Agents, Goals, and Quality in a Structured Requirements Engineering Framework - a case study -

Paolo Donzelli

Presidenza del Consiglio dei Ministri
Ufficio per l'Informatica, la Telematica e la Statistica
Via della Stamperia 8, 00187 Roma, Italy
p.donzelli@governo.it

Abstract. The paper presents a process modelling-based Requirements Engineering Framework, where advanced requirements engineering techniques are combined with software quality modelling approaches to better assist and drive analysts and stakeholders to an early definition and validation of the desired system functionality and quality attributes, while supporting the redesign of the application context to better exploit the new system's capabilities.
As a case study, the framework is applied to support the requirements engineering process for a Synthetic Environment. Synthetic Environments are complex software-intensive simulation systems, which are increasingly used to support vitally important operational, political and economic decisions, in a variety of industrial and governmental settings.

1 Introduction

As technologies advance, empowering designers to envision systems that increasingly tend to become integral parts of the encompassing organisational processes, attention is being more and more focused on the very early phases of Requirements Engineering (RE). The development of a successful system relies, in fact, on a firm understanding of the of the organisational context in which the system has to function: RE has to treat the system and its context as a larger social-technical system, whose overall needs are the ones to be fulfilled [1].

Consequently, in RE appropriate process modelling techniques are typically advocated [2,3], for helping understand the organisational context (as it exists), envision possible solutions (as it could exist with the new system in place), and compare feasible alternatives.

In such a perspective, the paper introduces a *process modelling-based Requirements Engineering Framework*, where requirements engineering techniques (agents and goals), and quality modelling methods, are combined to support discovery, verification and validation of both user-oriented and organisation-oriented system requirements [4,5], facilitating and driving continuous dialogue and negotiation between the analysts and the stakeholders.

The remainder of this paper is organised as follows. Section 2 introduces the Framework, by briefly describing its main characteristics. Section 3 elaborates some of the aspects introduced in Section 2, and specifies the adopted notation, by showing some extracts from a practical case study. Finally, Section 4 concludes by discussing some of the observed benefits and future activities.

2. The Requirements Engineering Framework

The framework is designed to allow the analysts to deal with the WHAT and the HOW of the organisational context (i.e. the tasks performed by the organisation, and the way in which they are performed), but also to explicitly model the WHY (i.e. the underlying reasons), expressed in terms of organisational goals. This enables the analysts and the stakeholders to focus on the 'right' system for a given context, to design (or re-design) the encompassing process that fully exploits the system's capabilities [6], and to improve their capacity to identify, justify and validate the system requirements [2,3,7].

The framework tackles the modelling 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 organisation [2,8]. The organisational context is modelled 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 organisational goals. The agent abstraction provides the analysts with a powerful modelling mechanism, fundamental in eliciting, through an effective involvement, the intentions, the knowledge and the behaviours the "real" organisational actors. *Goals* [2,3,9] are used to model agents' relationships, and, eventually, to link organisational 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 (e.g. "buy a computer"); 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 (e.g. "buy a fast computer"). In comparison to hard goals, soft goals can be highly subjective and strictly related to a particular context; they enable the analysts to highlight quality issues (e.g. the concept of "fast computer") from the outset, making explicit the semantics assigned to them by the stakeholders. While a hard goal will lead to a set of functions (functional requirements) that the software system will have to provide, a soft goal will be refined into more precise sub-ordinate soft goals, for reducing its initial fuzziness, until, eventually, it can be expressed as a set of hard goals and constraints upon them (non-functional requirements).

Distinguishing goal modelling from organisational modelling, and then, further distinguishing between hard goal modelling and soft goal modelling, helps to reduce the complexity of the modelling effort. The proposed framework, therefore, supports three inter-related modelling efforts (see Figure 1): the organisational modelling, the hard goal modelling and the soft goal modelling.

During *Organisation Modelling*, the organisational context is analysed and the agents and their 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 other agents for a specific goal, and might clash on some other ones. The found goals will then be refined, through interaction with the involved agents (i.e. the stakeholders), by hard goal and soft goal modelling.

