ITC irst

CENTRO PER LA RICERCA
SCIENTIFICA E TECNOLOGICA

38050 Povo (Trento), Italy
Tel.: +39 0461 314312
Fax: +39 0461 302040
e−mail: prdoc@itc.it − url: http://www.itc.it

Domain ontology analysis in agent−oriented requirements engineering

Donzelli P., Bresciani P.

September 2003

Technical Report # P03−12−42

An agent-based requirements engineering framework for complex socio-technical systems

Paolo Bresciani
ITC-irst
(Istituto per la Ricerca Scientifica e Tecnologica)
via Sommarive 14, 38050 Trento-Povo (Italy)
+39 0461 314342
brescian@itc.it

Paolo Donzelli
Department of Innovation and Technology
Italian Cabinet Office
Via Barberini 38, 00187 Roma
+39 329 4308228
p.donzelli@governo.it

ABSTRACT
Adopting Requirements Engineering (RE) techniques based on the fundamental notions of the agent-oriented programming paradigm, i.e. *Agent*, *Goal*, and *Intentional Dependency,* has been recognized as a crucial step towards a more homogeneous and natural software engineering process for complex socio-technical systems, among which Multi Agent Systems. The availability of simple representational tools is a key factor to guarantee stakeholders' active involvement during RE, and therefore the success of such techniques. The paper introduces an agent-based Requirements Engineering Framework (REF) devised to deal with socio-technical systems and support stakeholders participation. REF is designed around the adoption of a simple, but effective, graphical notation. Nevertheless, a limited expressiveness of the graphical language may constraint the analysis process, reducing its flexibility and efficiency. This trade-off is carefully analysed, and some extensions are proposed, which do not affect REF clarity and intuitiveness, while enhancing REF capability to support requirements engineers in planning and implementing their analysis strategies.

Keywords
Agent-based software engineering – User and agent modeling - Requirements engineering – Goals dependency analysis – socio-technical systems

1. INTRODUCTION
Agent-oriented programming paradigms and Multi Agent Systems (MAS) offer the great advantage of adopting the same concept of *Agent* (together with all the other related mentalistic notions of *Goal*, *Intentional Dependency*, *Resource*, and *Task*), commonly used also for describing the social setting in which the system has to operate. Thus, the possibility of using such notions as primitive elements for describing the system requirements during the Requirement Engineering (RE) process is becoming more and more relevant, especially for MAS application solutions. Agent- and goal-based RE approaches represent a first step to fill the gap between RE and Software Engineering (SE), especially, although not only [22], when MAS are concerned [4]. Many SE techniques have been developed for MAS [19,23], but few of them try to analyze and take into account the impact of the final system over the encompassing organization since the early phases of the RE process. In such a perspective, the paper introduces an agent-based RE Framework (REF), previously applied to an extensive project for the definition of the requirements of a simulation environment [13], explicitly devised to tackle such complex issues. REF is strongly based on diagrammatic notations, which immediately convey the dependencies among different actors (or agents)[1] and allow for a detailed analysis of the goals, upon which the actors depend, through a goal decomposition process. REF actors may be social (individuals and organizations, e.g. enterprises, departments, offices) as well as artificial (physical, as an hardware system, and software, as a *software agent* or a society of software agents inside a *software system*). The adopted graphical notation is widely inspired by the i* framework [25] for RE [26] and business analysis and re-engineering [24]. Aside the denotational inspiration to i*, REF introduces also a clear methodology to drive the process of requirement discovery, definition, refinement and reconciliation [13]. Another important aspect of REF is to adopt the i* notational ingredients at a basic and essential level [13]: this choice, although apparently constraining, results to be quite successful in practical terms. Several case studies [10,11,13] demonstrate, in fact, that the simplified notation facilitates the acceptance of the diagrammatic language by the stakeholders, and contributes to a quicker introduction of the methodology in the RE process. Nevertheless, such simplifications may still be consider as possible limits to the REF expressive power and, consequently, to the intentional analyses that could otherwise be carried out.

In the present paper, after a brief introduction to REF (Sections 2), an extensive and concrete case study is adopted to critically revise the analysis process underlying the current methodology, and propose two notational and methodological extensions as possible solutions (Section 3). A tradeoff between the simplicity, and thus the usability of REF (as it is), and its possibly increased expressiveness arises. The conclusions (Section 4) suggest that, when appropriately used, the proposed extensions do not jeopardize REF usability and its acceptance by the stakeholders, but support the analysts in planning and performing a more efficient analysis strategy.

2. THE CURRENT APPROACH
REF is designed to deal with, and reason about, socio-technical systems, i.e., where the software system and its application context form a larger *socio-technical system* that has to be treated and analyzed as a whole: the overall needs of such a system are the ones that have to be fulfilled [15,19,23]. Complexity of socio-technical systems goes beyond the complexity of the software

[1] A distinction between agent and actor may be introduced, the first indicating a specific instance of the second, as, for example, in the case of a specific *software agent*. This distinction is not crucial in the context of this paper, and both the terms will used almost interchangeably.

system itself, it encompasses the complexity generated by the impact of the system upon the organizational structure, from the business process, to the behavior of the single employee.

The basic idea is that REF has to provide the analyst with a powerful tool to capture high-level organizational needs and to transform them into system requirements, while changing and/or redesigning the application context in order to better exploit the new system. The framework tackles the modeling effort by breaking the activity down into more intellectually manageable components, and by adopting a combination of different approaches, on the basis of a common conceptual notation.

