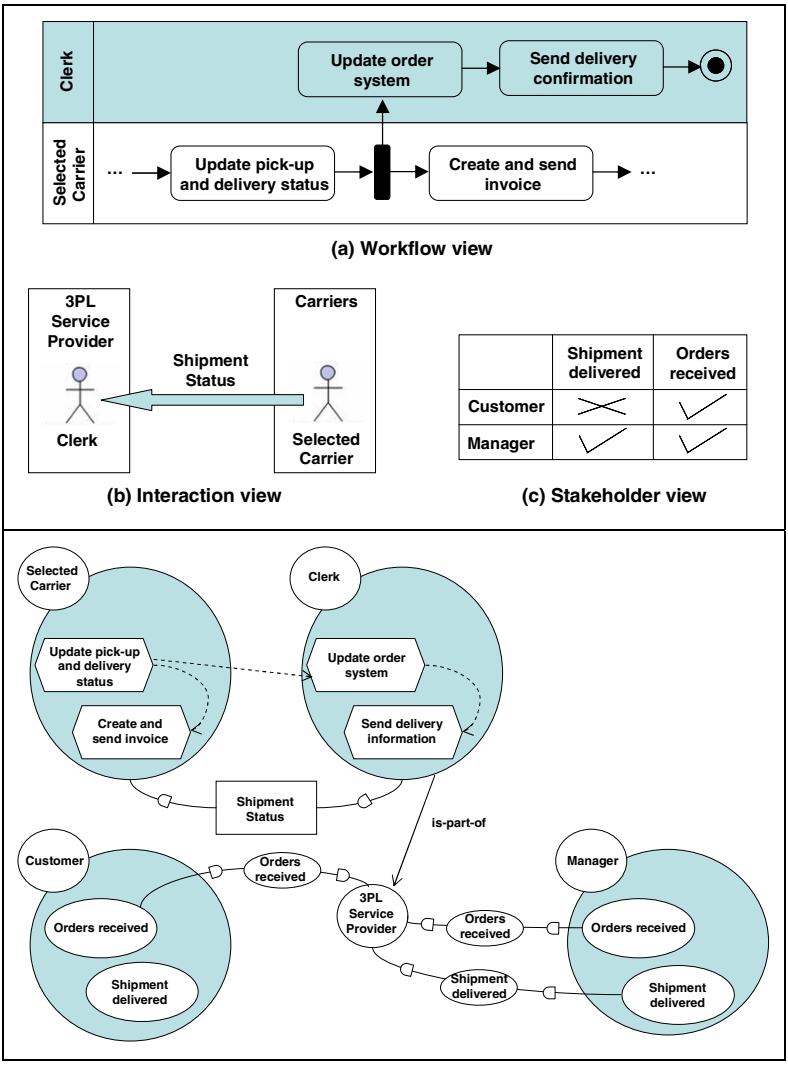

Fig. 2. An extract of process model according to the 3 views proposed by Balasubramanian and Gupta (up) and its mapping onto the *i** framework using the mapping given in Table 1 (down)

The *interaction view* reveals the interaction among the business participants and the business segments in which they operate. Interactions take the form of information transmitted between participants (e.g., transportation order, invoice, shipment status); therefore they may be modeled as resource dependencies: when the participant A interacts with B for transmitting the information C, we say that the actor B depends on A by means of a resource dependum C. For stating that a participant operates in a business segment, we again may represent segments by actors and represent this "operating" notion using a "is-part-of" relationship from the participant actor to the segment actor. We assume that a service provider will always be present in the model.

The *stakeholder-state* view reveals the important process stakeholders and the fulfillment of their process visibility needs with respect to identified process states or milestones. Again stakeholders may be modeled as actors. Milestones (e.g., orders received) can be represented as goals in *i**, placed inside the boundaries of those process actors that must satisfy the goal. If a stakeholder A has a visibility need with respect to milestone M, we include M also in A's boundary; if the need is satisfied, then we establish a dependency from that goal to the business segment corresponding to the service provider actor.

Each business process is constituted by these three views. In the usual case, the model will include several business processes whose views will coexist in the *i** model. Therefore, routines, stakeholders and milestones will appear altogether. Tasks may be part of different routines, with different constraints in the general case.

