On the Quantitative Analysis of Agent-Oriented Models

Xavier Franch

Universitat Politècnica de Catalunya (UPC),
C/Jordi Girona 1-3, UPC-Campus Nord (Omega), Barcelona, Spain
franch@lsi.upc.edu
http://www.lsi.upc.edu/~franch/

Abstract. Agent-oriented models are used in organization and information system modelling for providing intentional descriptions of processes as a network of relationships among actors. As such, they capture and represent goals, dependencies, intentions, beliefs, alternatives, etc., which appear in several contexts: business process reengineering, information system development, etc. In this paper, we are interested in the definition of a framework for the analysis of the properties that these models exhibit. Indicators and metrics for these properties are defined in terms of the model elements (e.g., actors, dependencies, scenario paths, etc.) Our approach is basically quantitative in nature, which allows defining indicators and metrics that can be reused in many contexts. However, a qualitative component can be introduced if trustable expert knowledge is available; the extent up to which quantitative and qualitative aspects are intertwined can be determined in every single case. We apply our proposal to the *i** notation and we take as main case study a highly-intentional property, predictability of model elements.

1 Introduction

Goal- and agent-oriented analysis methods and languages such as KAOS, *i**, GRL or TROPOS [1, 2, 3] are widespread in the information systems community for the refinement and decomposition of the customer needs into concrete goals, during the early phase of the requirements specification [4, 5]. This kind of models represents an organization and its processes as a network of actors and dependencies, which may be decomposed into simpler elements.

Once built, the models can be used for different purposes. Two of the most important ones are: analysis of the properties they exhibit, and comparison of alternatives. In the first case, it is checked whether some properties hold in the model; some actors or dependencies exhibit some property (either positive or not) are searched; etc. In the second case, different models, that represent different ways of implementing organizational processes or information systems, are compared with respect to properties that have been considered as crucial. In both cases, evaluation of models is the cornerstone of these analyses, and therefore some suitable metrics to rely upon are needed.

The use of metrics with this purpose is very common in other type of models. For instance, there are some suites of metrics in the field of object-oriented modeling [6, 7], which refer to structural properties like cohesion and coupling. Properties referring to

E. Dubois and K. Pohl (Eds.): CAiSE 2006, LNCS 4001, pp. 495 – 509, 2006.
© Springer-Verlag Berlin Heidelberg 2006

the system itself, such as security, efficiency or cost, which mainly fall into the category of non-functional or organizational requirements, appear when considering models of the system architecture [8]. These metrics are usually defined in terms of the components, nodes, pipes, etc., that compose the final configuration of the system.

In the case of goal- and agent-oriented modeling, typical approaches analyse models in a qualitative way, especially in conjunction with non-functional requirements [9], by targeting to specific properties such as availability, security and adaptability. These target properties are decomposed into simpler criteria that may be used to evaluate different candidate models for the system-to-be [10]. This evaluation is basically qualitative, which means that the extent up to which a criterion is fulfilled by a candidate model is determined by expert judgement. Although qualitative analysis is a powerful mechanism that is satisfactory in many cases, it may introduce a certain degree of uncertainness because it relies completely on the claims that experts make. The dichotomy among qualitative and quantitative analysis is not new and by no means exclusive of organization or information system modelling, or even the computer science discipline (see [11, p. 40] for an abridged comparison). Some researchers advocate that both types of analysis are exclusive [12], but others believe that they are compatible [13] and even complementary [14]. In goal-orientation, some contributions exist that combine quantitative and qualitative analysis for finding assignment of labels to nodes and determine its propagation in goal graphs [15, 16].

In this paper we are interested in the analysis of agent-oriented models with special emphasis on the quantitative side. To be able to express our approach in detail, we consider agent-oriented models written in the *i** language, although we think that the underlying concepts could be adapted to other approaches. More precisely, we want to take profit of the networked structure of *i** models to define structural indicators that are quantitative in nature, counting actors, dependencies, and other elements; indicators can be used to define metrics that measure model properties. Our definitions will make it possible to include some expert judgement if considered necessary to obtain more accurate results; in fact, we will see that indicators are highly customizable depending on both the knowledge available on the problem (expert judgement and current state of refinement of the model) and the effort to be invested in this process. Due to its structural nature, our framework is expressed in terms of the OCL [17]; operators such as `allInstances` and `select` suit well for working with model elements.

The paper is structured as follows. In section 2, we define the *i** framework using UML. In section 3, we introduce our framework for measuring *i** model properties. We analyse one particular property, predictability, in section 4, using the concepts introduced. Finally, we provide some comparison, conclusions and future work in sections 5 and 6.

2 A UML Definition of *i**

In this section, we introduce the *i** framework using the UML for defining rigorously its concepts. We think that this section is necessary because, as reported in [18], there are several variations in the literature for the *i** notation and thus we need to make explicit which constructs do we use in this paper and which properties do we assume.

Our *i** framework is based on the seminal Yu's proposal [2] with some minor simplifications. Yu proposes two types of models, each corresponding to a different abstraction level (see fig. 1): a *Strategic Dependency* (SD) model represents the intentional level and the *Strategic Rationale* (SR) model represents the rational level.