Agents are used to model the organization [9,13,26]. The organizational context is modeled as a network of interacting agents (any kind of active entity, e.g. teams, humans and machines, one of which is the target system), collaborating or conflicting in order to achieve both individual and organizational goals. *Goals* [1,2,8,13,26] are used to model agents' relationships, and, eventually, to link organizational needs to system requirements. According to the nature of a goal, a distinction is made between *hard goals* and *soft goals*. A goal is classified as hard when its achievement criterion is sharply defined. For example the goal *document be available* is a hard goal, being easy to check whether or not it has been achieved (i.e., is the document available, or not?). For a soft goal, instead, it is up to the goal originator, or to an agreement between the involved agents, to decide when the goal is considered to have been achieved. For example, the goal *document easily and promptly available* is a soft goal, given that as soon as we introduce concepts such as "easy" and "prompt", different persons usually have different opinions. Another characteristics of soft goals is that they can often be seen as a kind of modifiers or *quality attributes* associated to an hard goal; thus, in the previous example, the soft notion of easily and promptly *modifies* the precise objectives of having the document available.

Distinguishing goal modeling from organizational modeling, and then, further distinguishing between hard goal modeling and soft goal modeling, is a key aspect of REF, and helps to reduce the complexity of the modeling effort. The proposed framework, therefore, supports three inter-related modeling efforts: the organizational modeling, the hard goal modeling and the soft goal modeling. During *Organization Modeling*, the organizational context is analyzed and the agents and their hard and soft goals identified. Any agent may generate its own goals, may operate to achieve goals on the behalf of some other agents, may decide to collaborate with or delegate to other agents for a specific goal, and might clash on some other ones. The resulting goals will then be refined, through interaction with the involved agents, by hard and soft goal modeling. The *Hard Goal Modeling* seeks to determine how the agent can achieve a hard goal placed upon it, by decomposing them into more elementary subordinate hard goals and tasks (where a task is a well-specified prescriptive activity). The *Soft Goal Modeling* aims at producing the operational definitions of the soft goals that emerged during the organizational modeling, sufficient to capture and make explicit the semantics that are usually assigned implicitly by the involved agents [12,13,14] and to highlight the system quality issues from the start. A soft goal is refined in terms of subordinate soft goals, hard goals, tasks and constraints. Soft goals refinement has to be reiterated until only hard goals, tasks and constraints are obtained (that is, until all the "soft" aspects are dealt with). Constraints are associated with hard goals and tasks to specify the corresponding quality attributes. So, for example, the soft goal *to make a document easily and promptly available*, beside spawning the hard goal *to make a document available*, will lead also to a set of constraints (e.g., types of access channels, number of hours after which a document is available, etc.) specifying the concepts of easy and prompt. In other words, for each soft goal, the resulting set of constraints represents the final and operationalised views of the involved quality attributes, i.e. the quality models that formalize the attributes for the specific context [3,6]. As it will be discussed in section 2.2, goal fuzziness has been reduced by repetitively asking the agents what they needed to know, to perform, have delivered or have performed in order to consider the soft goal as achieved. This approach leads to model a soft goal model with a set of hard goals, tasks and constraints, that is with a full operationalization of all the *softness* initially hidden in the goal.

2.1 The case study: ERMS

In order to introduce REF, pawing at the same time the way for discussing its limits and possible extensions (Section 3), we refer to an on-going project aiming at introducing an Electronic Record Management System (ERMS) into a complex administrative organization. The impact of such a system on the common practices of the communities and the sub-communities of knowledge workers who will adopt is quite relevant. Indeed, ERMS is at the moment used by more than 300 employees and handles a flow of about 200.000 documents/year, but it is expected to reach about 2000 users and 2 million documents/year.

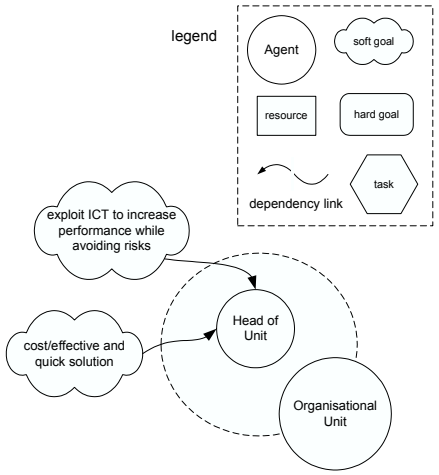

Figure 1 - Introducing the ERMS: the initial model

A ERMS is a complex Information and Communication Technology (ICT) system which allows efficient storage and retrieval of document-based unstructured information, by combining classical filing strategies (e.g. classification of documents on a multi-level directory, cross-reference between documents, etc.) with modern information retrieval techniques. Moreover, it usually provides mechanisms for facilitating routing and notification of information/document among the users, and supporting interoperability with similar (typically remote) systems, through e-mail and XML [18]. A ERMS represents the basic element for a *knowledge workplace*, i.e. a working environment where a knowledge worker can easily access and gather information, produce knowledge and deliver results through a multitude of channels (from personal computers, to laptops, PDAs, mobile phones, etc.). It is, in fact, a necessary step for introducing more sophisticated document management tools, such as workflow technology and digital signature, both fundamental mechanisms for a paperless and ubiquitous working environment. Several factors (international benchmarking studies, citizens demand, shrink budgets, etc.) called for the decision of leveraging new technologies to transform the

organization's bureaucratic structure into a more creative, and knowledgeable environment. The initial organization model expressing such a situation is shown in Figure 1. Circles represent Agents, and dotted lines are used to bound the internal structure of complex agents; that is, agents containing other agents. In Figure 1, the complex agent *Organization Unit* corresponds to the organizational fragment into which it is planned to introduce the new ERMS, whereas the *Head of Unit* is the agent, acting within the *Organizational Unit,* responsible for achieving the required organizational improvement (modeled by the soft goals *exploit ICT to increase performance while avoiding risks*, and *cost/effective and quick solution*). Goals, tasks and agents (see also next Figures) are connected by dependency-links, represented by arrowhead lines. An agent is linked to a goal when it needs or wants that goal to be achieved; a goal is linked to an agent when it depends on that agent to be achieved. Similarly, an agent is linked to a task when it wants the task to be performed; a task is linked to an agent when the agent is committed at performing the task. Again, an agent is linked to a resource when it needs that resource; a resource is linked to an agent when the agent has to provide it. By combining dependency-links, we can establish dependencies among (i.e. two or more) agents. As mentioned, the soft goal modeling allows analysts and stakeholders to *operationalise* the *soft* aspects implicitly included in the meaning of the soft goal (i.e., to transform them into "implementable" properties). For example, Figure 2 describes how the soft goal *exploit ICT to increase performance while avoiding risks* is iteratively top-down decomposed to produce a set of tasks, hard goals, and constraints that precisely defines the meaning of the soft goal, i.e. the way to achieve it.

