On the Lightweight Use of Goal-Oriented Models for Software Package Selection

Xavier Franch

Universitat Politècnica de Catalunya (UPC),
UPC – Campus Nord, Omega-122, 08034 Barcelona (Spain)
franch@lsi.upc.edu
http://www.lsi.upc.edu/~gessi

Abstract. Software package selection can be seen as a process of matching the products available in the marketplace with the requirements stated by an organization. This process may involve hundreds of requirements and products and therefore we need a framework abstract enough to focus on the most important factors that influence the selection. Due to their strategic nature, goal-oriented models are good candidates to be used as a basis of such a framework. They have demonstrated their usefulness in contexts like early requirements engineering, organizational analysis and business process reengineering. In this paper, we identify three different types of goal-oriented models useful in the context of package selection when some assumptions hold. *Market segment models* provide a shared view to all the packages of the segment; *software package models* are derived from them. The selection can be seen as a process of matching among the *organizational model* and the other models; in our proposal this matching is lightweight, since no model checking is performed. We define our approach rigorously by means of a UML conceptual data model.

1 Introduction

In the last years, software-package (SP)-based information systems have become not the exception but the rule in the software-based solutions for managing the informational resources of organizations worldwide. This is true for both SP that have a visible impact in the services offered by the organization, such as ERP and CRM systems, and SP that take care of the daily functioning of the organization, such as mail or meeting scheduler systems, and security-related tools.

Successful SP-based system development requires a unique set of activities to be performed, among which we find the selection of the SP themselves. This activity is becoming increasingly more and more critical, due to the ever-growing nature of the SP market, both in the variety of market segments (MS) available and the SP offered therein. As a consequence, several SP selection methodologies, processes and techniques have been formulated [1, 2, 3, 4], which propose different ways of eliciting requirements and evaluating SP in the context of package selection. In spite of this variety, we think that these proposals do not address in an effective way 2 fundamental questions:

O. Pastor and J. Falcão e Cunha (Eds.): CAiSE 2005, LNCS 3520, pp. 551 – 566, 2005.

- *MS variety*: How can we describe the different existing MS in a way such that an organization becomes aware of their applicability in a particular selection problem?
- *SP proliferation*: How can we describe the great deal of SP belonging to a MS in a way such that their comparison can be made uniformly?

The answer to these questions should take into account the following facts:

- *Complexity of the problem*: Not only the SP market is large, also the number of requirements can be very high in typical selection processes for information systems, as well as the number of quality factors that characterise a particular kind of SP. As a consequence, we cannot aim at considering the detailed system requirements nor all the SP quality factors from the beginning.
- *Intertwining of requirements elicitation and SP evaluation*: We need ways to facilitate the identification and reformulation of requirements from the observation of the SP market.
- *Evolution of the SP market*: New SP, and versions of existing SP, appear continuously, preventing therefore the use of exhaustive descriptions of these SP.

A natural way for answering the raised questions considering the enumerated facts is to rely on goals in the early stages of SP selection, rather than on detailed requirements, specifications or quality models [5]. In fact, this idea aligns with the observation that current goal-oriented analysis methods and languages such as KAOS, *i**, GRL or TROPOS [6, 7, 8, 9] are widespread in the requirements engineering community for the refinement and decomposition of the customer needs into concrete goals in the early phase of the requirements specification [10]. In our context, goals and goal-based analysis can be used to model organizational needs, to identify which MS are interesting for the problem at hand and to make a screening of the candidate SP belonging to these identified MS. Of course, detailed requirements and evaluation of interesting quality factors should be made, but at a later stage when the whole scenario has been clarified and the solution space has been pruned down to a small set of candidate SP.

Once the applicability of goals to SP selection has been stated, we face a problem. Existing goal-based methodologies for software systems construction (e.g., TROPOS, KAOS methodology) do not handle explicitly the particularities of SP selection. In other words, they seem to be addressed more to support the development of bespoke software rather than the construction of systems based on the composition of SP.

In this paper, we use of goal-oriented models for supporting SP selection. More precisely, we propose dependency-based goal-oriented models, i.e. models that state which are the actors involved in the information system and which actor goals depend on which other actors. We use *i** Strategic Dependency models [7] with this purpose. We distinguish 3 types of models: 1) organizational models for modelling the needs of the organization; 2) MS models for modelling the services that a particular MS addresses; and 3) SP models for modelling the services that a SP belonging to a particular MS offers. We show how the relationships among these 3 models can be defined and how do they help to answer the 2 fundamental questions above taking the 3 identified facts into account (see fig. 1). We provide a rigorous description of all the models and their relationships through a UML conceptual model [11], including a complete set of constraints that are presented in textual form (see [12] for their OCL form).