A SD model consists of a set of nodes that represent actors, and a set of dependencies that represent the relationships among them, expressing that an actor (*depender*) depends on others (*dependees*) in order to obtain some objective (*dependum*). Altogether form a network of knowledge that allows understanding "why" the system behaves in a particular way [19]. The dependum is an intentional element that can be a *resource*, *task*, *goal* or *softgoal* (see [2] for a detailed description).

A SR model allows visualizing the intentional elements into the *boundary* of an actor to refine the SD model with reasoning capabilities. Once SR models are built, the dependencies of the SD model may be linked to the appropriate intentional elements inside the actor boundary. According to their intentional meaning, some restrictions apply: goal dependencies can be assigned to goals and tasks in the dependee side; the same for task dependencies; and resource dependencies just to task dependencies.

The elements inside the SR model are decomposed accordingly to 2 types of links:

- *Means-end links* establish that one or more intentional elements are the *means* that contribute to the achievement of an *end*. The "end" can be a goal, task, resource, or softgoal, whereas the "means" is usually a task. There is a relation *OR* when there are several means, which indicate the different ways to obtain the end. The possible relationships are: *Goal-Task*, *Resource-Task*, *Task-Task*, *Softgoal-Task*, *Softgoal-Softgoal* and *Goal-Goal*. In *Means-end* links with a *softgoal* as end it is possible to specify if the contribution of the means towards the end is negative or positive; this label may also appear in softgoal dependencies.
- *Task-decomposition links* state the decomposition of a task into different intentional elements. There is a relation *AND* when a task is decomposed into more than one intentional element.

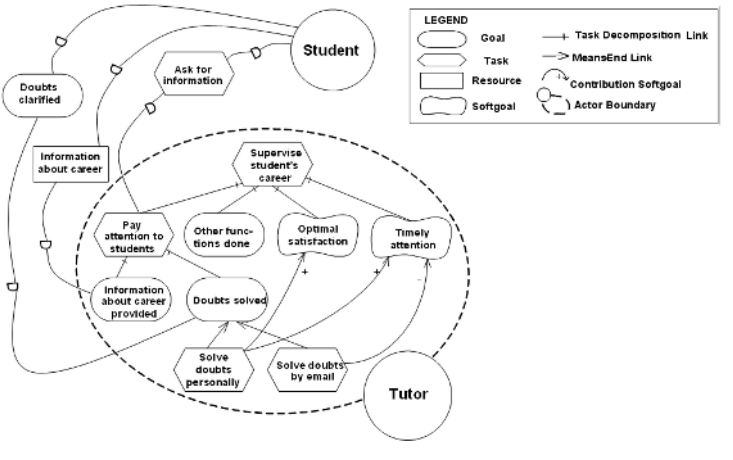

Fig. 1. Example of an *i** model for an academic tutoring system

SR models have additional elements of reasoning such as routines. A *routine* represents one particular course of action to attain the actor's goal among all the existing alternatives. The concept of routine appears in [2] but no notation is provided, so we use the similar notion of *scenario path* as defined in [20] based on the use case map concept appearing in GRL [21].

In Fig. 2 we show the conceptual model in UML, corresponding to our version of the *i** language; OCL constraints are not included for the sake of brevity. It is remarkable that dependencies are not defined as a ternary association; we have opted for composing two binary associations to facilitate the OCL expressions that we will write later in the metrics framework. We remark some modeling elements of interest: the *Model* class (singleton), which gives a name to the model; the *Node* class that provides a key to model elements; the *DependableNode* class, which models the intentional elements for which it is possible define dependencies, that is, actors and intentional elements of the SR model; and the *MeansEndContribution* and *SoftgoalContribution* classes, that differentiate means-end links and dependencies that involve softgoals.

As an additional point, it may be argued that, in order to formulate metrics to evaluate and eventually compare *i** models, it is necessary not only to rigorously define the semantics of the *i** elements that we use, but also how the models are built, since different people may build correct models very dissimilar in nature and of course too much diversity would make our quantitative framework difficult to apply. We have tackled this point in our previous work, by defining two similar, complementary methodologies for building *i** models, PR*i*M [22] and R*i*SD [23], depending on whether we create the model as a process reengineering exercise or from the scratch, respectively. Both methodologies define rules, checkpoints and procedures to guide model construction, therefore we may say that using them we can obtain models in a predictable and repeatable enough manner.