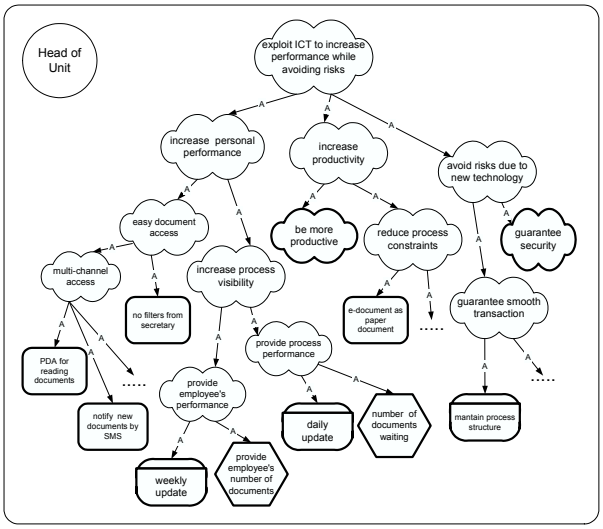

Figure 2 - The "*exploit ICT …..*" Soft Goal Model

Figure 2, in other terms, represents the strategy that the *Head of Unit* (as result of a personal choice or of a negotiation with the upper organizational level) will apply to achieve the assigned goal. Again, the arrowhead lines indicate dependency links. A soft goal depends on a sub-ordinate soft goal, hard goal, task or constraint, when it requires that goal, task or constraint to be achieved, performed, or implemented in order to be achieved itself. These dependency links may be seen as a kind of top-down decomposition of the soft goal. Soft goals decompositions may be conjunctive (all the sub-components must be satisfied, to satisfy the original soft goal), indicated by the label "A" on the dependency link, or disjunctive (it is sufficient that only one of the components is satisfied), indicated by the label "O" on the dependency link (see Figure 4).

According to Figure 2, the *Head of Unit* has to *increase personal performance*, to *increase productivity* of the whole unit, and also to *avoid risks due to new technology*. Let's consider in details only the first sub - soft goal, i.e. *increase personal performance*. It spawns two subordinate soft goals, *easy document access*, for which the *Head of Unit* will require a *multi-channel access* system in order to be able to check and transfer the documents to the employees also when away from the office (asking at the same time that the secretary does not have to operate as a filter for documents), and *increase process visibility*, to take better informed decisions. In particular, the soft goal *increase process visibility* will eventually lead to the identification of some tasks (functionalities) the system will have to implement in order to collect and make available some data about the process (e.g. number of documents waiting, number of documents under elaboration) and about the employees (number of assigned documents), and of some associated constraints, represented by a rounded-rectangle with a horizontal line, characterizing such data. In Figure 2, for example, they specify the frequency of update: daily for the process data and weekly for the employee's ones.

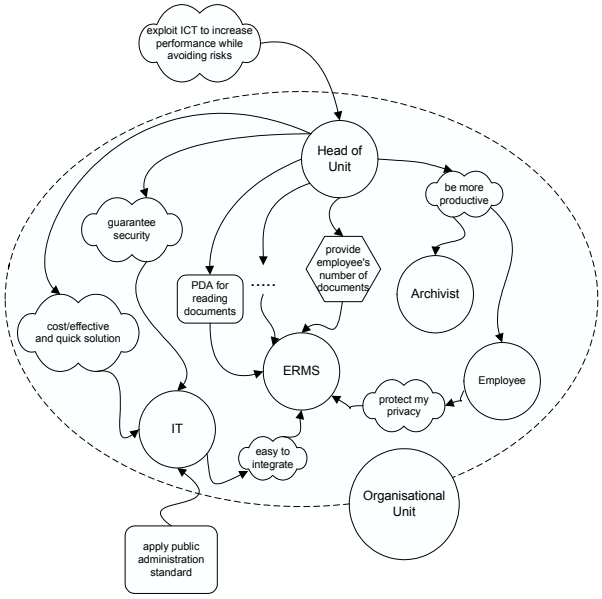

Figure 3 - The evolving organization model

In Figure 2, the *items in bold outline* are those that the *Head of Unit* will pass out, having decided to depend on other agents for their achievement. For such a reason, they are not further analyzed; instead they will be refined as further agreement between the *Head of Unit* and the agent that will be appointed of their achievements. The results of this analysis allow us to enrich the initial organization model in Figure 1, leading to the model in Figure 3, where some details have been omitted for the sake of clarity. In Figure 3 some new agents have been introduced: the *Archivist* and the *Employee*, which have to be more productive, the *IT (i.e.,* the *Information Technology* Unit*)*, which has to guarantee security and the *ERMS,* upon which the identified goals, tasks and constraints will be placed. From Figure 3, we can also see that the *Head of Unit* has decided to delegate the soft goal *cost/effective and quick solution* to the *IT* agent, which, on its turn, will have to achieve other goals coming from the external environment, such as, for example, *apply public administration standards*. At this point, the analysis can proceed, for example, by focusing on how the *Employee* will try to achieve the soft goal be *more productive*. To *be more productive*, the *Employee* will define its own strategy, eventually reaching an

agreement with the *Head of Unit*. Such a strategy is shown by the soft goal model in Figure 4, where we can see how in order to *be more productive* the *Employee* will ask that the system will be *easy to learn* and will *make collaboration easier* with the other employees, which are dealing with the same document. For sake of brevity, only a partial refinement of these soft goals is shown.