Fig. 2 shows an extract of a process model appearing in [6] and its correspondence in *i** according to the explanation above. The workflow view generates 2 actors and 4 tasks, as well as several precedence constraints (represented in the model as dotted arrows instead of textually) reflecting the activity relationships. The interaction view shows one interaction involving the same two actors, generating a resource dependency between them. Also, the operation of business participants in business segments appears in the view (just one of them has been represented in the *i** model by means of a is-part-of relationship). Last, the stakeholder view shows some milestones of the different stakeholders and three of them are satisfied in the given process model (so the fourth one will be unsatisfied or alternative means for satisfaction should be explored).

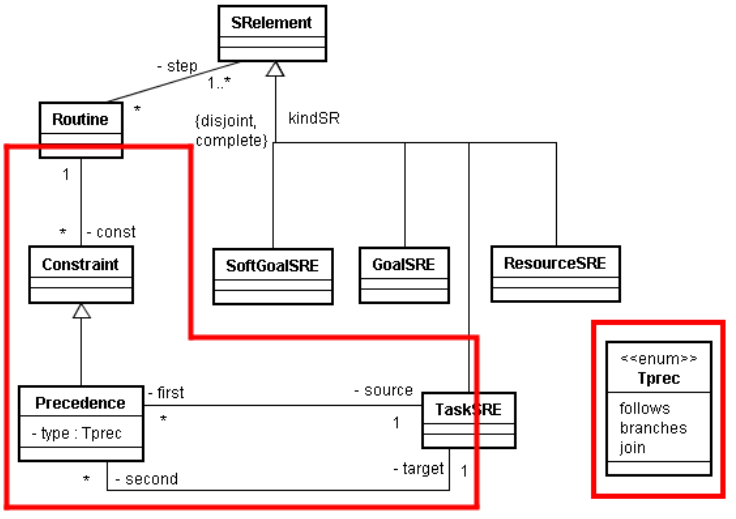

Fig. 3. Extract of the *i** metamodel as defined in [11] including the extension for Balasubramanian and Gupta's suite of metrics (framed). Integrity constraints not included. The complete class diagram can be downloaded from www.lsi.upc.edu/~franch/supplementaryMaterial/iStarMetaModelWithBPMconstructs.pdf.

The *i** metamodel must be customized to this mapping (this customization will be completed in Step 2):

1) We add an OCL invariant for restricting routine steps to tasks.

2) A subclass of *Constraint* named *Precedence*, with an attribute type that takes values from {Follows, Branches, Join}, is added (see Fig. 3, where the changes are framed; the whole metamodel is not included for space reasons, see [11] for details). It has two roles bound, *source* and target, to *tasks* that belong to the same routine than the constraint. New integrity constraints for avoiding error conditions (e.g., loops, joining disjoint paths) must also be added, as well as an integrity constraint for ensuring that there is just one initial state (a task that is not target of any other).

3) Resource dependencies inferred from the interaction view must be aligned with the precedence relationships stated in the workflow view. If a resource dependency stems from actor A to actor B, then some of B's activities must be executed before some A's activity according to those relationships. An integrity constraint ensures it.

It must be mentioned that the *i** model generated with this mapping does not present a lot of fundamental concepts: softgoals, resources inside actors' boundaries, links inside actors' boundaries, etc. We could have chosen to add some integrity constraints (or directly to prune the metamodel) to reflect this fact, but we think that better not: the modeler may decide to add this information to exploit fully the capabilities of the *i** framework. This has an important consequence when formulating the metrics: to be general enough, we have to consider this fact and in particular, we mention that allowing decomposition of tasks inside SR diagrams will have an impact on the final form that metrics will take in *i**.

4.2 Step 2: Domain Metric Analysis

We summarize next the business process metrics framework proposed in [6]. Since the departing proposal was quite uniform, consolidation was straightforward, i.e. definitions are basically quotations from that paper except in one case (APF, see below); for sake of brevity we do not provide the rationale for the metrics, see [6] for discussion. Underlined terms stand for concepts added to the domain ontology.

- *Branching Automation Factor* (BAF). Proportion of decision activities in a process that do not require human intervention. A decision activity is an activity that branches into several others in the workflow view.
- *Communication Automation Factor* (CAF). Proportion of inter-participant information interchanges in a process where the information source is a system.
- *Activity Automation Factor* (AAF). Proportion of total activities in a process that are either interactive or automated. An interactive activity is an activity performed by a human actor and assisted through a system. An automated activity is one that is performed entirely by a system.
- *Role Integration Factor* (RIF). Ratio of number of activities performed by a process actor where the process control is not passed to another participant within the same organization to the total number of activities performed by that actor.
- *Process Visibility Factor* (PVF). Proportion of number of process states required to be visible to process stakeholders that are actually reported to or recorded for the relevant stakeholders. A process state is a point where a milestone is achieved.