The *Hard Goal Modelling* 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 activity, leaving little room for initiative). Of course, the granularity of refinement of a hard goal into subordinate hard goals and tasks depends on the level of capability and autonomy of the agent. So, for example, while the hard goal "buy a computer" could be clear enough for a human agent, it might need to be translated within an organisation into a set of actions (tasks), necessary to procure a computer: “organise a competitive tender”, “inform the bidders”, etc.

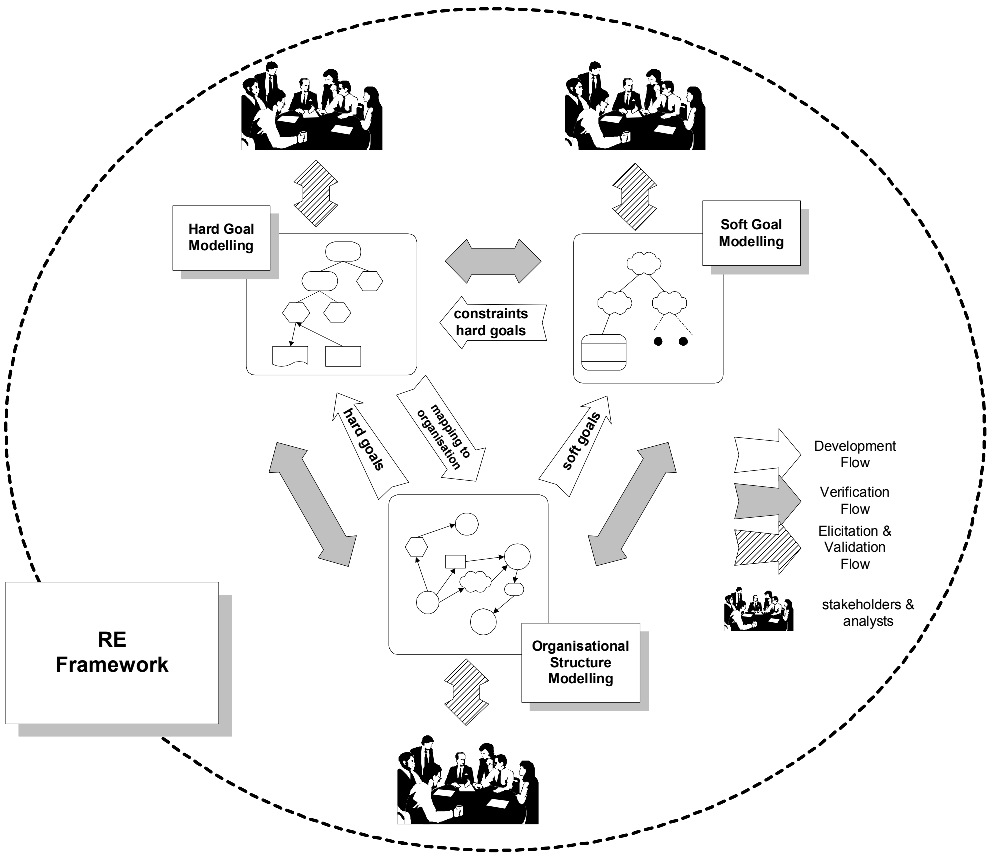


Fig.1. The Requirements Engineering Framework.

The *Soft Goal Modelling* aims at producing the operational definitions of the soft goals that emerged during the organisational modelling, sufficient to capture and make explicit the semantics that are usually assigned implicitly by the involved agents [10,11,12] 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. Resulting soft goal, in turn, will have to be progressively refined until a set of hard goals, tasks and constraints is obtained (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 “buy a fast computer”, beside spawning the hard goal “buy a computer”, will lead also to a set of constraints (e.g. CPU speed, memory size, cache characteristics, etc.) specifying the concept of fast computer. 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 measurement models that formalise the attributes for the specific context [10,11].

As shown in Figure 1, the three discussed modelling activities do not exist in isolation, rather they are different views of the same modelling effort. In particular, as shown by the *Development Flows*, information discovered in one model may feed the others, in a continuous loop. The analysts can in such a way handle in a controlled fashion a large amount of information, while building a model of the organisational context at the desired level of detail. Through continuous interaction with the stakeholders, supported by the different models, the analysts will deal first with the high level organisational structure, then will descend step by step into the details of application context, until the focus will be placed upon the single agents and their role within the organisation.