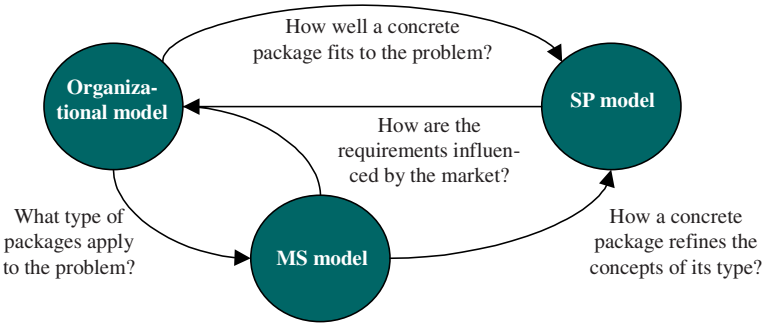

Fig. 1. Using goal-oriented models in the selection process

2 Scope of the Proposal

This research has been motivated by a collection of selection processes in which we have been recently involved, some of them reported elsewhere [13, 14, 15]. These experiences have resulted in the identification of the questions and facts presented in the introduction and they have yield to the adoption of goal-oriented models as explained in this paper. Therefore, we may say that our proposal has been validated through experiments with the following assumptions:

- **Usefulness:** The MS addressed is a segment of general interest. This means that a great deal of organizations need to select SP from this MS. Some examples are: communication infrastructure (mail servers, videoconference, etc.), ERP systems, security-related systems, etc.
  - *Consequence*: The number of selection processes that take place in this MS will be high and then reusability of the models likely to occur.
- **Variety:** There are a lot of SP offered in this MS that are competitive enough. The first part of the statement is a direct consequence of the previous point: if there is a universal need, of course lots of products will be offered. The second point excludes some particular MS like operating systems in which options are few and the analysis we propose does not pay.
  - *Consequence*: The number of SP to be analysed in selection processes in this MS will be high and then we need a common framework as a basis for analysing and comparing them.
- **Size:** The type of SP offer a great deal of features. This makes SP understanding more difficult, time-consuming and cumbersome. SP such as ERP systems are typical examples, whilst time or currency converters are not.
  - *Consequence*: The concepts embraced by the type of SP are much and then light descriptions focusing in the most fundamental notions are needed for pruning the set of candidates in a cost-effective manner before detailed evaluations occur
- **Continuity:** The selection activity is monitored by an organization that accumulates experience from past selection processes. This organization will find

valuable to have means to transfer knowledge from one experience to another and to assist their clients in the maintenance of their SP-based information system.
  - *Consequence*: The organization will be involved in an increasing number of selection processes being able to transfer knowledge while improving its skills.

- **Uncertainty:** The starting requirements stated by the organization are vague, incomplete and often ill-justified. Sometimes the organization even does not know exactly which MS is addressing to.
  - *Consequence*: The statement of the organizational departing needs must cope this incompleteness focusing in the strategic underlying needs instead of the concrete requirements. Furthermore, it should be possible to add new needs from the SP analysis, which means that both organizational and software models should be described similarly.

The use of our approach in SP selection processes with different assumptions would require further experimentation.

3 Background: Strategic Dependency Models in the *i** Notation

An *i* Strategic Dependency* (*SD*) *model* comprises two types of elements, *actors* and *dependencies* among them. Actors are intentional entities, that is, there is a rationale behind the activities that they carry out. Dependencies connect source and target actors, called *depender* and *dependee*. Altogether they form a network of knowledge that allows understanding "why" the system behaves in a particular way [16].

For our purposes, actors play roles that are abstract characterizations of a behaviour within some specialized context or domain of endeavour [7]. We distinguish four types of actors: human, organizational, software and hardware. We find convenient to allow the definition of hierarchies using the typical is-a construct.

SD models express dependencies using four main types of dependency link (see fig. 2). For *goal dependencies* the *depender* depends upon the *dependee* to bring about a certain state in the world. A *goal* represents a condition or state of the world that can be achieved or not. For *task dependencies*, the *depender* depends upon the *dependee* to carry out an task. A *task* is a detailed description of how to accomplish a goal. For *resource dependencies*, the *depender* depends upon a *dependee* for the availability of an entity. *Resources* can be physical or informational, and considered as the availability of some entity or even the finished product of some process. For *soft goal dependencies*, the *depender* depends upon the *dependee* to perform some task that meets a non-functional requirement, or to perform the task in a particular way.

Fig. 3 presents a the class diagram from a UML conceptual model that formalizes the concept of SD model; the integrity constraints required here are: (IC1) an actor shall not be a specialization of itself; (IC2) specialization shall preserve the type and model of an intentional element; (IC3) the depender and dependee of a dependency shall belong to the same model and shall not be the same neither one a specialization of the other. We include in the diagram the 3 specializations of SD models enumerated in the introduction that will be presented next.

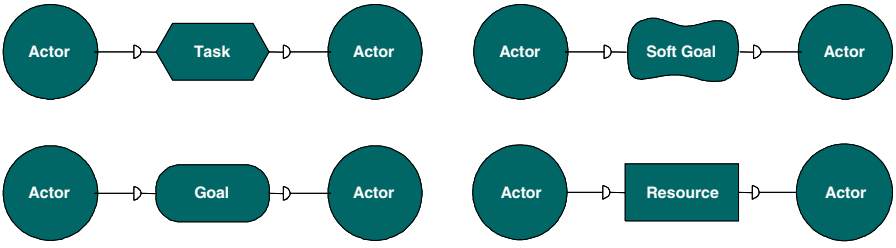

Fig. 2. Graphical representation of *i** Strategic Dependency actors and dependencies