- *Person Dependency Factor* (PDF). Proportion of activities performed by human participants that are executed using human discretion within the entire process. In [6], human discretion is an attribute of activities in the workflow view.
- *Activity Parallelism Factor* (APF). Proportion of activities that are executed in parallel in a process. We think that this definition is not much accurate, for instance given a process with 5 activities T1, …, T5 the result would be the same for one process with branches (T1, T2), …, (T1, T5) and another with (T1, T2), (T1, T3), (T2, T4), (T2, T5). Therefore, we prefer to adopt the definition by Nissen [14] as the length of longest path of activities that must be executed sequentially divided by the total number of activities.
- Transition Delay Risk Factor (TDRF). Ratio of the number of activity control transitions to any human participant to the total number of transitions between participants in a process.

It is worth to mention that one of the metrics, RIF, is different than the others, since it is local to roles. For finishing this unifying step, we define an augmented version of RIF, RIF+, as the average of the local measures of RIF for all human roles.

4.3 Step 3: *i** Metrics Formulation

In this section we tackle *i** metrics formulation. We base this step on the *i*MDF language of patterns [5], see Table 2 below for a sample. For space reasons, we cannot present the metrics in detail. For illustration purposes, we show the two most difficult cases that may give an upper bound of the complexity of the process.

AAF. The main pattern applied is Normalization (shown in Table 2). The *Elem* is the *i** Routine that represents the business process under analysis in the *i** model, and the resulting *Type* is a float number. The *Size* is the number of tasks in that Routine:

```
Size ::= self.step.oclAsType(TaskSRE)-> size()
```

Table 2. Examples of patterns (given in abridged form, see [5] for details)

Category	Name	Description
Declaration	Individual	The metric applies just to one type of element, *Elem*
context *Elem*::metric(): *Type*		
Definition	Sum	The element metric's value is the sum of its components' values
context *Aggregated*::metric(): *Type* **post:** result = self.*aggregees*().metric()->sum()		
Numerical	Normalization	The metric needs to be restricted to some interval
context *Elem*::metric(): *Type* **post**: *Size* = 0 implies result = 1.0 **post**: *Size* > 0 implies result = *Value* / *Size*		
Navigational	Property evaluation	The value of some property is needed
context Node::propertyEval(name: String): *Type* **pre**: self.value->select(v | v.property.name = name)->size() = 1 **post**: self.value->select(v | v.property.name = name).val		

For *Value*, it must be noted that an automated activity is represented in an *i** model as a task that: 1) belongs to a Software actor and, 2) the task and all of its subtasks, if any, have dependencies just to other Software actors. An interactive activity is a task that: 1) belongs to a Human actor and, 2) the task itself, or some of its subtasks, has some dependency going to a Software actor:

```
Value ::= self.const.oclAsType(Precedence)->select(t | t.interactive() or t.automated())->size()
```

The refinement of the auxiliary operations is given below; they are obtained by applying the following patterns: *isHuman* and *isSoftware*, applying *propertyEval* (see Table 2); *allSubtasks*, applying *TopDownDecomposition* over task-decomposition links; *requiresSoftware*, applying *TransitiveCheck* (if task T1 depends on task T2, the condition holds either if T1 is inside a human actor, or if T2 or any of its subtasks depends on a human actor). Other auxiliary patterns were also applied.

```
Task::interactive() ::= isHuman(self.owner) and
                        exists(t | self.allSubtasks()->includes(t) and t.requiresSoftware())

Task::automated() ::= isSoftware(self.owner) and
                      forAll(t | self.allSubtasks()->includes(t) implies not t.requiresSoftware())
```

APF. Again we apply the Normalization pattern, computing the length of the longest path of the business process (i.e., Routine) and dividing by the total number of activities. Computing the length of the longest path is not straightforward due to possible branches and joins:

```
Elem ::= Routine; Type ::= Float
Size ::= self.step.oclAsType(Task)->size()
Value ::= self.allPaths()->select(p | self.allPaths()->forAll(p2 | size(p) >= size(p2))) ->size()
Routine::allPaths() ::= self.step->select(t | t.initialActivity()).allPaths()
Task::allPaths() ::= if self.finalActivity() then self
                     else self.allDirectSuccessors().allPaths()->prepend(self) end-if
Task::allDirectSuccessors() ::= self.firstComp->select(source=self).target
Task::finalActivity() ::= self.firstComp->size() = 0
```