The other two other kind of information flows shown in Figure 1 are the Verification and the Elicitation & Validation Flows. *Verification Flows* indicate activities that attempt to guarantee consistency between models, for example, to verify that all the hard goals emerging from the soft goal models are properly addressed by the hard goal models, whereas *Elicitation & Validation Flows* show where interaction with the stakeholders occurs.

Although strongly based upon I*, the modelling framework suggested by Eric Yu et al. [2], it is worth noting that the Framework suggested in this paper differs from it in many respects. Apart from leaving out some of the more sophisticated mechanisms, e.g. the distinction between agent, role and position, the proposed Framework introduces, for example, some conceptual (and therefore notational) changes regarding the "dependency links". A dependency link, in fact, as shown in the next Section, is here used to establish a connection between an active modelling item (i.e. agent) and a passive one (i.e. soft and hard goals, tasks and resources): a dependency between agents as described by I* can therefore be established only by combining dependency links. This new approach guarantees more flexibility in dealing with modelling problems, for example to represent multi-agent dependencies, to build the model through continuous refinement and to bridge the different modelling efforts. The main difference, however, relies upon the role played by soft goals. Soft goals, in fact, as suggested by I* can provide strong support in reasoning between alternatives since the early stages of a project, but they can also provide a systematic and organise way of representing and handling non-functional requirements, while dealing with functional ones. The proposed Framework, thus, clearly recognises this capability, making of soft goal modelling, together with organisation modelling and hard goal modelling, one of the main enterprise modelling efforts. In particular, by exploiting methods proper of quality modelling approaches [10,11,12], developed and normally used in isolation, soft goals are explicitly "resolved" and translated into implementable (and agreeable upon) functionalities and constraints (non-functional requirements), concerning both the organisation and the target software system.

3. Applying the Requirements Engineering Framework

Synthetic Environments (SEs) are complex software-intensive systems [5], which usually combine real equipments and real (human) operators with computer-based simulation models to provide greater flexibility and capability in supporting various aspects of a business enterprise. The classical application for SEs is training, where they offer potential cost and flexibility benefits. However, they have also been exploited in a number of other application areas: from equipment procurement (to inform the various life-cycle phases from feasibility to disposal), to operational support, to policy and plan formulation.

SEs are becoming vitally important in a variety of industrial and government settings, as knowledge acquisition and decision support tools, leading to a growing awareness of the need to define and validate them adequately [4]. It is crucial, in fact, that the requirements engineering process for a SE be able to ensure that the SE is suitable for its intended use. In other words, it is crucial that the type, the amount, and the quality of the information provided by a SE reflect the specific needs of the business process it supports. For such a strict interdependency with its application context, a SE can be considered as a suitable test bed for the introduced requirements engineering framework.

3.1. The case study

In the following, the framework is applied to support the requirements engineering process for a hypothetical SE, which is needed to investigate the feasibility of providing an aircraft with a infrared pod: an infra-red camera which is normally used on aircraft and helicopters dedicated to "search & rescue" and "anti-fire" roles to improve the air crew vision capability. Although hypothetical, the case study is firmly based upon a real application, where SEs have been used for a similar purpose [13]. In this paper, for brevity, the presentation is limited to a few extracts from this example application [5].

In Figure 2, a simple organisation model, representing the initial problem, is shown. Circles represent Agents, and dotted lines are used to bound the internal structure of complex agents; that is, agents containing other agents. Consequently, in Figure 2, the *Feasibility Study* is a complex agent, modelling the business process that has to investigate the feasibility of providing an aircraft with the infrared pod, whereas the *Project Leader* is a simple agent, acting within the *Feasibility Study*.