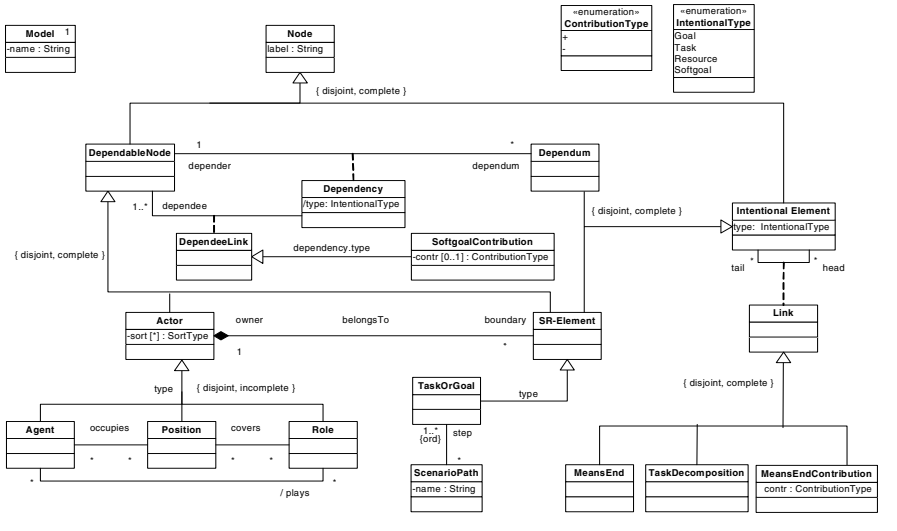

Fig. 2. A UML conceptual model for *i**

3 A Framework for Metrics on *i**

In this section, we explore the use of *structural indicators* that can be used to define *structural metrics* that measure the *properties* of an *i** model, i.e. those properties that depend on the form of the model and the types of its elements. Structural metrics are valuable for both analysing a highly abstract model of a system of any kind, composed basically by roles, and for comparing different feasible realizations of this abstract model (which take the form of actor models too, but composed basically by positions and agents) with respect to the most relevant criteria established in the modelled world. Some examples of properties that appear in the literature are:

- Ability, workability and commitment [2].
- Predictability, security, adaptability, coordinability, modularity and others [10].
- Correctness, completeness, verifiability, modifiability and traceability [24].

For a given property object of measure, it may be the case that all its elements (actors and dependencies) influence the indicator. However, it is also possible that just elements of some particular type affect this property. Furthermore, some individual elements may be identified as especially relevant for the property; in the most general case, all the elements may have a different weight in the indicator. We need then to take into account all these situations if we aim at having a widely applicable metrics framework.

For a given property, different indicators can be defined according to two criteria:

- Returned value. We distinguish among *numerical*, *logical* and *model-element* indicators. Numerical indicators return a value in the interval [0, 1]; this value measures the degree of accomplishment of some criteria. Logical indicators evaluate true or false, and are used to discern if a property is fulfilled or not. Model-element indicators return a (set of) model element (typically, actors, scenario paths or dependencies) that fulfils a property (e.g., scenario path that maximizes a given criteria, or set of actors that are greater than some threshold).
- Subject of measure. We can measure the whole model, individual elements or even groups of individual elements. In the first case we have *global* indicators, which produce a single value of any type. In the second case, we have *local* indicators, which compute a value for any element of a given type (actor, scenario path, etc., or even dependency of some type). In the third case, we talk about *group* indicators, which compute a value for any combination according to the grouping criteria (e.g., pairs of actors).

Therefore, given a property such as completeness, we may measure completeness of the model, of an element (e.g., an actor) or a group of related elements (e.g., all the actors of the model), with the purpose of deciding if they are complete or not, or to what extent they are complete (e.g., measuring the percentage of undefined elements) or obtaining the elements that are not complete yet. Some of the indicators can be built on top of the others, typically (but not always): logical and model-element indicators are defined in top of numerical ones; global and group metrics are defined on top of local ones.

In the next section we develop as example indicator for one property, predictability, following the concepts introduced in this section.

4 Analysing Predictability of *i** Models

Predictability is used in [10] as one of the properties of interest when analysing organizational styles. Its interest comes from the fact that "actors can have a high degree of autonomy depending on the way they undertake action and communication in their domains. It can be then sometimes difficult to predict individual actor characteristics as part of determining the behaviour of an organization at large" [10]. Therefore, discerning up to what extent the actors of a model are predictable may be useful for knowing more about a model.

From the several points of view we can take to analyse predictability, we opt by an external perception, i.e. how an actor perceives predictability of other actors. To be more precise, an actor is interested to know how predictable is the behaviour of those actors it depends upon, and this yields to select dependencies as the main construct of interest for defining the metrics. In the rest of the section, we first analyse predictability of individual dependencies and then we show several indicators that may be defined upon individual predictability. We will use OCL for measuring predictability on its different forms.

4.1 Predictability of Individual Dependencies

Yu states very clearly which is the degree of freedom bound to dependencies [2]:

- Goal dependencies. The dependee is free to, and is expected to, make whatever decisions are necessary to achieve the goal.
- Task dependencies. The depender makes the decisions, therefore the dependee cannot take a behaviour different than expected.
- Resource dependencies. They represent the finished product of some deliberation-action process, and it is assumed that there are no open issues to be addressed.
- Softgoal dependencies. The depender makes the final decision, but does so with the benefit of the dependee's know-how.