4.4 Step 4: *i*MDF Update

Special importance takes the update of the language or patterns. It is still too early in our research to decide the removal of some pattern from the language, although some were not used in this particular case. Concerning the discovery of new patterns, we remark that 7 out of the 8 metrics were defined by a similar application of the *Normalization* pattern as done with AAF:

- The *Size* parameter is the size of a collection of *i** elements, e.g. the collection of all tasks in AAF, the collection of all stakeholder goals in PVF, etc.
- *Value* is defined by applying a filter (i.e., an aggregation operation such as Sum, Count, etc.) over the same collection than before.
- As a consequence, the *Type* is a float (in the interval [0, 1]).

Let's call *Col* that collection. Thus, the form that the *Normalization* pattern takes in these seven metrics is:

context Routine::metric(): Float
post: *Col*->size() = 0 **implies** result = 1.0
post: *Col*->size() > 0 **implies** result = *Col.filter*()->size() / *Col*->size()

Given its rationale, it is reasonable to expect that this variation of the Normalization pattern will be useful in future experiments and case studies. This is why we have decided to enlarge our pattern language with this expression upgraded to pattern (specialization of the *Normalization* pattern –we have specializations in our pattern language), just abstracting *Routine* to *Elem*.

Concerning the metrics catalogue, we incorporated this new suite.

5 Observations

In this section we summarize the key observations on this application of the *i*MDF$_M$ method and we try to extract some general risks and facts summarized in Table 3.

5.1 Step 1: Domain Analysis

This first step was really crucial for the success of the experiment, even more than expected beforehand. We observed two different sources of difficulties, corresponding to the two identified activities.

Creation of the domain ontology. In the 3-view model of business processes proposed in [6], we faced the problem of model integration (risk R1 in Table 3). We found two concrete difficulties:

Table 3. Risks (R_i) and facts (F_j) that may appear during the application of *i*MDF$_M$

Step	Act.		Risk / Fact
Domain Analysis	1.1	R1	Need of aligning different types of domain models that do not match exactly
	1.2	R2	Lack of some of the *i** expressive power in the domain ontology
		R3	Some concepts of the domain ontology cannot be directly mapped to *i**
Metrics Analysis	2.1	R4	The metrics suite is not completely aligned with the domain ontology
	2.2	R5	The suite of metrics is not uniformly defined
		R6	Some metric is not accurately defined and demands further investigation
Metrics Formulation	3.1	R7	Definition of many properties in the metamodel needed for mapping metrics
		R8	Mapping of metrics defined over an *i** metamodel richer than strictly needed
	3.2	R9	Some inherent characteristics may make the process harder (e.g., transitivity)
		F1	As the process progresses, reuse of concepts and OCL facilitates the process
		F2	The language of patterns is a good starting point for the metrics definition
iMDF$_M$ update	4.1	F3	Keeping track of pattern use statistics is essential for maintaining the framework
	4.2	F4	Upgrading an OCL expression into patterns mostly depends on frequency of use
	4.3	F5	Incorporating the result of the process into the catalogue will be the usual case

- The relationship among interactions in the interaction view and transitions in the workflow view is not explicit. If just one transition exists among two actors in the workflow view, it may be inferred, but otherwise the link must be established by observation or even it may require further investigation.
- The relationship among the milestones in the stakeholder view and the activities in the workflow view is not explicit. In this case, it is much harder to try to observe the link.

Both are instances of the same generic difficulty: aligning different types of models that are part of the departing domain.

Mapping onto the *i metamodel.** Basically we found two types of difficulties:

- Those coming from the departing ontology. If we assume that a business process model cannot be modified, the consequence is that the resulting *i** model is not as rich as it could be (risk R2). For instance, as a consequence of the observations above, resource dependencies cannot be established at the level of intentional elements but just at the level of actors. We may argue that this is not a problem since our goal is to define metrics on the *i** models that are equivalent to the original ones. Thus if the departing metrics were formulated without needing this information, we can do the same over the *i** models. This being true, we also think that wasting some capabilities of *i** models may make the approach less useful and less attractive. We envisage two solutions: 1) to refine the departing ontology; 2) to refine the resulting *i** model adding the missing information. Trade-offs between all the options should be considered in detail before taking any decision.
- Those coming from the fundamental differences between the domain ontology and the *i** metamodel (risk R3). Here, we have succeeded in translating all the constructs from the business process case, even those that had not a direct counterpart in *i**, by enriching the *i** metamodel. Enriching the metamodel means losing some kind of standardization, but as shown e.g. in [11], there are a lot of variants in the *i** framework and this is one of the features that makes *i** attractive.