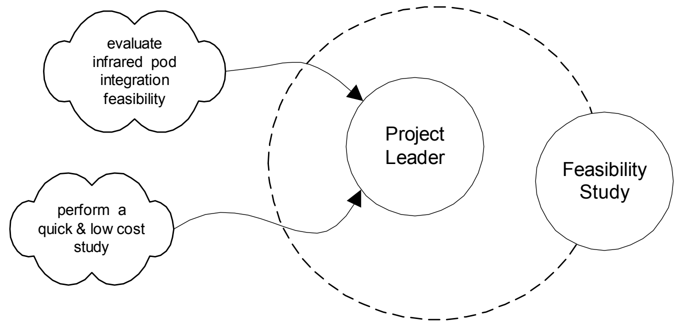

evaluate infrared pod integration feasibility
perform a quick & low cost study
Project Leader
Feasibility Study

Fig. 2. The initial organisational model to perform the Feasibility Study.

Agents interact by exchanging goals and tasks. As shown in Figure 2, soft goals are represented by clouds, whereas, as show in the next Figures, hard goals are represented by rounded-rectangles and tasks by hexagons. Goals, tasks and agents are connected by dependency-links, represented by arrowhead lines. An agent is linked to a goal when he needs 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 has to perform the task. By combining dependency-links, we can establish a dependency among two or more agents [2]. Thus, in Figure 2, the *Project Leader*, that here we have assumed as the agent in charge of the feasibility study, receives, from the enclosing context (out of the scope of our analysis), the soft goals "evaluate the infrared pod integration feasibility" and "perform a quick and low cost study". These goals can be seen as the result of the decision, made at a higher organisational level, of developing a feasibility study to gather more information before proceeding in the project.

Each agent works as a goal transformer. Having received a goal, an agent will operate according to its own experience, knowledge or position within the organisation, in trying to achieve the goal. It will decide how to achieve the goal in terms of tasks and subordinate goals, and may choose to depend on other agents, by passing out some of these tasks and subordinate goals. An agent's behaviour can be analysed through and captured by goal modelling. As example, Figure 3 shows the "evaluate infrared pod integration feasibility" soft goal model, representing how the *Project Leader* will act to achieve the corresponding goal.

In particular, Figure 3 shows how the soft goal "evaluate infrared pod integration feasibility" spawns three subordinate soft goals: "evaluate integration risks", "evaluate final integration time & cost", and "evaluate technical feasibility". These subordinate soft goals are further decomposed. In reverse order, the last soft goal leads to further subordinate soft goals such as "evaluate avionics integration feasibility", and "evaluate software integration feasibility". The soft goal "evaluate final integration time & cost" leads to the hard goals "estimate final software integration time & cost", and "estimate final avionics integration time & cost", together with the associated constraint (a rounded-rectangle with one line) stating that a 20% error (for a feasibility study) can be tolerated. Finally, the soft goal "evaluate integration risks" leads to the soft goal "evaluate compatibility with other projects", indicating the need of assessing this project in the wider contexts of all the activities aimed at improving the aircraft.