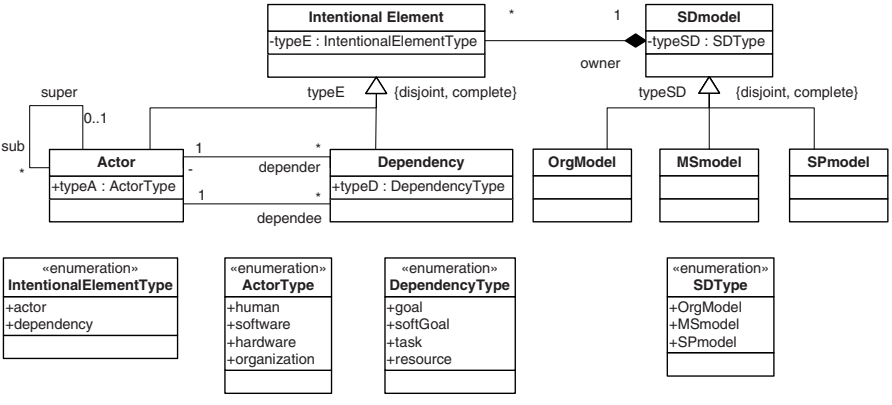

Fig. 3. A UML conceptual model representing SD models and their specializations

4 A Goal-Oriented Model for Stating Organizational Needs

As mentioned in the introduction, the classical use of goal-oriented models in the context of information systems is to provide an early specification of the system-to-be focusing on strategic aspects [5, 10]. We consider that the first set of temptative, incomplete and high-level requirements in a selection process is the formulation of such a model, which is obtained using some methods widespread in the requirements engineering community such as GBRAM [17]. Since this is not a contribution of our work, in this section we just introduce an example that we are going to use in the rest of the paper.

Let's consider an organization concerned with ensuring data integrity when data interchange take place with human users. Fig. 4 presents its intentional elements. Users who interchange data require the organization to keep their information preserved (D1) and not to send them undesired information of any kind (D2). Since users are aware that they may submit incorrect information, they also require the organization to warn them in this case (D3). Information checking shall be transparent to users (D5). At its turn, the organization just requires users not to submit hazardous information of any kind (D4). On the other hand, the organization needs support to discern whether its managed information contains unwanted data or not (D6). This gives light to a third actor –a data integrity expert, capable of informing whether the information suffers from some hazards or not.

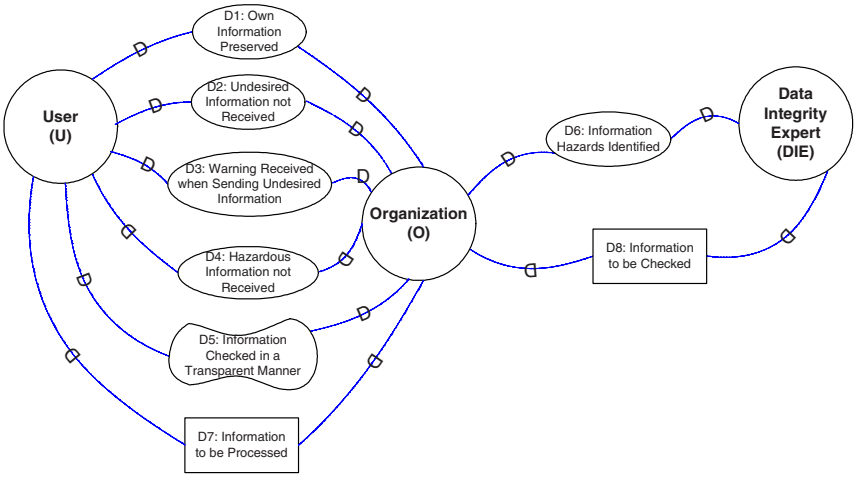

Fig. 4. SD model for an organization with data integrity needs

5 *i** SD Models for the Strategic Description of Market Segments

*i** SD models are appropriate artefacts to describe the services that SP belonging to a MS offer to organizations. When used with this purpose, we call them *market segment SD models*, *MS models* to abbreviate. MS models are characterised by the following properties (see fig. 5):

- There is a designated actor that represents the type of SP characterising a MS.
- There is at least another actor representing the main SP beneficiary, that most of the cases is an organization. Sometimes, this actor is specialized into different subactors using an "is-a" construct.
- Often there are some other actors bound to the type of SP of interest, for instance:
  - Abstract software actors, standing for software that may require connection to the type of SP of interest but that is unknown in advance.
  - Human users representing stakeholders have particular interests that are distinguishable from those organizational ones (e.g., end users).
  - SP administrator, in charge of administration duties when packages are large.
- The dependencies among the actors are kept to the minimum extent. There are two main reasons for this decision:
  - The model shall be general enough to be inclusive, i.e., every single SP of the modelled type must be compliant with that model.
  - SP selection will include a matching process involving this model, and in this context it is convenient to handle models of a reasonable size.
- Most dependencies are goal dependencies. Soft goal dependencies are restricted to crucial non-functional requirements on the type of SP. Resource depen-

dencies are restricted to fundamental concepts of the MS that stand for some kind of data. Task dependencies are restricted to activities that are out of the control of the depender.