Therefore we may conclude that task and resource dependencies are totally predictable whilst goal and softgoal ones are not. Considering that 1 represents the highest predictability and 0 the lowest, we may define predictability of dependencies as:

```
context Dependency::predictability(): Real
 post: type = Task implies result = 1.0
 post: type = Resource implies result = 1.0
 post: type = Goal implies result = goalPredictability()
 post: type = Softgoal implies result = softgoalPredictability()
```

To define goal and softgoal predictability we may opt among different strategies:

- To assign a fixed weight to every single goal and softgoal dependency of the model. This is a very basic quantitative approach, with the assumption that the factor that rules predictability is the existence of a dependency, whilst its particular meaning or hidden intentionality is not so relevant.
- To provide weights to individual dependencies by expert judgement. This option yields to a qualitative reasoning issue appearing in the context of our quantitative procedure, which aligns with the point of view of [14]. This is the option to choose

when we have just the SD model, which happens in the first stages of organization analysis. For instance, if we apply our R*i*SD method [23], we build a SD model from the scratch and then perform analysis before proceeding further on. At this stage, we have just the most relevant elements in the model, which means that qualitative analysis is feasible in terms of cost. Experts may use techniques such as laddering [25] or AHP [26] as a help during their assessment.

- To find some suitable rationale for determining predictability. This alternative makes our approach basically quantitative; in fact, it may be defined in a total quantitative manner. This option seems the most appropriate when a SR model is available, which may happen in two ways: a) from the starting SD model, obtained e.g. applying R*i*SD, dependencies and actors are refined; b) the *i** model is synthesised from observation of the current organization and then the SR model exists from the very beginning, as we do in our PR*i*M method [22].

Fig. 3 summarizes these possibilities. It shows how expert judgement is needed in almost all possible combinations. Expert judgement is represented by underlined elements, i.e. values or functions that must be provided in order to build the metrics.

We focus on the last case, which requires more decisions to take. Considering softgoal dependencies, we decompose their evaluation into two factors. First, a factor bound to the depender actor, which represents how capable it is to take predictable decisions when resolving softgoals; we consider this factor bound to actors' ability and not to individual softgoal dependencies. Second, a factor bound to the dependency, which represents the available know-how with respect to the given dependum. For the OCL expression, we must take into account that the depender can be an actor or an SR element, and in the second case we obtain its owner; a `let` expression makes this easier to write:

```
context Dependency::softgoalPredictability(): Real
pre: type = Softgoal
let ownerActor(x: DependableNode): Actor =
           if x.oclIsTypeOf(Actor) then x else x.owner in
post: result = ownerActor(depender).dependerExpertise()
               * knowHow()
```

Depender expertise may be dealt with by two different strategies: considering expert judgement to weight individual actors, or else to agree a given weight for all the actors. Concerning available know-how, we may define a strategy for measuring predictability using the SR model as follows. We define the know-how as the number of dependees that state a contribution value to the dependum. Then, we need a function such that: 1) when the number of contributions is 0, the function is also 0 (worst predictability because the dependees do not know how to contribute to the softgoal); 2) as the number of contributions grow, the function tends to 1 (best predictability).

An easy, problem-independent way to define the function is $1 - (slope/n+1)$, being n the number of known contributions for the softgoal dependum and *slope* a constant (defined as an attribute of the model) that determines the slope of the function (see fig. 4, left). Another possibility is to define a utility function [27] such that we define a straight line from 0 to the maximum number of dependee contributions to a softgoal dependum that exists in the model (see fig. 4, right).

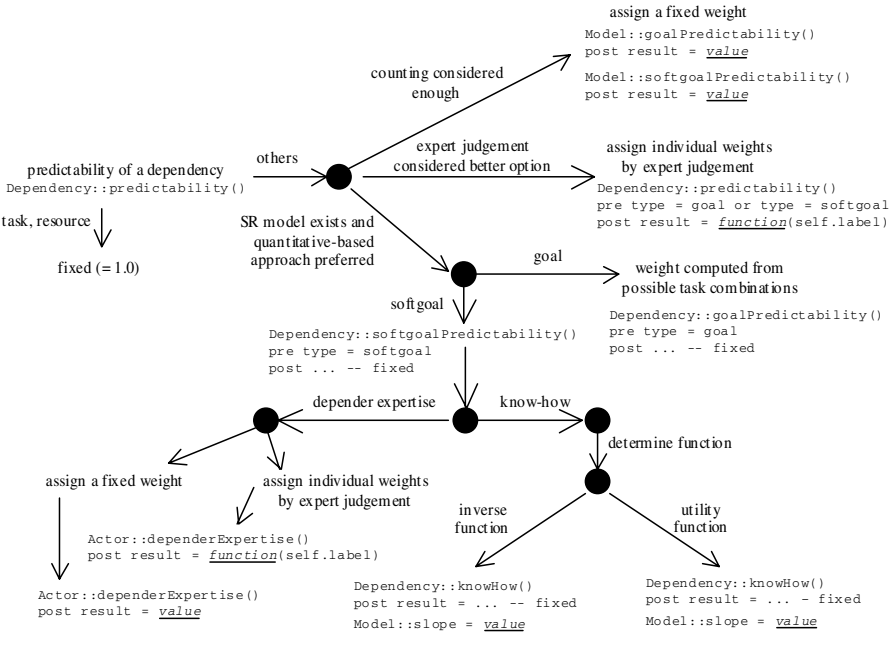

Fig. 3. Procedure for determining the *Predictability* of individual dependencies

Fig. 4. 2 different possibilities of know-how functions: left, inverse function with *slope* = 1; right, utility function (*n* = maximum number of dependee contributions to softgoal dependum)

The resulting OCL definition for the first case is:

```
context Dependency::knowHow(): Real
pre: self.type = Softgoal
 let theModel: Model = Model.allInstances()->any() in
 let contributionsToSoftgoalDep(d: Dependency): Integer =
       d.dependeeLink.oclAsType(SoftgoalContribution)->
       select(contr->notEmpty())->size() in
post: result = 1 - theModel.slope /
                  (contributionsToSoftgoalDep(self)+1)
```

Fig. 5 presents an example of this case. It is an excerpt of a model for a distance learning environment. The dean has as one of her goals to achieve academic quality, and for this goal she depends on teachers and tutors for having *Good Course Dynamics*.