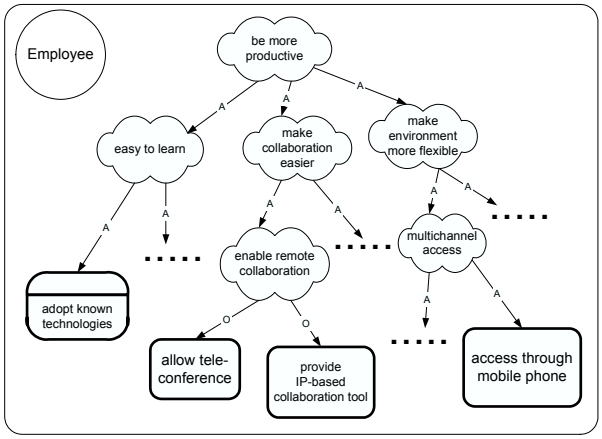

Figure 4- The "be more productive" Soft Goal Model

The soft goal *easy to learn* will spawn, among other items here omitted, the constraint *adopt known technologies* (i.e. technologies which the employee is already used to), whereas the soft goal *make collaboration easier* will lead, through further refinement, to a series of hard goals implying specific capabilities (e.g. either a teleconference or an IP-based collaboration tool) and access channels (e.g. mobile phone, laptop, etc.).

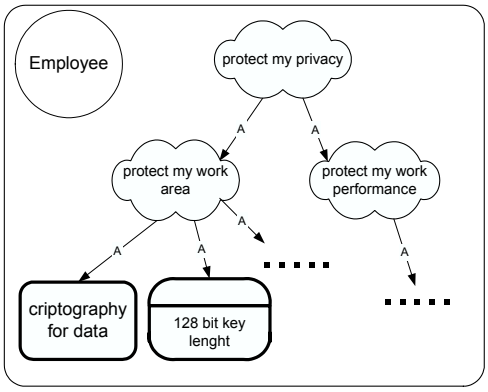

Figure 5- The "protect my privacy" Soft Goal Model

As already seen for the *IT* agent, also the *Employee* may have his/her own needs, leading to new goals. For example, the *Employee* may be concerned of possible side effects on the privacy related to the introduction of the *ERMS*, represented by the soft goal *protect my privacy* upon the *ERMS*, introduced in Figure 3. By modeling the soft goal *protect my privacy* (Figure 5), we can discover that the *Employee* wants to be sure that its own private area of work will not be accessible by others, asking for example to be enabled to adopt a cryptography tool (whit a 128 bit length security key) to protect particular data, but above all, given that all the activities will be performed through the *ERMS*, that s/he does not want someone to be able to monitor his/her actions. This clearly conflicts with the possibility of monitoring the employee's performance that was required by the *Head of Unit* (Figure 2), requiring some trade-off to be identified.

2.2 REF background

REF combines advanced *requirements engineering techniques* (i.e. agents [9,13,26] and goals [1,2,8,13,20,26]) with *software quality modeling approaches* [3,6,17], to capture the stakeholders' perception of quality since the beginning of a new project, and to produce agreeable-upon and implementable functionalities and constraints.

Ideas from RE Techniques. REF is strongly based upon i*, the modeling framework suggested by Eric Yu [25,26]. However, it introduces some simplifications and tends to adopt a more pragmatic approach in order to obtain a greater and more active involvement of the stakeholders during the requirements discovery, elicitation and formalization process. The aim is to let the stakeholders to easily grasp the notation since the beginning of a project. Thus, the adopted simplifications include: 1) the use of only one type of actor/agent (i.e., no distinction is made between *agent*, *role* and *position*); 2) the use of only one kind of link, both as *dependency link* as well as *decomposition* and *means-end* link. This choices allows for a more intuitive reading of the REF diagrams, as well as a more natural flow of dependencies, from the dependencies among actors, in the organization model, to the decomposition dependencies, in the (hard and soft) goal models.

REF adopts a strict top-down approach, during which the analysts, in close cooperation with the stakeholders, drill through the organization, until reaching the desired level of detail, by using three different (but similar) modeling tools: organization models, soft goal models, and hard goal models. In particular, once an initial model of the organization is built, the REF process evolves in a cyclic way through two main phases: a) goal modeling phase, during which the soft and hard goals discovered during organization modeling are refined; and b) organization modeling phase, during which the analysts use the information gained modeling the goals to enrich and extend the initial organizational model: i.e. to replace the goals with their models, and to introduce the new agents identified as relevant to achieve those goals. New agents usually lead to new goals, triggering the goal-modeling phase again. Such a cycle is continued until the desired (and needed) level of details is reached. REF emphasizes the operational role that soft goals can play in deriving the requirements of a new system. Soft goals, in fact, beyond playing an important role in supporting reasoning between alternatives (as in i*), since the early stages of a project, can provide a systematic and organized way of handling non-functional requirements in a very pragmatic way. REF recognizes the need of explicitly "resolving" soft goals, by turning them into more manageable (and implementable) constraints.

Ideas from Quality Modeling Techniques. To refine a soft goal, the analysts have first to specify the actions that are usually implied, and sometimes hidden in the goal (e.g., to make a document available in the soft goal *document easily and promptly available*), then they have to make operational its softness (e.g. in identifying the constraints that describe the concepts of easily and promptly). In refining such a softness, i.e. to make it operational, the analysts perform *a two-step process:* 1) First, they *build a quality model* [6] of the soft (quality) attribute, by identifying the characteristics that the stakeholders assume to be important in judging it. For example, for judging how easily and promptly a document is available, we can identify, as relevant, system characteristics such as the types of access channels, the hours after which the document is available, the possibility of mobile access, and so on. 2) Second, they *populate the quality model and freeze the result*, by specifying for each characteristic the desired values (or range of values),

according to the corresponding measurement scale [14]. In other terms, for each characteristic the analysts specify a constraint. So, for example, for the limit of hours after which the document has to be available (absolute scale), a number or a range of numbers have to be assigned, while, for the types of access channels (nominal scale), the desired values have to be specified.