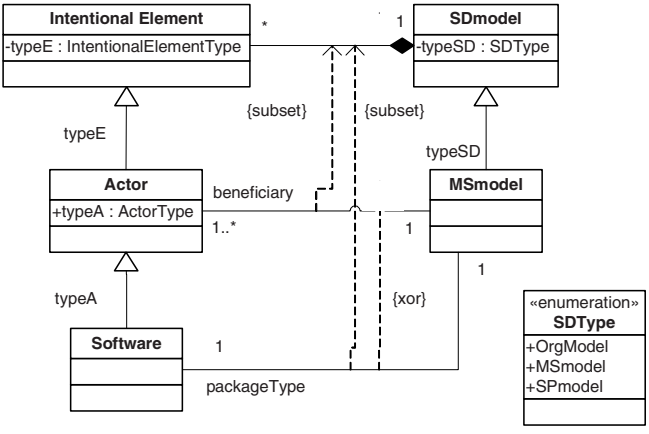

Fig. 5. A UML conceptual model representing MS models

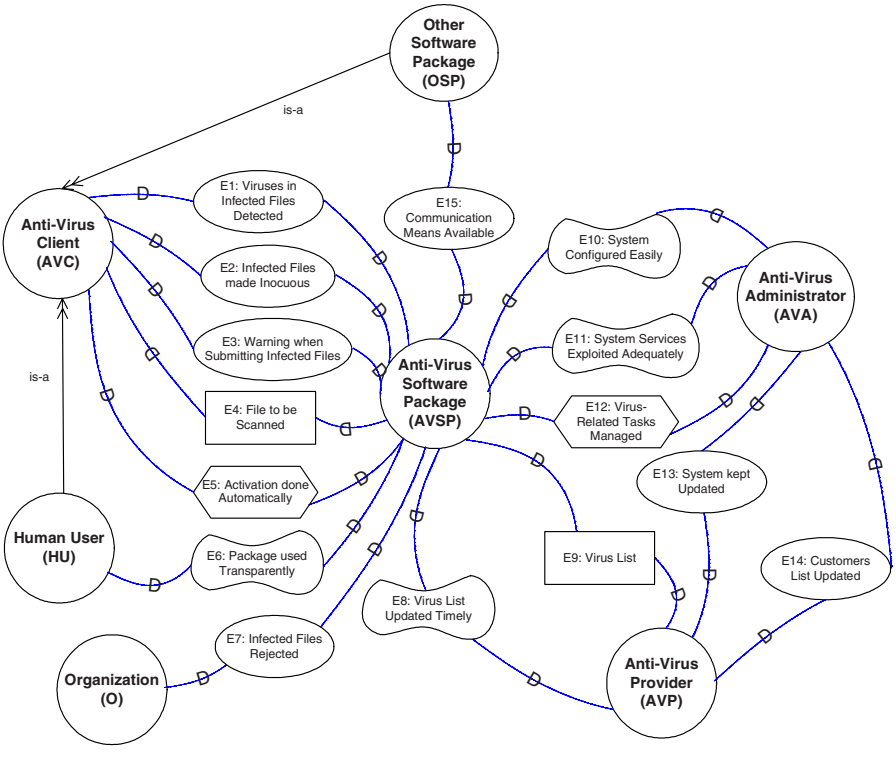

Fig. 6. An SD model for the market segment of anti-virus software packages

A MS that can be used for satisfying the organizational model presented in section 4 is the one of anti-virus software packages (AVSP) (see fig. 6). This segment fulfils the requirements outlined in section 2 in order to be a valid target of our proposal.

The guidelines concerning dependencies can be checked in the figure. For instance, some non-functional properties such as efficiency are not required at any point –of course we would like the solution to be efficient, but we consider that for AVSPs, efficiency is not as critical as for other types of packages and therefore it does not appear in this highly strategic model. The two resource dependencies show the two most significant data concepts: the target of the SP (the file to be scanned, E4) and the object that threatens the file (the viruses, E9). Task dependencies reveal that AVSP activation (E5) and task management (E12) occur in a particular manner.

6 Software Package Description Using *i** SD Models

As mentioned, a MS model exhibits a fundamental property: every single SP of this segment shall be compliant with this model. This helps to tackle one of the fundamental questions identified in the introduction, namely SP proliferation. However, each package may have its own peculiarities with respect to the services offered to and requested from its environment. We propose to use again *i** SD models to represent them and, in particular, we claim that the description of a particular SP shall be build starting from the MS model. We call this model *software package SD model*, or *SP model*. The process of obtaining a SP model from a MS one is called *derivation*.

In SP models, intentional elements can be of 3 types depending on their relationship with MS models:

- *Kept elements* are those already appearing in the MS model. They stand for intentional elements that are not further refined for any of the following reasons:
  - They are resources or tasks detailed enough in the departing MS model.
  - Their refinement would not add strategic value to the selection process.
- *Refined elements* are those not present in the MS model but refining one or more of this model, which are called *refinable elements*. Refined elements express some MS concept in a more concrete way. Thus, some situations are not allowed, more precisely a task or resource dependency cannot be refined into a goal dependency. Non-kept SP actors that are defined as specializations also fall into this category, since in our context specialization may be conceptually considered as refinement.
- *New elements* are those not present in the MS model and not refining any of its elements. Usually they express advanced capabilities of a particular SP that are not standard in the corresponding MS.

As an example, we derive a SP model for a particular kind of AVSP, the McAfee VirusScan v. 4.5.1 [18], see fig. 7. Kept and refined dependencies identify which elements of the corresponding MS model for AVSP (cf. fig. 6) they correspond to. We introduce two new software actors inheriting from the “Other Software Package” actor, because the VirusScan supports connection with mail systems and web browsers. In both cases, communication is implemented via some files placed at some designated directories; this yields two refined dependencies. On the contrary, the

AVSP actor: in the case of the mail system, refines the concept of "File to be Scanned" into "Attached File"; in the case of web browser, requires a specification of the addresses that must be avoided in visits, which are new dependencies generated by the new actor. The other actors are kept from the MS model.