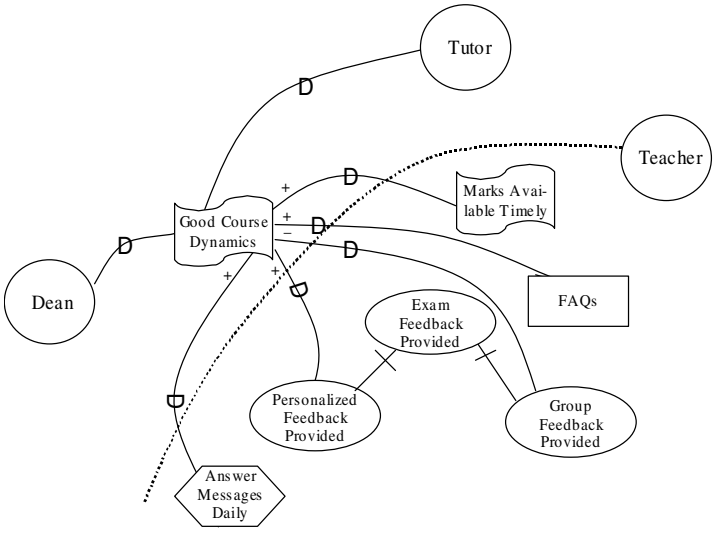

Fig. 5. Distance learning environment model: predictability of softgoal dependencies

There are several ways in which teachers may contribute positively to this softgoal: publishing exams' marks timely, answering students' messages daily and making FAQs lists available. An important issue that influences course dynamics in distance learning is the feedback that teachers provide to students about their exams. There are roughly two strategies: sending personalized messages to students commenting their mistakes, or giving group support by making public the solution and the evaluation criteria, and sending personalized information just on demand. The first strategy is considered to impact positively into the dynamics of the course, but not the second. Concerning tutors, it has not been investigated yet how they contribute to course dynamics. Thus, we have 5 contributions to the softgoal dependency; applying the definition above with `Model.slope` = 1, `GoodCourseDynamics.knowHow()` = 0,83. Since the dean is a highly strategic actor, we may assume that her `dependerExpertise()` = 1,0 and `GoodCourseDynamics.softgoalPredictability()` = 0,83.

Concerning goal dependencies, unpredictability depends on how many ways the dependees have to fulfil the goal. As stated in section 2, a goal dependency may have as intentional elements on the dependee side just goals and tasks. In both cases, the different task combinations that we may find descending by the goal or task, using means-end and tasks decompositions, are computed: the more combinations are found, the less predictable is the dependee with respect to that dependency. It is worth to remark that if the dependency involves more than one dependee, unpredictability appears from the very beginning, because this means that there are many ways to attain the goal dependum. Also we have to deal with the case that the dependee is not a SR element but an actor, which means that the dependency has not been assigned yet to an intentional element and thus unpredictability is maximized (i.e., equals to 0).

Similarly to the case above, a problem-independent function can be defined as the inverse of the number of combinations. We outline the corresponding OCL function, not including the function that computes the number of combinations:

```
context Dependency::goalPredictability(): Real
pre: self.type = Goal
let nbTaskCombinations(d: Dependency) = … in
post: nbTaskCombinations(self) = 0 implies result = 0
post: nbTaskCombinations(self) > 0 implies
              result = 1 / nbTaskCombinations(self)
```

Fig. 6 presents an example of this case focusing on how exam evaluation feedback is provided. The two goals introduced in fig. 5 are refined. The most general goal that appears, *Evaluation Feedback Provided*, is the dependee of the student's goal *Feedback from Exams Acquired*. Since this goal has two means-end decomposition (which are implicitly OR-ed, see section 2), two different ways to provide feedback are being stated. Therefore, the evaluation for this dependency is 1 / 2 = 0,5. Effects of unpredictability are clear if we analyse how the elements that appear in the decomposition relate to other model elements. For instance, *Personalized Feedback Provided* has a negative contribution to the *Personal Workload kept Low* softgoal that the teacher has. This contribution is stating that deciding among *Personalized* or *Group Feedback Provided* depends on what the teacher considers a reasonable threshold for her workload, and since this is out of the student's control, predictability gets damaged.

As a final remark, we would like to point out that the obtained indicator for dependency predictability is highly customizable (therefore reusable and repeatable); key points are: does the SR model exist or not?, do I really need expert judgement or do I keep my approach purely quantitative?, if expert judgement is chosen, do I prefer to weight individual elements or do I assign the same weight to all of them? The procedure depicted at fig. 3 shows clearly the needed steps; there we represent the information required during the process by underlined italics in the body of OCL expressions.