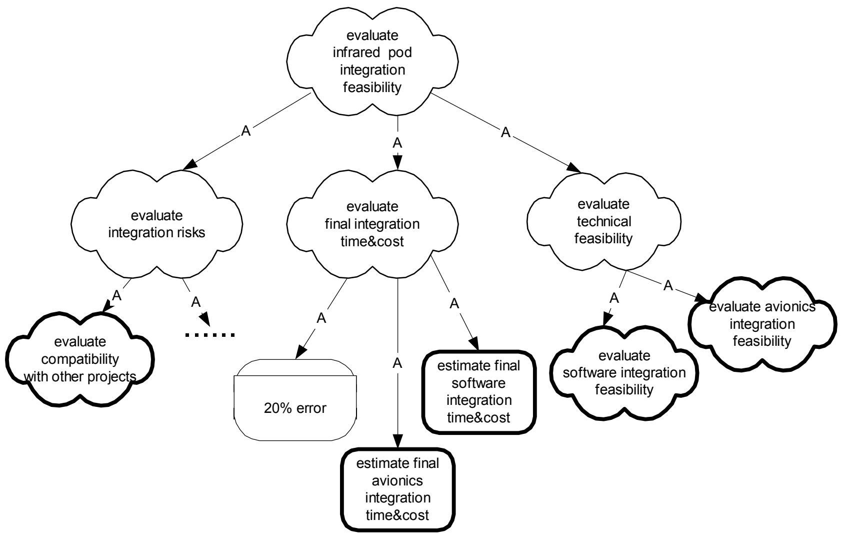

evaluate infrared pod integration feasibility
A
A
A
evaluate integration risks
evaluate final integration time&cost
evaluate technical feasibility
A
A
A
A
A
A
A
evaluate compatibility with other projects
20% error
estimate final software integration time&cost
estimate final avionics integration time&cost
evaluate software integration feasibility
evaluate avionics integration feasibility

Fig.3. The "evaluate infrared pod integration feasibility" Soft Goal Model.

Again, as in Figure 2, 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. A dependency link tells us that some kind of dependency exists, but it does not go any further. For example, it does not allow the modeller to specify alternative, or collaborating refinement paths. For this reason, when necessary, a dependency link can be refined into an AND-link or into an OR-link. As shown in Figure 3, the "A" annotation on each dependency link indicates that all the corresponding goals and constraints must be satisfied ("AND" refinement); as it will be shown in the next Figures, the "O" annotation on each dependency link indicates that at least one of the corresponding goals and constraints must be satisfied ("OR" refinement).

Thus, the soft goal model in Figure 3 shows how the abstract concept of "feasibility" can be reified, making it relevant to the specific context, captured and formalised, making it eventually reusable in similar projects.

In Figure 3, the items in bold outline are those that the *Project Leader* will pass out, having decided to depend on other agents for their achievement. For such a reason, they are not further analysed, instead they will be refined as further agreement between the *Project Leader* and the agent that will be appointed of their achievements. These may be items for which the *Project Leader* thinks he better get some support, or which may be out of the scope of his responsibility. So for example, the *Project Leader* can decide to ask support to an expert of avionics (e.g. to "evaluate avionics integration feasibility"), to get help from a software expert (e.g. to "evaluate software integration feasibility"), and to signal to the external context the necessity to "evaluate the compatibility with other projects".

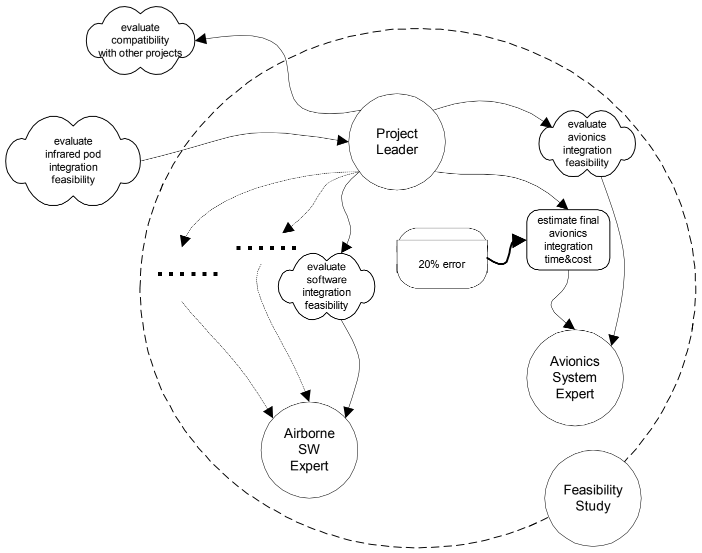
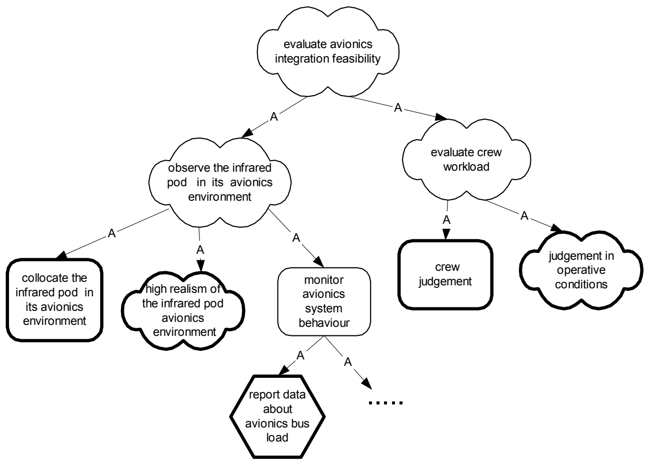

evaluate compatibility with other projects
evaluate infrared pod integration feasibility
Project Leader
evaluate avionics integration feasibility
estimate final avionics integration time&cost
20% error
evaluate software integration feasibility
......
......
Avionics System Expert
Airborne SW Expert
Feasibility Study