5.2 Step 2: Domain Metrics Analysis

Extension of the Domain Ontology. In the definition of the departing metrics appeared some concepts that were not present in the ontology after Step 1 (risk R4). Notions like “decision activity”, “parallel execution of activities” and “process state” were not described precisely enough in the framework and we were forced to set our own interpretation of these terms. Other concepts like “activity control transitions” and “passing process control” had to be carefully examined. As a result, our [6]-based business process ontology grew.

Consolidation of the Metrics Suite. The definition of the metrics needed to be examined in detail. In our case, we found risks that may arise in future situations:

- Non-uniform definition of the metrics suite (risk R5). Uniformity is a fundamental property for conceptual frameworks. In our case, the metric RIF was clearly different from the others since it focused not just on a business process but also on a role. Thus, it did not fit with the overall goal of the metric framework, namely evaluating business processes. As a result, we proposed a slightly modification RIF+, although we also kept RIF to be respectful with the original proposal.

– Not accurate definition of a metric (risk R6). We look for metrics giving as much relevant information as possible, thus we were not happy with the definition of APF given in [6] and we preferred the original definition in [14], even paying the price of having a metrics quite different in structure than the others, therefore hampering somehow the uniformity criteria stated above.

5.3 Step 3: *i** Metrics Formulation

Mapping the Metrics onto the *i Metamodel.** Firstly, an issue is to what extent we need to add information (represented by properties in the *i** metamodel) to *i** models (risk R7). Too many properties would eventually require a lot of effort in the definition. In this case, we just needed 3 properties for the 8-metric framework. One of them, *Nature*, for knowing the type of an actor (human, software, etc.) was already introduced in *i*MDF before this experiment. Another one, *Process-Stakeholder*, to know the actor that owns a routine, has a high probability of reuse. Both of them are quite straightforward to evaluate. Thus their need is not really a strong drawback in terms of effort. The third one, *DecisionActivity*, to check if an activity is a decision activity, could be more difficult to handle in the general case but, in the departing proposal, decision activities are explicitly labeled as such. So, this third property does not raise any relevant problem neither (although it has a lower chance of reuse).

On the other hand, we have defined our metrics without considering some simplifications of the model to make them more robust (risk R8). Just to mention an illustrative example, in some metrics we have considered that tasks could be decomposed into subtasks although the departing framework as defined in [6] did not mention this case.

Expression of the metrics in OCL. Two of the most error-prone and cumbersome characteristics to face are transitive clousure and transitive definition of some operations (risk R9). Illustrative examples are: for the first case, the generation of all the paths or subtasks; for the second case the analysis of chains of dependencies.

In the positive side, as the definition of metrics progresses, it becomes easier to write them (fact F1). Two related reasons behind: the flavor of the metrics is similar after Step 2, and also some OCL expressions may be reused.

Use of the pattern language. For the definition of metrics itself, we used intensively the pattern language. The detailed results are given in the next subsection, but as a kind of summary we are quite happy with the results, the language demonstrated to be powerful and versatile enough (fact F2).

5.4 Step 4: *i*MDF update

Updating statistics. We have applied 59 patterns to define the 8 metrics (without considering RIF+). Each metric needs two declaration patterns (one for the context, other for the type) and a third pattern applied is *Proportion* (*Normalization* in the case of APF). The most complex in terms of number of applications has 11 whilst APF has just 4 (because we couldn't solve allPaths() by patterns) and next, PDF has 6. If we consider RIF+, we add 6 new applications of patterns. Keeping track of this statistics provide useful insights to the *i*MDF framework (fact F3).

Updating the pattern language. We have been able to formulate most of the metrics using intensively our pattern language. During the experiment, we faced two different expressions that could be upgraded into patterns. As seen in section 4.4, we defined a new pattern *Proportion* due to its intensive use in this framework and the conjecture that the situation dealt is likely to happen in the future. On the contrary, the *allPaths()* operation needed in APF, which was difficult and done ad-hoc, seemed so particular that we decided not upgrading it into a pattern (fact F4).

Updating the catalogue of metrics. For the metrics catalogue, since all the metrics were successfully solved, the whole suite of metrics could be incorporated into the catalogue. After the several experiences we have had, this is expected to be the usual case, provided that the whole experiment makes sense (fact F5).