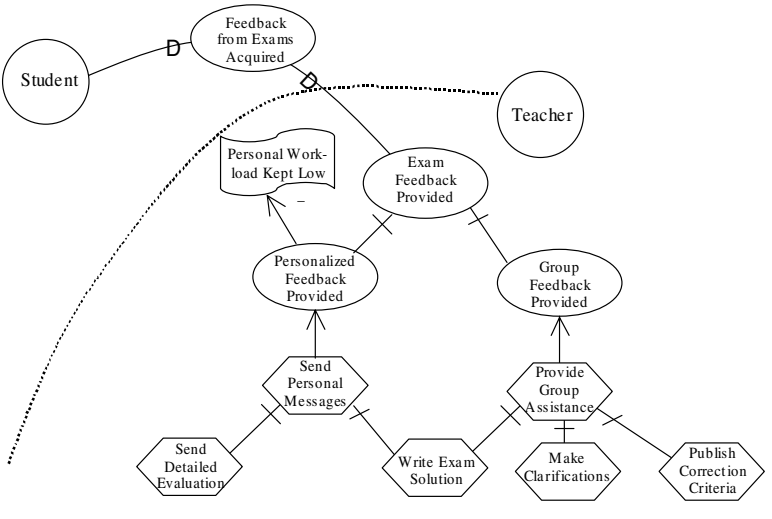

Fig. 6. Distance learning environment model: predictability of goal dependencies

4.2 Indicators for Predictability

Next we talk about the different indicators that may be defined on top of dependency evaluation. The dimensions presented in section 3 can be used. Of particular interest is the dimension about the subject of measure. We present 3 feasible possibilities:

- We may analyse predictability of actors. We may adopt two different points of view: how predictable an actor perceives its environment, and how predictable an actor looks to its environment. In the first case, we group the dependencies in which the actor is a depender, whilst in the second case, we group the dependencies in which an actor is a dependee. For instance, for the first point of view we obtain:

```
context Actor::perceivedPredictability(): Real
  let actorDependencies(a: Actor): Set(Dependency) =
        Dependency.allInstances()->
        select(d | d.depender = a or d.depender.owner = a) in
  post: actorDependencies(self)->size() = 0 implies result = 1
  post: actorDependencies(self)->size() > 0 implies result =
             actorDependencies(self).predictability()->sum()
             / actorDependencies(self)->size()
```

- Another possibility is to concentrate on scenario paths as representative of business processes. A scenario path is composed by steps that are tasks or goals. Each step is either decomposed inside the boundaries of the actor or as depending on external actors; these two cases rule the OCL decomposition below. In both cases, predictability depends on the number of task combinations that exist to carry out the step:

```
context ScenarioPath::predictability(): Real
  post: result = step.predictability()->sum() / step->size()

context TaskOrGoal::predictability(): Real
  let dependsUpon(): Boolean =
                  self.dependency[depender]->notEmpty() in
  post: dependsUpon() implies
        result = dependency[depender].predictability()->sum()
                 / dependency[depender]->size()
  post: not dependsUpon() and nbTaskCombinations() = 0
           implies result = if type = task then 1 else 0
  post: not dependsUpon() and nbTaskCombinations() > 0
           implies result = 1 / nbTaskCombinations(self)
```

  being `TaskOrGoal::nbTaskCombinations()` a function that computes the number of task combinations for that task or goal, defined analogously to `Dependency::nbTaskCombinations()` introduced in section 4.1.

- As done in [10], we may define predictability for the whole model, obtaining therefore a single value. They use this property to compare different organizational patterns such as joint venture, structure in 5, and others:

```
context Model::predictability(): Real
  post: result =
           Dependency.allInstances().predictability()->sum()
          / Dependency.allInstances()->size()
```

Concerning the second dimension, we can use these numerical indicators to obtain boolean or model elements ones, allowing e.g.: finding out if strategic actors exceed

some threshold; given two models, which one is the most predictable; ordering all the actors in terms of predictability; checking that scenario paths are fully predictable; etc.

5 Comparison with Related Work

In the introduction we have mentioned the existence of qualitative approaches for analysing *i** models but, to the best of our knowledge, there is not much related work from a quantitative point of view. The most remarkable proposal in this area is part of the AGORA method [24] that provides techniques for estimating the quality of requirements specifications in a goal-oriented setting. In fact, AGORA puts more emphasis in the analysis of the AND/OR graph resulting from decomposition than in the kind of analysis that has been the focus of this paper. Therefore, comparison is not really possible and in fact, we could think of using AGORA and our approach jointly. Also, it is worth mentioning the work by Sutcliffe and Minocha [28] which proposes the analysis of dependency coupling for detecting excessive interaction among users and systems. They use expert judgement to classify the dependencies of the system in a qualitative scale and then define a metric on the model that use to compare alternative scenario. This metric for coupling is a good example of structural metric and we can check that it is definable using our framework in a straightforward way.