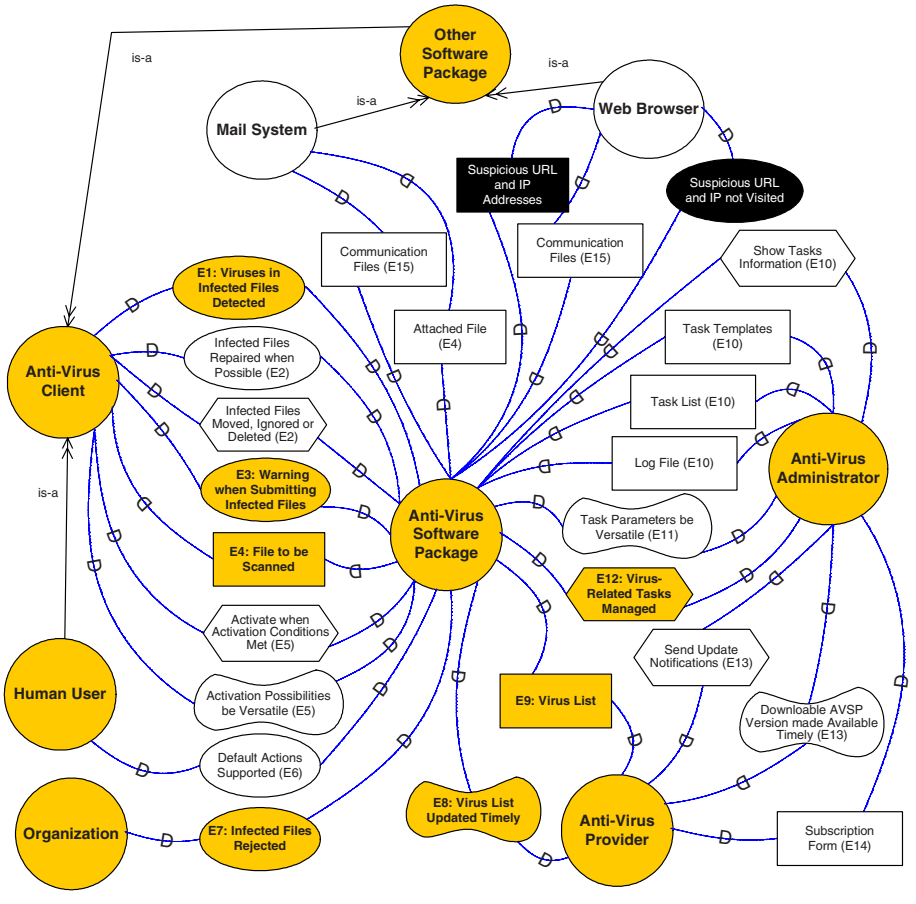

Fig. 7. An SD model for the VirusScan anti-virus SP (grey: kept elements; white: refined elements; black: new elements; enclosed in parenthesis, references to the MS model).

Fig. 8 models the concept of derivation as an association among MS and SP models. At its turn, the derivation concept can be defined as a set of links among elements from the MS and SP models[1]. To do so, elements of the involved models are bound to the association class and then put together with the *DerivationLink*

[1] We could have used a ternary association, but splitting into two binaries makes the integrity constraints easier to write, specially in their OCL form.