To perform quality modeling, the analysts may turn to quality models already available in literature [14], such as the McCall, the Bohem, or the IEEE standard, or adopt more sophisticated empirical measurement identification methods, such as the Goal Question Metric (GQM) approach proposed by Basili [3] and its extensions [6]. Quality models show little flexibility and may result difficult to customize to the specific context's and agents' needs. On the contrary, empirical measurement identification methods recognize that quality issues can not live in isolation, but have to be derived from, and linked to, their operational context (i.e. stakeholders, problem domain, underlying reasons, etc.)

The most classical of the empirical methods is the GQM approach, based on the idea that measures have to be identified starting out from the analysis of the goal that has to be achieved. The goal (be precisely stated, by specifying the *object of study*, the *purpose*, the *quality focus*, the *viewpoint* and the *context*) is refined into several questions that break down the issue into its major components, and each question is refined into metrics. Basically, measures are obtained by applying a question-answer mechanism: ask which "Questions" we should be able to answer in order to achieve the "Goal", and then ask what "Metrics" we should collect to be able to answer those questions. In such a perspective, a GQM-like approach has been applied for goal refinement during soft goal modeling. For each soft goal, in fact, REF clearly identifies the target object (object of study), the reasons (purpose), the soft attribute (quality focus), the stakeholders (viewpoint), and the application context (context), providing the basis upon which *a multi-step GQM-like question-answering mechanism* can be adopted as technique to support *elicitation*. In other words, goal fuzziness has been reduced by repetitively asking the agents what they needed to know, to perform, have delivered or have performed in order to consider the soft goal as achieved; provided answers have been used to build the soft goal models. This approach allows to eventually close each soft goal model with a set of hard goals, tasks and constraints, that is with a full operationalization of all the *softness* initially hidden in the goal.

3. LIMITS AND EXTENSIONS

As described in Section 3, REF aims at providing a representational framework for requirements definition and analysis characterized by a sufficiently expressive graphical notation that, at the same time, be simple enough to be easily and quickly grasped by the stakeholders, even by those unfamiliar with requirement engineering techniques. Indeed, these are very important factors that make REF applicable to real projects, and ensure a concrete involvement of the stakeholders, allowing for a quicker and more effective process of knowledge acquisition and requirements elicitation, that is, a process leading to domain descriptions much closer to the real state of affairs [1,13].

REF has shown its simplicity and effectiveness in different projects [10,11,12], allowing a concrete involvement of the stakeholders. Such properties we believe are mainly based on two key points: 1) the use of only one type of link (the dependency link); 2) the focus, during both hard and soft goal analysis, on only one goal (and one actor) at time; this leads to the drawing of very simple goal analysis diagrams, strictly generated by a top-down process. Thus, only the generation of trees is considered: in this way the resulting diagram is independent from the order in which it is developed (breath first instead of depth-first, o mixed). These two characteristics allow for a very easy readings of the goal models; the second feature, in particular, allow the stakeholders (and the analysts as well) to concentrate the attention on one problem at time, and not being worried about the order in which different analyses of different sub-trees should be interleaved in order to obtain different possible diagrams: the tree is always generate in the same shape, whatever node expansion sequence is followed. We now aim here at analyzing whether or not these two very important REF characteristics may represent a limit to some relevant aspects of the process of domain description. In particular, in this paper, we want to tackle two particular possible cases in which the present version of REF tends to show some limits. In the following we describe these two type of situations, also by means of few examples related to our case study ERMS, and suggest that simple extensions of REF are possible, in order to allow for more flexibility in the process of model description and requirements elicitation. In the next and concluding section we will argue that the suggested extensions do not restrict the simplicity of REF in a considerable way, if used appropriately.

3.1 Sharing goals (tasks, constraints…)

In REF only tree goal diagrams are produced, thus, there is no the explicit possibility to deal with sub-goals (or constraints, or tasks, or resources) that may be shared by different upper-level goals. This situation may be further distinguished in at least three different sub-cases.

First case: top-down tree expansion and analysis induces at introducing different sub-goals (or constraints, or tasks, or resources) for any different goal that will be found during the goal analysis activity, even when different goals could be (partly) satisfied by the same sub-goal (or constraint, or task, or resource).

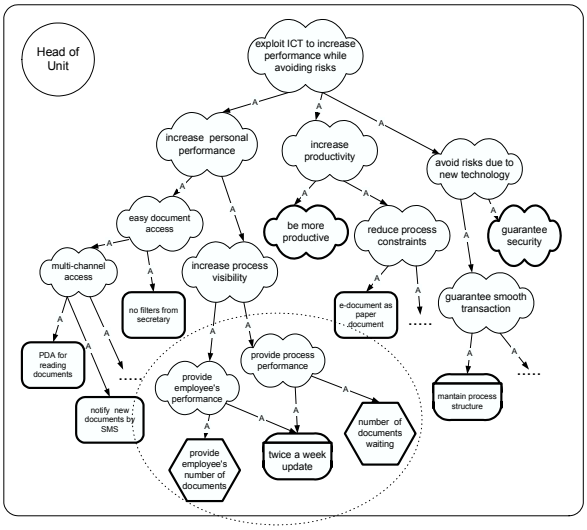

Figure 6 - The "exploit ICT ….." Soft Goal Model revised

For example, in Figure 2, two distinct constraints have been introduced for satisfying the two soft goals *provide process's performance* and *provide employee's performance*, namely the constraints *daily update* and *weekly update*. Instead, for example, the *Head of Unit* could have accepted that the two soft goals, rather than requiring two different specialized constraints (as in Figure 2), would have shared the same constraint, e.g. *a twice a week update* (as in Figure 6). After all, according to REF, any