On the other hand, we have already mentioned some work on combining quantitative and qualitative analysis of *i** models for finding assignment of labels to nodes and determine its propagation in goal graphs. In [15], qualitative reasoning is based on a sound and complete set of rules that determine backward propagation in a goal-oriented, SR-like graph. The rules combine 4 different types of relationships among goals, depending on whether a goal fully/partially satisfies/denies another goal. Quantitative reasoning consists on assigning weights to those relationships. In [16], assignment of labels to goals, and the use of these labels to propagate values both forward and backwards, become the subject of study. The main difference of these approaches with the work presented in this paper is the interest of the analysis. Whilst [15, 16] focus on goal satisfaction, our work is more interested in the analysis of structural properties of the model. Therefore, we can say again that both approaches are not exclusive but complementary. The way the authors encode the qualitative framework is a good example of how knowledge may be represented in both a simple and accurate way, and it could be thought that this description style of qualitative knowledge may be used also in our context.

6 Conclusions and Future Work

We have presented a framework for the definition of structural metrics for agent-oriented models using the *i** language. The metrics are bound to properties of the system model, which usually represent correctness concerns, organizational issues or information systems requirements. The framework considers the definition of indicators organized according to two dimensions (returned value and subject of measure). The indicators are customised to use expert judgement as considered necessary,

although we may say that they are basically quantitative in nature. We have shown with an example how these indicators may be used to find out properties of the system.

The most relevant characteristics of our approach are:

- Accuracy. We have provided a UML definition of *i** models that is used as a baseline upon which we have build our framework. Indicators and metrics are expressed with the OCL. The approach is complemented with two methodologies to drive the construction of *i** models in a consistent way.
- Expressiveness. The use of the OCL allows expressing metrics both in a comfortable and expressive way. Comfortability comes from the easy of structuring inherent to object-orientation, which has been shown in the predictability example.
- Sensitivity. Metrics can be defined more or less accurately depending on: 1) the expert judgement available; 2) the state of refinement of the model; 3) the effort we want to invest in model analysis. Therefore, we have a highly configurable framework that allows defining metrics in several ways (see fig. 3 as an example).
- Easy tool support. The form that our framework takes allows implementation of tool support to drive indicators definition, model edition, generation of alternatives and evaluation of models. We have a first prototype [29] which uses metrics patterns as a way to improve productivity (although it is not based in the OCL). Tool-support may also be used to customise the indicators in a particular setting by means of wizards that basically asks for the required information following a data flow such as the one presented in fig. 3.
- Reusability. The indicators and metrics obtained are independent of the domain and therefore applicable to any model.

The framework presented here has been analysed with a few properties such as the one presented in this paper. However, a proper validation plan has not been yet executed. A long-term goal is to apply the framework to large-scale case studies but, in the meantime, we are validating with respect to some exemplars that are widespread in the *i** community, such as the one of predictability presented in this paper. Validation is necessary also to gain more understanding on the property being analysed and then to define more accurately OCL formula. In our example, this kind of validation would help to know if the strategies applied to define goal and softgoals are accurate enough and to compare different strategies. For instance, an alternative to the definition in the case of goals would be to take into account the depth of task decompositions: the deeper the decomposition appears, the less it affects predictability. A thorough validation plan would allow choosing which alternative is better.

It may be said that one of the limitation of our approach is the need to elicit expert judgement at some extent. However, we should remark that the involvement of experts is highly customizable. For instance, we have shown in our case study that this expert judgement may be kept reduced if required by prioritising the quantitative part of our framework (see fig. 3). In any case, we do think that some degree of qualitative reasoning is necessary to obtain information that is accurate with respect to some departing assumptions (which encode the knowledge of the expert). We remark also that expert judgement will usually be necessary in the context of comparison of alternatives that has been cited in the introduction, because given two alternatives, in the general case some metrics will behave better in one model and some in other, therefore expert judgement is needed to prioritize appropriately.

We have identified several ways to proceed along in this line of research. For making our proposal useful, we remark the following:

- Construction of a catalogue of reusable indicators and metrics. Basically in three directions: 1) model-related properties (predictability is one example); 2) organizational-related properties (such as segregation of duties [30]); 3) properties addressing non-functional aspects such as security, efficiency and so on.
- Identification of patterns for indicators and metrics. We have realized that most of the indicators and metrics definitions apply similar rules over and over. In [31] we have identified some patterns that capture some of these situations and we plan to enlarge the catalogue.
- Better tool-support. We plan to enlarge our current prototype and adapt it to the OCL as the language for metrics definition.
- Integration of the framework with other proposals. In particular, we are especially interested in using this framework in the analysis of system architectures [8, 32]. We think that metrics on goal-oriented models may provide first-cut criteria for classifying candidate architectures.

Acknowledgements

This work has been done in the framework of the research project UPIC, ref. TIN2004-07461-C02-01, supported by the Spanish Ministerio de Ciencia y Tecnología. The author wants to thank Gemma Grau for her valuable comments.

References

[1] A. Dardenne, A. van Lamsweerde, S. Fickas. "Goal-directed Requirements Acquisition". *Science of Computer Programming*, 20, 1993.

[2] E. Yu. *Modelling Strategic Relationships for Process Reengineering*. PhD. thesis, University of Toronto, 1995.

[3] J. Castro, M. Kolp, J. Mylopoulos. "Towards Requirements-Driven Information System Engineering: The Tropos Project". *Information Systems*, 27, 2002.