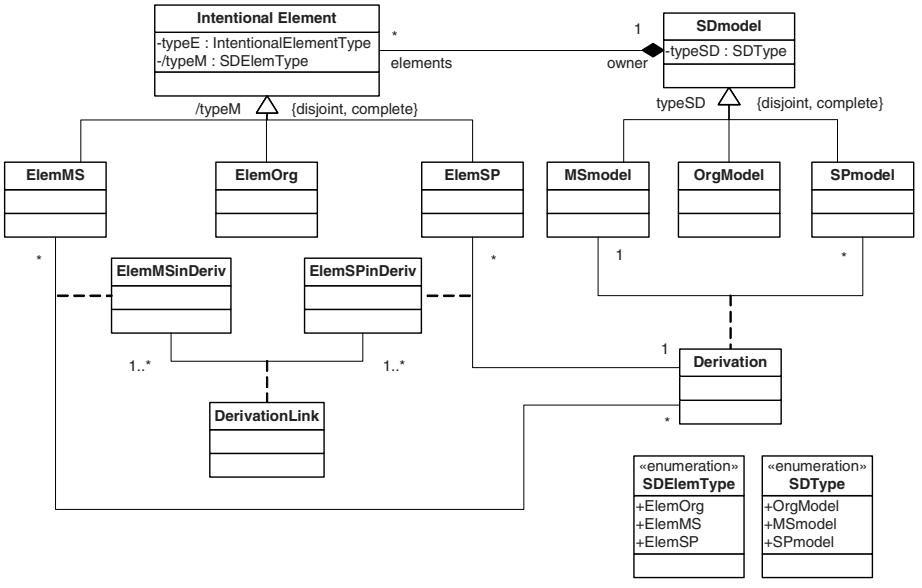

Fig. 8. Derivation from MS to SP models

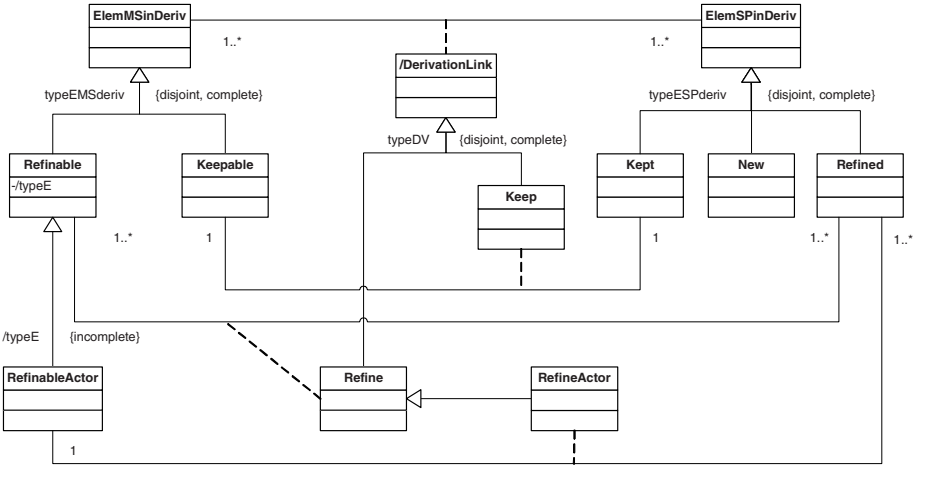

Fig. 9. Derivation from MS to SP models (detailed view).

association class. Five integrity constraints are required to state that the derivations shall preserve the type of their intentional element (IC4), the type of their actors (IC5), their specialization hierarchies (IC6) and also the actors involved in their dependencies (taking actor specialization into account; IC7), and shall belong to the same model (IC8). We need an additional constraint to state that derivations shall be complete (i.e., each element from the MS model shall be derived into the SP model;

IC9). Then, at fig. 9 we further distinguish among kept, new and refined elements which allows to define more detail in the model, specially concerning multiplicities and particularly in the case of actors. We introduce also two more integrity constraints to state that kept dependencies preserve their type (C10) whilst refined ones do not allow converting tasks or resources into goals or soft goals (C11).

7 Matching MS and SP Models with Organizational Models

MS and SP models can be compared with organizational models. In the first case, we address the other fundamental question mentioned in the introduction, namely the variety of MS. In the second case, we can use this comparison to carry out a first screening of candidates and eliminate those that clearly do not apply to the case under consideration. In both cases, furthermore, new goals can be generated and incorporated into the organizational model. The comparison takes the form of a matching process among the actors and dependencies of the involved models.

The matching can be classified according to:

- Coverage of the organizational model: a matching is *complete* if every intentional element of the organizational model is matched with one or more intentional elements in the MS or SP model, otherwise it is *incomplete*.
- Coverage of the MS model: a matching is *surjective* if every dependency of the MS model that is linked to at least one actor that is matched with some actor of the organizational model, is also matched with one or more dependencies in the organizational model, otherwise it is *overloaded*. In other words, the definition of coverage cannot take into account dependencies that link actors that have arisen in the MS model.

Note that every combination of the two pairs of classes is valid. For instance, in our example, the matching with the MS model is complete and surjective, as shown in table 1; therefore, we may conclude that the MS of anti-virus packages is useful for the problem at hand. In the figure, please note that the software and administrator actors (AVSP and AVA) are not matched with any organizational actor; therefore, according to the explanation above (coverage of the MS model), the dependencies among them are not considered. Of course, if the matching with the MS is complete, so will be the matching with any of the SP, since we force SP models to be a derivation from MS ones.

Table 1. Matching among organizational and MS models (left, actors; right: dependencies)

Org	MS
U	AVC
	HU
	OSP
O	O
DIE	AVP

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E13	E14	E15
D1	x	x										
D2	x	x										
D3			x									
D4							x					
D5					x	x						x
D6								x	x	x		
D7				x								
D8											x	

Unlike this example, model matching will usually be incomplete and overloaded. In the case of MS, if the degree of incompleteness is too high, the MS will not apply to the problem. Otherwise, one or more of the following statements will hold: 1) the information-system-to-be will combine SP coming from different MS; 2) some glue code is necessary to achieve the expected functionalities; 3) uncovered organization requirements must be prioritised to decide if they can be relaxed; 4) the MS offer some functionalities or behavioural characteristics that were not foreseen by the initial requirements of the system and that can be incorporated. Even a complete matching does not exclude that other MS are also selected as applicable. In our case, in addition to AVSP, we could thought of data encryption and spyware tools MS as applicable.

The UML class diagram for specifying the matching concept is presented in fig. 10. There are some similarities with the specification of derivation presented in the previous section, both for the class diagram itself and the integrity constraints[2]: we remark that IC4 hold and also IC6, IC7 and IC8 but just partially: dependencies in the organizational model may be matched with dependencies in the MS or SP models among the same two actors up to specialization (and then IC6 to IC8 apply) or among one of the organizational actors and one the new actors (the system, the administrator, …). There are not further restrictions about the type of actors or dependencies. On the other hand, both types of matching are closely related: we need an integrity constraint (IC11) stating that if an intentional element *A* belonging to an organizational model matches with an intentional element *B* belonging to a MS model and with an intentional element *C* belonging to a SP model, then *C* shall be derived from *B* (either kept or refined). We define as derived attributes the coverage of matching.