sequence may have been followed in analysing the two soft goals, and the two constraints may have been introduced in two very different moments, making it very difficult to spot that a common (although slightly different) constraint could have been adopted. This compromise, instead, could have been identified and judged as acceptable if considered by the analyst together with the *Head of Unit* at the proper moment during the design activity. The difference between Figure 2 and Figure 6 is minimal, regarding only leaf nodes, as highlighted by the dotted circle. Thus, Figure 6 can be obtained as a simple transformation of Figure 2. But let us consider a more complex hypothetical case, in which the two nodes *collapsing* in one are non-leaf nodes, with possibly deep trees expanding from them: relevant parts of the two sub trees, rooted in the two nodes, would have to be revised, in order to consider an alternative non-tree-based analysis. Thus, in this case, it would be strategic to be able to introduce a common child for the two different nodes before proceeding with the analysis of the nodes' sub-trees. It is clear that, now, different diagram evolution strategies, and thus development sequences, may lead to quite different results or, even when producing the same result, this may be obtained with different degrees of efficiency (depending on the strategy). For example, a top-down bread-first diagram expansion could be probably preferred to a top-down depth-first strategy. In this way, it may appear appropriate to develop a *shared* sub-tree only once, with two advantages: 1) at the design level, the analysis has not to be carried out twice; 2) at the implementation level, the complexity of the system to be implemented will be reduced, being two potentially different requirements, and all the following details, collapsed in one. Of course, not always collapsing two nodes (and eventually all their sub-trees) in one is a reasonable choice, and may be the result of a compromise that has to be carefully evaluated. It is up to the analyst (together with the stakeholders) to decide if the compromise is acceptable. The important fact here is that, allowing the analysts to show the possible set of shared requirements (easily and visually appraisable) allows also to reason about the amount of savings in design and implementation time, and this is a factor otherwise not evidenced. In other terms, the analysts have a more powerful tool to evaluate design alternatives, also with respect to the possible consequence in terms of system detailed design and development costs. Thus, considerable advantages seems to be introduced by the possibility of considering also directed acyclic graphs, instead of only trees, as design artefacts. Indeed, a trade-off is present, and has to be carefully considered: the *strict* REF approach is much more simple, and make the tree expansion order irrelevant, that is a feature very favourable to usability and comprehension of the process; to find, instead, possible common sub-goals, continuous (or at least very frequent) comparison of sub-goals is necessary, and the expansion sequence may make some difference: this constraint make the process users (that are, the analysts, but specially the stakeholders) more bound to rules and intuition on the *best path*.

Second case: as a specialization of the first one, we can consider the case in which the similar sub-goal *sharing* happens among goals originally attached to an actor as introduced during the organizational model analysis. In this case, the REF methodology would lead the stakeholder to duplicate the sub-goal in two different diagrams, possibly with slightly different labels, although with the same semantics. Catching these cases would be very important in order to avoid duplicated analysis. Moreover, as soon as a sub-goal is recognized to be shared by different goal analysis, immediately it becomes more relevant than other goals: its satisfaction is in fact useful (or better, necessary or sufficient, or both) to two different goals. Thus, the decision could be taken that more design, analysis and implementation efforts should be applied to this "valuable" sub-goal.

Third case: a final, and more intriguing situation may arise when the very same sub-goal can be shared among two different actors, as a consequence of two different and autonomous analyses of two different goals of the two actors, respectively. Consider for example the *increase personal performance* soft goal in Figure 2: its analysis leads to the sub-soft goals *easy document access* and *increase process visibility*. The first one, on its turn, leads to the soft goal *multi-channel access*. The same soft goal *multi-channel access* is present also in Figure 4, as result of the analysis of the soft goal *be more productive* for the employee, that is as a soft goal that the employee considers as relevant in order to become more productive. Again, as in the previous case, the analysis of such a shared soft goal immediately assume a higher relevance and priority over the analysis of other goals. Its satisfaction is desired by two actors! The analysis can therefore be carried out once for both the actors, exploiting the approach to better combine their needs in a synergic way, and avoiding duplications that, although most of the times are due only to slightly different ideas, may generate over complexity in terms of system's functionalities. For example, leading to the selection of only one kind of mobile access channel able to satisfy both the agents.

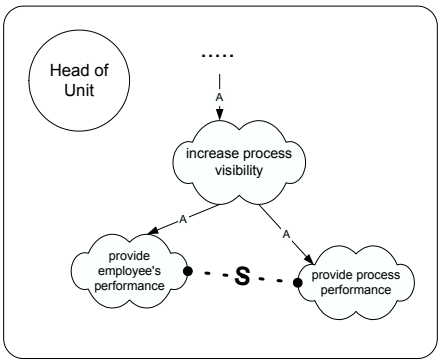

Figure 7 –Sharing between goals of the same agent

From the analysis of all the previous three cases, clearly emerge the need of introducing in REF some kind of mechanism to support the analysts during their analysis, refinement and navigation through the goals and sub-goals models. In particular, we propose to provide the analysts with a specific notation to be used to highlight situations (e.g. such as those described in the previous cases) where they believe that some commonalities could be hidden, i.e. that shared-goals could arise during the analysis. In other terms, to introduce in REF a notation suitable to act as a high-level reasoning support tool to enable the analysts to record their intuitions while building the goals models: i.e. making notes to drive their strategies. For example, to highlight where a top-down bread-first diagram expansion may be preferable to a top-down depth-first strategy. As such a notation, we introduce a dotted labeled -line to connect the two or more goals among which a possible sharing could emerge. The line (the *S-connection*) does not have arrows, being the relationship perfectly symmetric, and it is marked by the label "S", where s stands for "Sharing".

Figure 7 shows a fragment of Figure 2 where the S-connection has been adopted. In particular, it shows how the S connection could have been used during the analysis of the soft goal to highlight in advance the possibility of sharing between the soft goals *provide employee's performance* and *provide process performance* (the first example case previously analyzed). In the same way, in Figure 8 is depicted the use of the S-notation to

highlight, within the soft goal analyzed in Figure 2, a possible sharing between the soft goal *increase personal performance*, that the *Head of Unit* wants to achieve, and the soft goal *be more productive,* that the *Head of Unit* imposes, transfers to, the *Employee* (the third example case previously analyzed).