[4] E. Yu. "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering". *Procs. 3rd Intl. Symposium in Requirements Engineering* (ISRE), 1997.

[5] A. van Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". *Procs. 5th Intl. Symposium on Requirements Engineering* (ISRE), 2001.

[6] M. Lorenz, J. Kidd. *Object-oriented software metrics: a practical guide*. Prentice-Hall, 1994.

[7] S.R. Chidamber, C.F. Kemerer. "A Metrics Suite for Object-Oriented Design". *IEEE Transactions on Software Engineering*, 20(6), 1994.

[8] L. Baas, P. Clements, R. Kazman. *Software Architecture in Practice, 2nd edition*. Addison-Wesley, 2003.

[9] L. Chung, B. Nixon, E. Yu, J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[10] M. Kolp, J. Castro, J. Mylopoulos. "Organizational Patterns for Early Requirements Analysis". *Procs. 15th Intl. Conf. on Advanced Information Systems Engineering* (CAiSE), 2003.

[11] M.B. Mile, A.M. Huberman. *Qualitative Data Analysis*. Sage Publications, 1994.
[12] T.A. Schwandt. "Solutions to the Paradigm Conflict: Coping with Conflict". *Journal of Contemporary Etnography*, 17(4), 1989.
[13] M.Q. Patton. *Qualitative Evaluation and Research Methods*. Sage Publications, 1990.
[14] R.B. Johnson, A.J. Onwuegbuzie. "Mixed Methods Research: A Research Paradigm Whose Time Has Come". *Educational Researcher*, 33(7), 2004.
[15] P. Giorgini, J. Mylopoulos, E. Nicciarelli, R. Sebastiani. "Formal Reasoning Techniques for Goal Models". *Procs. 21st Intl. Conference on Conceptual Modeling* (ER), 2002.
[16] R. Sebastiani, P. Giorgini, J. Mylopoulos. "Simple and Minimum-Cost Satisfiability for Goal Models". *Proceedings of 16th Conf. on Advanced Information Systems* (CAiSE), 2004.
[17] Object Management Foundation (OMG). "UML 2.0 OCL Specification", available at www.omg.org/docs/ptc/03-10-14.pdf, 2003.
[18] C. Ayala, C. Cares, J.P. Carvallo, G. Grau, M. Haya, G. Salazar, X. Franch, E. Mayol, C. Quer. "A Comparative Analysis of *i**-Based Goal-Oriented Modeling Languages". *Procs. Intl. Workshop on Agent-Oriented Software Development Methodology* (AOSDM), 2005.
[19] E. Yu. "Understanding 'why' in software process modeling, analysis and design". *Procs. 16th Intl. Conference on Software Engineering* (ICSE), 1994.
[20] L. Liu, E. Yu, J. Mylopoulos. "Analysing Security Requirements as Relationships among Strategic Actors". *Procs. 2nd Symposium on Requirements Engineering for Information Security* (SREIS), 2002.
[21] D. Amyot. "Use Case Maps Quick Tutorial Version 1.0". Available at http://www.usecasemaps.org/pub/UCMtutorial/, last accessed Nov. 2005.
[22] G. Grau, X. Franch, N. Maiden. "A Goal-Based Round-Trip Method for System Development as Business Process Reengineering". *Procs. 11th Intl. Workshop on Requirements Engineering: Foundation for Software Quality* (REFSQ), 2005.
[23] G. Grau, X. Franch, E. Mayol, C. Ayala, C. Cares, J.P. Carvallo, M. Haya, F. Navarrete, P. Botella, C. Quer. "R*i*SD: A Methodology for Building *i** Strategic Dependency Models". *Procs. 7th Intl. Conf. on Software Engineering & Knowledge Engineering* (SEKE), 2005.
[24] H. Kaiya, H. Horai, M. Saeki. "AGORA: Attributed Goal-Oriented Requirements Analysis Method". *Procs. 10th Joint Conference on Requirements Engineering* (RE), 2002.
[25] T.J. Reynolds, J. Gutman. "Laddering Theory, Method, Analysis and Interpretation". *Journal of Advertising Research*, vol. 28, 1988, pp. 11-31.
[26] T.L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, 1990.
[27] R. Keeney, H. Raiffa. *Decision with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, 1993.
[28] A. Sutcliffe, S. Minocha. "Linking Business Modelling to Socio-technical System Design". *Procs. 11th Intl. Conf. on Advanced Information Systems Engineering* (CAiSE), 1999.
[29] G. Grau, X. Franch, N. Maiden. "REDEPEND-REACT: an Architecture Analysis Tool". *Procs. 13th Intl. Conference on Requirements Engineering* (RE), 2005.
[30] A. Burt. "Internal Controls and Segregation of Duties". *UF Bridges Project*, University of Florida, 2004.
[31] X. Franch, G. Grau, C. Quer. "A Framework for the Definition of Metrics for Actor-Dependency Models". *Procs. 12th Intl. Conf. on Requirements Engineering* (RE), 2005.
[32] P. Grünbacher, A. Egyed, N. Medvidovic. "Reconciling Software Requirements and Architectures - The CBSP Approach". *Procs. 5th Intl. Symposium on Requirements Engineering* (ISRE), 2001.