6 Conclusions and Future Work

In this paper we have presented a method for defining metrics in *i** using the *i*MDF framework. Since we are interested in porting already existing, validated metrics to *i**, the method is largely concerned with the analysis of the domain and the metrics themselves, and the mapping onto the *i** metamodel, more than on design of com-pletely new metrics, which would a different matter of research. The method has been articulated by defining the relevant activities (organized into steps) and the artifacts involved. We have identified some risks and facts that may be used in future cases.

In addition, this paper has fulfilled a second goal, to offer a new suite of performance metrics for business process models represented in *i**. This new suite enforces our current catalogue in a domain we hadn't addressed before. Our language of patterns has been enlarged and we have obtained more statistical data about pattern use.

As future work, we are planning new experiments on different fields to the method and the whole *i*MDF framework whilst offering an increasingly large catalogue of metrics to the community. The experiments shall also assess the effort required to use this approach; this is a crucial validation to perform, since *i*MDF requires knowledge in: domain analysis, ontology construction, metamodeling and metrics. On the other hand, about implementation, after a first prototype available over an existing tool, we are starting to build a new tool taking advantage of the recent proposal of an XML-like standard for encoding *i** models called *iStarML* [18]. Our plans are to build the tool able to import models expressed in an *iStarML*-based grammar (the codification of the customization of the *i** metamodel). Translators from other models to *iStarML* (following the rules coming from Steps 1 and 2 of the process) would allow evaluating metrics over models built in the departing ontology.

References

1. van Lamsweerde, A.: Goal-oriented Requirements Engineering: A Guided Tour. In: Procs. 5th ISRE Intl' Symposium. IEEE, Los Alamitos (2001)
2. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Dissertation, Univ. of Toronto (1995)

3. Giorgini, P., Mylopoulos, J., Nicciarelli, E., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503. Springer, Heidelberg (2002)
4. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and Minimum-Cost Satisfiability for Goal Models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 20–35. Springer, Heidelberg (2004)
5. Franch, X., Grau, G.: Towards a Catalogue of Patterns for Defining Metrics over *i** Models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 197–212. Springer, Heidelberg (2008)
6. Balasubramanian, S., Gupta, M.: Structural Metrics for Goal Based Business Process Design and Evaluation. Business Process Management Journal 11(6) (2005)
7. Kaiya, H., Horai, H., Saeki, M.: AGORA: Attributed Goal-Oriented Requirements Analysis Method. In: Procs. 10th RE Intl' Conference. IEEE, Los Alamitos (2002)
8. Sutcliffe, A., Minocha, S.: Linking Business Modelling to Socio-technical System Design. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, p. 73. Springer, Heidelberg (1999)
9. Bryl, V., Giorgini, P., Mylopoulos, J.: Designing Cooperative IS: Exploring and Evaluating Alternatives. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 533–550. Springer, Heidelberg (2006)
10. Franch, X., Maiden, N.A.M.: Modelling Component Dependencies to Inform Their Selection. In: Erdogmus, H., Weng, T. (eds.) ICCBSS 2003. LNCS, vol. 2580. Springer, Heidelberg (2003)
11. Ayala, C.P., et al.: A Comparative Analysis of i*-Based Goal-Oriented Modeling Languages. In: Procs. 17th SEKE Intl' Conference, KSI (2005)
12. Franch, X.: On the Quantitative Analysis of Agent-Oriented Models. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 495–509. Springer, Heidelberg (2006)
13. Grau, G., Franch, X.: A Goal-Oriented Approach for the Generation and Evaluation of Alternative Architectures. In: Oquendo, F. (ed.) ECSA 2007. LNCS, vol. 4758, pp. 139–155. Springer, Heidelberg (2007)
14. Nissen, M.E.: Valuing IT through Virtual Process Measurement. In: Procs. 15th ICIS Intl' Conference. ACM, New York (1994)
15. Nissen, M.E.: Towards Enterprise Process Engineering: Configuration Management and Analysis. NPS Technical Report, NPS-GSBPP-02-003 (2002)
16. Kueng, P., Kawalek, P.: Goal Based Business Process Models: Creation and Evaluation. Business Process Management Journal 3(1) (1997)
17. White, S.A.: Introduction to BPMN.Report at BPMN website (2004), `http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf`
18. Cares, C., Franch, X., Perini, A., Susi, A.: iStarML: An XML-based Model Interchange Format for i*. In: Procs. 3rd i* Intl' Workshop, CEUR Workshop Proceedings, vol. 322 (2008)