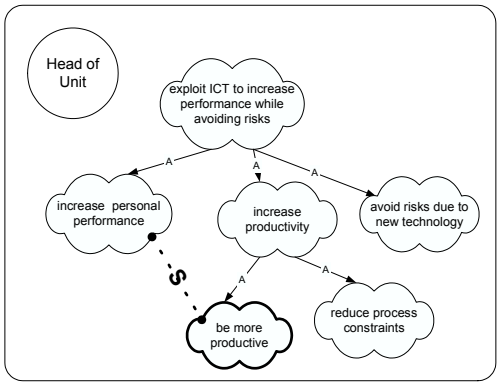

Figure 8 –Sharing between goals of different agents

It is worth noting how the S-notation is only a reasoning support mechanism that tends to disappear once the analysis proceeds. In other terms, the S-notation's purpose is to mark possible sharing situations to drive the analysis (e.g. bread-first, multi-agents analysis, repeated back to back comparisons, and so on), but does not have any reason to exist any more once the goals have been explored: the initial REF notation, with its simplicity, is sufficient for that regard.

3.2 Clashing goals (tasks, constraints…)

Up to here we have been concerned with the possibility of sharing (among different actors, or among higher level goals of one actor) one or more sub-goals (or constraints, tasks, and resources). Another interesting, and actually very common situation, regards the possibility of efficiently dealing with clashing needs (goals, or constraints, tasks, and resources). Of course, as far as a strictly top-down analysis is performed (as currently done in REF), there is no explicit reason to introduce *contrasting* elements in the design. But they may arise as soon as considering the situations discussed above about the possibility of *sharing* sub-goals. As well as during a top-down analysis a top-down introduced sub-goal may be recognized as helpful for another goals (possibly of another actor), in fact, in a similar process the introduced sub-goal may be recognized as (possibly) harmful for another goal. In addition, during the analysis, new agents may have to be introduced into the context (e.g., the *Head of Unit* requires the *Employee* to *be more productive*), and such new agents may express their own needs by introducing new goals in the context. Such goals may very easily clash with other goals already in the context. A clear example is represented by the *Employee*'s soft goal *protect my work performance* (in Figure 5), leading to the constraint that the *ERMS* should not be allowed to monitor and record data regarding the *Employee*'s actions. This clashes with the desired expressed by the *Head of Unit* of knowing the number of documents each *Employee* is dealing with (in Figure 2). It is worth noticing that hurting goals may easily emerge also within the analysis of a single agent. Indeed, as just seen in the discussed example, REF already provide the tool for the detailed recognition of such situations. When fully operationalised in terms of tasks and constraints, goals models in fact can be adopted to detect clashing situations and to resolve them. In this case, for example, the decision could be made of providing the *Head of Unit* only the average of the documents dealt with by all the Employees, so protecting the privacy of the single one. Nevertheless, we think it may be very useful to be able to recognize such situations as early as possible, also at a very qualitative level, before than pushing the analysis down to the final constraints and details. For such a reason, to enable the analysts to mark possible conflicting situations (and build their refinement strategy to deal with them), we introduce the *H-connection*, H for "Hurting". This is a powerful tool to detect possible conflicts and try to reconcile different stakeholders points of view, allowing to evolve the analyses only along the most promising alternatives.

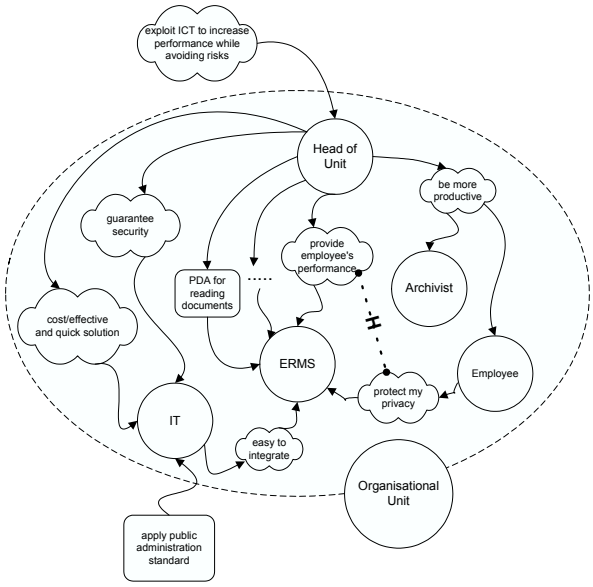

Figure 9 – A possible conflict between goals

An example application is shown in Figure 9, where the H-notation has been used to highlight the possibility of a conflict between two goals before proceeding in their analysis (i.e. the soft goal *provide employee's performance* is not broken down into tasks before taking into account the *protect my privacy* one).

4. CONCLUSIONS

The paper introduced an agent-oriented requirements engineering framework (REF), explicitly designed to support the analysts in reasoning about socio-technical systems, and transform high-level organizational needs into system requirements, while redesigning the organizational structure itself. The underlying concepts and the adopted notations make of REF a very effective and easy to deal with (usable) tool, able to tackle complex real case situations, while remaining simple enough to allow a concrete and effective stakeholders involvement. However, we felt that REF could be improved with the regard to the support it could provide to the analysts in dealing with more complex, and system/organizational design related issues, such as shared and clashing stakeholders' needs. In both cases, an early detection of such a situation could lead to better analysis results: shared needs could be objects of a more intensive analysis effort to exploit commonalities to reduce complexity and increase reusability; clashing needs could be solved at a very early stage, to focus then the analysis only towards the most promising alternatives. Two graphical notations (i.e. S-connection and H-connection) have thus been introduced to allow the analysts to mark such situations and better reason about how to build their strategy. They are pure analyst-oriented tools that do not affect REF usability in terms of stakeholders' perspective, but greatly improve the analysts capabilities of building the strategy to deal with shared and clashing interests. Such notations allow the analysts to easily connect items (goals, constraints, tasks and

resources) across different models and diagrams to support their reasoning about the problems and about how to face them (e.g. whether or not perform a multi-agent elicitation session), but do not have to appear while interacting with the stakeholders. In addition, once analysis is completed, only REF original notation is sufficient. Finally, although REF addresses only the early stages of the RE process, the possibility of combining its outcome with techniques more suitable for dealing with further system development phases has been investigated. For example, practical results suggest that it can be usefully applied as a forerunner to both object-oriented approaches, such as those based upon UML [15], as well as agent-oriented approaches for MAS systems [4,16,21].