As already mentioned, the matching can be used to point out new organizational needs for the organization. In particular we may say the following:

- When matching with a MS model, it holds that: each dependency *D* in the MS model that links a new actor (one that does not match with a organizational actor) and a matched actor (one that matches with a organizational actor) is indicating some dependency that is not identified in the departing organizational model.
- When matching with a SP model, it holds that: for each dependency *D* in the SP model that links a new actor and a matched actor:
  - If *D* is derived or kept from a dependency *E* that appears in the MS model from which the SP model derives, then analyse the matching among the MS model and organizational model and behave as in the case before. We act this way because the MS model is more abstract and then closer to the organizational point of view.
  - Otherwise, the dependency *D* itself is considered.

On the other hand, actors in the MS and SP models that inherit from other actors that are matched to organizational one, are also worth to be considered for inclusion in the organizational model. This is the case of the mail system and web browser actors from the McAfee VirusScan model.

[2] In [12] we have defined some metaclasses that generalise the derivation and matching concepts and we define some of the associations and integrity constraints in the superclasses.

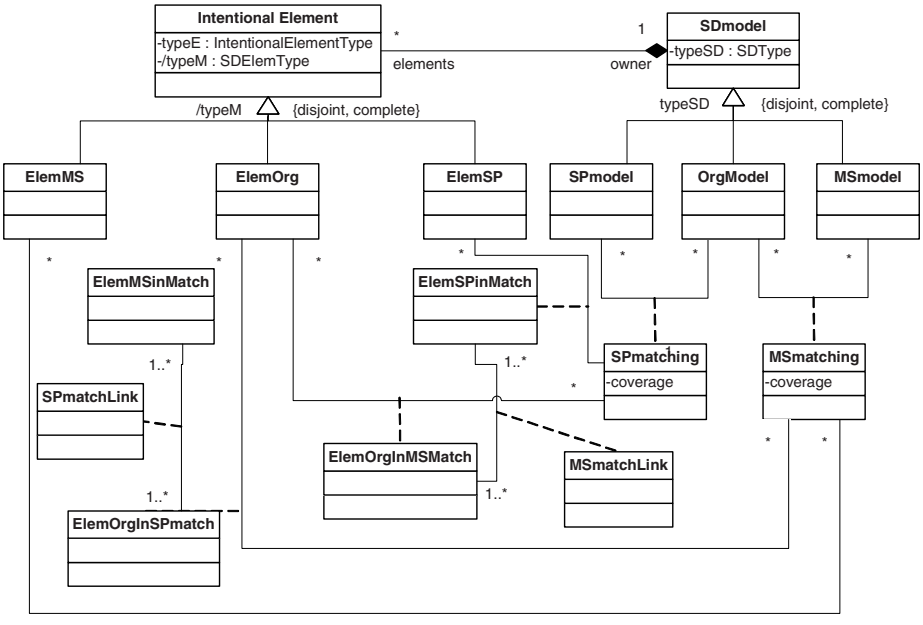

Fig. 10. Matching among organizational, MS and SP models.

Please note that we are not proposing automatic updating of the organization model, since this requires a careful strategic analysis. The contribution here is that we provide a systematic way to identify candidate modifications to be performed.

8 Conclusions

We have presented an approach for using a widespread specification technique such as goal-oriented modelling in the increasingly important context of software package selection. Our proposal is based on the notions of matching and derivation which bind intentional elements belonging to different goal-oriented models. We classify our approach as lightweight because these 2 kinds of correspondences have been defined without any type of model-checking techniques to validate that the elements involved satisfy some formula. We have provided a rigorous meaning to the proposal by means of a UML conceptual model. We believe that our proposal has a positive impact to both the context of SP selection and goal-oriented modelling. For SP selection:

- It provides a starting point for this activity, focusing on goals before more detailed issues such as measurable requirements.
- It acts as a high-level documentation for recording the most strategic decisions and services arisen in the selection process.
- It supports the classical SP-based software systems life cycle in which SP selection and requirements elicitation are two complementary activities.

- It supports transfer of knowledge (and therefore return on investment) from one selection process to other of the same MS.
- It fits well with the extreme volatility of the market, because of the traceability implied by the derivation and matching notions.

Concerning goal-oriented modelling, existing approaches such as TROPOS [9] are primary concerned with the use of goals for guiding software development. We think that our approach can be linked to TROPOS to obtain a slightly different methodology aimed at SP-based software development, in which the requirements elicitation phase may be defined using the concepts presented in this paper and system implementation may be seen mainly as a SP integration activity. Addressing to these issues is part of our future work.

As a possible criticism, it could be argued that building this type of models may be time-consuming and requires a medium-high level of expertise. This is why in section 2 we have stated that our method is not intended to be universal, it requires some conditions to be applicable: 1) the MS addressed is of general interest: 2) there are a lot of SP offered in this MS; 3) SP are big; 4) the selection activity is monitored by a team that accumulates experience from past selection processes; 5) the requirements are not absolutely known in advance. Although it seems to be a lot of restrictions, a great deal of selection processes in the information systems development satisfy them altogether (communication infrastructure, packages of various kinds: CRM, ERP, document management, …); furthermore, the economical impact of the selection process in such cases is very high. We think that departments, institutes and teams continuously involved in selection activities, namely consultant companies (e.g., Gartner, Forrester, etc.), IT divisions in large organizations (including public ones) and groups of expert with academic profile, either large or smaller (e.g., our own team) are therefore the main targets of our proposal.