5. REFERENCES

[1] A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour, IEEE 5th Int l Symp. on Requirements Engineering, pp. 249-261, August 2001.

[2] Antón, A.I. Goal-Based Requirements Analysis, 2nd IEEE Int'l Conf. on Requirements Engineering (ICRE 96), Colorado, pp. 136-144, 15-18 April 1996.

[3] Basili, V. R., Caldiera, G., and Rombach, H. D., "The Goal Question Metric Approach", Encyclopedia of Software Engineering, Wiley&Sons Inc., 1994.

[4] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J. "A knowledge level software engineering methodology for agent oriented programming". Proceedings of the 5th International Conference on Autonomous Agents, Montreal, Canada, May 2001.

[5] Bresciani,P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A. TROPOS: An Agent-Oriented Software Development Methodology, to be Published on Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers.

[6] Cantone, G., Donzelli P. "Production and Maintenance of Goal-oriented Software Measurement Models". International Journal of Knowledge Engineering and Software Engineering, World Scientific, Vol 10, N 5, October 2000

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

[8] Dardenne, A., Van Lamsweerde, A., Fickas, S. "Goal-Directed Requirements Acquisition" Science of Computer Programming, Vol. 20, North Holland, 1993.

[9] D'Inverno M., Luck, M. "Development and Application of an Agent Based Framework" Proceedings of the First IEEE International Conference on Formal Engineering Methods, Hiroshima, Japan, 1997.

[10] Donzelli P., Setola R., "Handling the knowledge acquired during the requirements engineering process", Proceedings of the Fourteenth International Conference on Knowledge Engineering and Software Engineering (SEKE), 2002.

[11] Donzelli P., Setola R., "Putting the customer at the center of the IT system – a case study" , Proceedings of the Euro-Web 2001 Conference – The Web in the Public Administration, Pisa, Italy, 18-20 December 2001.

[12] Donzelli, P., Moulding M.R. "Application Domain Modelling for the Verification and Validation of Synthetic Environments: from Requirements Engineering to Conceptual Modelling", Proceedings of the Spring 2000 Simulation Interoperability Workshop, Orlando, FL, March 2000.

[13] Donzelli, P., Moulding M.R. "Developments in Application Domain Modelling for the Verification and Validation of Synthetic Environments: A Formal Requirements Engineering Framework", Proceedings of the Spring 99 Simulation Interoperability Workshop, Orlando, FL, March 1999.

[14] Fenton N. E., S. H. Pleeger. Software Metrics: a rigorous and practical approach – second edition, International Thomson Computer Press, UK, 1997.

[15] Fickas, S., Helm, B.R. "Knowledge Representation and Reasoning in the Design of Composite Systems", IEEE Transactions on Software Engineering, Vol. 18, N. 6, June 1992.

[16] Giorgini, P., Perini, A., Mylopoulos, J., Giunchiglia, F., Bresciani, P. "Agent-oriented software development: A case study". Proceedings of the Thirteenth International Conference on Software Engineering and Knowledge Engineering (SEKE), 2001.

[17] Haruhiko Kaiya, Hisayuki Horai, Motoshi Saeki. AGORA: Attributed Goal-Oriented Requirements Analysis Method, IEEE Joint Int'l Requirements Engineering Conf. (RE 02), Essen, 9-13 Sept. 2002.

[18] http://www.w3.org/XML/

[19] Iglesias, C., Garijo, M., Gonzalez, J.C. A survey of agent-oriented methodologies, proceedings of Fifth International Workshop on agent Theories, Architectures, and Languages (ATAL-98), Springer-Verlag: Heidelberg, 1999.

[20] Kendall, E.A., Palanivelan, S., Kalikivayi, S. "Capturing and structuring goals: analysis patterns", European Patterns Languages of Programming, EuroPlop 98, July 1998.

[21] Mylopolous, J., Castro, J. "Tropos: A Framework for Requirements-Driven Software Development", Information System Engineering: State of the Art and Research themes, Brinkkemper, J., Solvberg, A. (eds), Lecture Notes in Computer Science, Springer Verlag, 2000.

[22] Perini, A., Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. "Towards an Agent Oriented approach to Software Engineering". Proceedings of WOA 2001 - Dagli oggetti agli agenti: tendenze evolutive dei sistemi software, Modena. Pitagora Editrice Bologna. September 2001.

[23] Wood, M.F., DeLoach, S.A. An overview of the multiagent system engineering methodology, Proceedings of the First International Workshop Agent-Oriented Software Engineering (AOSE-2000), Cianciarini, P. and M. Wooldridge (ed), Springer-Verlag: Limerick, Ireland, 10 June 2000.

[24] Yu E., Mylopoulos, J. "Using Goals, Rules, and Methods to Support Reasoning in Business Process Reengineering". International Journal of Intelligent Systems in Accounting, Finance and Management. (5) (1),1996.

[25] Yu, E. "Why Agent-Oriented Requirements Engineering" Proceedings of 3rd Workshop on Requirements Engineering For Software Quality, Barcelona, Catalonia, June 1997.

[26] Yu, E. Modeling Strategic Relationships for Process Reengineering, PhD Thesis, Department of Computer Science, University of Toronto, Toronto, 1995.