One could aim at formalizing in the future the decision of applicability of a MS or SP to a problem and convert our lightweight approach into heavyweight, but we must be aware that this requires lot of work to be done in the selection. First, the non-matching elements of the organizational model should be weighted somehow (as done for instance in [19] using AHP with all the involved stakeholders). Second, one should decide the threshold in which one MS become non-applicable. Third, some kind of decomposition of the goals is surely needed to further explore organizational needs and SP services. A lightweight approach is cost-effective and useful enough to have a first organizational analysis of the elements involved in selection; the utilization of a heavyweight one should be studied carefully.

For related work, we specially mention the notion of goal matching presented by Rolland [20] and the CARE approach by Chung and Cooper [21]. Both proposals are more comprehensive than ours in the sense that they propose to use goal models for driving the whole selection process, focusing on goal decomposition and creating goal graphs; whilst in our proposal the goal-oriented framework is to be used for a first approximation of the problem, with the goal of clarifying the high-level organizational model and pruning the solution space. In this sense, we may say that the late treatments proposed by Rolland and Chung are not contradictory but complementary to our proposal; in other words, they could be part of the heavyweight approach mentioned in the previous paragraph. For the rest of their proposals, the following fundamental differences arise:

- They compare directly organizational needs with product functionalities; no notion of MS model is proposed. This is a significant difference, in connection with the 2 fundamental questions stated in the introduction (proliferation and variety). We have already given arguments supporting the existence of this intermediate model.
- It is not clear how they tackle the usual situation in which a single SP does not solve the problem at hand. They mention the use of glue code and the further customization of the selected SP, but our experiences show that usually the information system must combine several SP. Our proposal deals with this situation in a natural way.
- Our formalization using UML is a good starting point for developing tool support in the future.

Acknowledgements

This work has been partially supported by the CICYT programme, project TIN2004-07461-C02-01.

References

1. J. Kontyo. "A Case Study in Applying a Systematic Method for COTS Selection". In *Proceedings 18th ICSE*, 1996.
2. N. Maiden, C. Ncube. "Acquiring Requirements for COTS Selection". *IEEE Software* 15(2), 1998.
3. C. Alves, F. Alencar, J. Castro. "Requirements Engineering for COTS Selection". In *Proceedings 3rd WER*, 2000.
4. S. Comella-Dorda, J. C. Dean, E. Morris, P. Oberndorf. "A Process for COTS Software Product Evaluation". In *Proceedings 1st ICCBSS*, LNCS 2255, 2002.
5. A. van Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". In *Proceedings 5th ISRE*, 2001.
6. A. Dardenne, A. van Lamsweerde and S. Fickas. "Goal-Directed Requirements Acquisition". *Science of Computer Programming* Vol. 20, North Holland, 1993.
7. E. Yu, *Modelling Strategic Relationships for Process Reengineering*, PhD. thesis, University of Toronto, 1995.
8. GRL web page. http://www.cs.toronto.edu/km/GRL/. Last accessed November 2004.
9. A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, P. Traverso. "Specifying and analizing early requirements in Tropos". *Requirements Engineering Journal*, 9(2), 2004.
10. E. Yu. "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". In *Proceedings 3rd ISRE*, 1997.
11. Object Management Group. *UML 2.0*. http://www.omg.org/. Last accessed Nov. 2004.
12. X. Franch. "On the Lightweight Use of Goal-Oriented Models for Software Package Selection". Technical Report LSI-04-58-R, LSI-UPC, November 2004.
13. X. Burgués, C. Estay, X. Franch, J.A. Pastor, C. Quer. "Combined Selection of COTS Components". In *Proceedings 1st ICCBSS*, LNCS 2255, 2002.
14. J.P. Carvallo, X. Franch, C. Quer. "A Framework for Selecting Workflow Tools in the Context of Composite Information Systems". In *Procs. 15th DEXA*, LNCS 3180, 2004.

15. J.P. Carvallo, X. Franch, C. Quer. “Towards the Selection of a Requirements Management Tool”. Book chapter in *Requirements Engineering for Sociotechnical Systems*, J.L. Maté, A. Silva (ed.), IDEA Group, 2005.
16. E. Yu, J. Mylopoulos. “Understanding "Why" in Software Process Modelling, Analysis, and Design”. In *Proceedings 16th ICSE*, 1994.
17. A.I. Anton. “Goal-Based Requirement Analysis”. In *Proceedings 2nd ICRE*, 1996.
18. McAfee web page. http://www.mcafee.com. Last accessed November 2004.
19. H. Kaiya, H. Horai, M. Saeki. “AGORA: Attributed Goal-Oriented Requirements Analysis Method”. In *Proceedings 10th RE*, 2002.
20. C. Rolland. “Requirements Engineering for COTS Based Systems”. *Information and Software Technology*, 41, Elsevier, 1999.
21. L. Chung, K. Cooper. “Defining Goals in a COTS-Aware Requirements Engineering Approach”. *System Engineering Journal*, 7(1), 2004.