Fig.4. The enriched organisational model to perform the Feasibility Study.

The results of this analysis allow us to enrich the initial organisation model in Figure 2, leading to the model in Figure 4, where two new agents have been introduced, the *Avionics System Expert* and the *Airborne SW Expert*. In particular, Figure 4 shows how the *Project Leader* has decided to depend upon the *Avionics System Expert* for the soft goal "evaluate avionics integration feasibility" and the hard goal "estimate the final avionics integration time & cost" (with a constraint stating that a 20% of error can be tolerated), and upon the *Airborne SW Expert* for the soft goal "evaluate the software integration feasibility". In addition, it is also shown how a new goal is emerging from the *Feasibility Study*, i.e. "evaluate compatibility with other projects", placing new responsibilities upon the encompassing context.

At this point, the analysis can be carried on by analysing, for example, how the *Avionics System Expert* will try to achieve the soft goal "evaluate avionics integration feasibility". The resulting soft goal model is shown Figure 5.



evaluate avionics integration feasibility
A
A
observe the infrared pod in its avionics environment
evaluate crew workload
A
A
A
A
A
collocate the infrared pod in its avionics environment
high realism of the infrared pod avionics environment
monitor avionics system behaviour
crew judgement
judgement in operative conditions
A
A
report data about avionics bus load
.....

Fig.5. The "evaluate avionics integration feasibility" Soft Goal Model.

In order to achieve the received soft goal "evaluate avionics integration feasibility", the *Avionics System Expert* will need to "observe the infrared pod in its avionics environment", and to "evaluate the crew workload". The first soft goal will spawn some precise goals. That is, the hard goal "collocate the infrared pod in its avionics environment", with the associated soft goal "high realism of the infrared pod avionics environment", and the hard goal "monitor avionics system behaviour", which leads, among the others, to the task "report data about avionics bus load". The second one, instead, to be achieved, will require a "crew judgment" hard goal and the soft goal "judgment in operative conditions".

Again, as in Figure 3, also in Figure 5, the goals and tasks in bold outline are those that the *Avionics System Expert* will pass out, and they are not further refined. In particular, see Figure 6, the agent thinks that these goals can be accomplished only through the availability of a synthetic environment.

The results of the analysis in Figure 5 (and of a similar one performed for the Airborne SW Expert) have been used to produce the further refined organisation model illustrated in Figure 6. Here a new agent *Synthetic Environment* has been added, and it is shown how the *Project Leader*, the *Avionics System Expert* and the *Airborne Software Expert* interact with each other and with the new agent to collect the information necessary to assess the feasibility of equipping the aircraft with the infrared pod.

At this point of the analysis, the focus turns upon the *Synthetic Environment*, and again, the analysis has to start from the goals and the tasks placed upon it. In particular, in this case, it is evident that some initial decisions about its structure can be made: the hard goal "crew judgment", in fact, clearly suggest that a new agent, i.e. a crew, will have to be taken into account. It is, in fact, typical of SEs to encompass human operators who have to interact with the simulated environment, but who are not direct stakeholders of the process that the SE is designed to support. In this case, for example, the crew is part of the overall SE, so that the effects of the infrared pod integration on crew performance can be determined, but it is the *Project Leader* who is the primary (feasibility study) process owner. In other words, the *Synthetic Environment* may be modelled as a complex agent, encompassing other two simple agents, the *Flight Test Crew* and the *Avionics System Rig* (i.e. a particular kind of ground-based equipment, capable of simulating the characteristics of the aircraft and of its components).

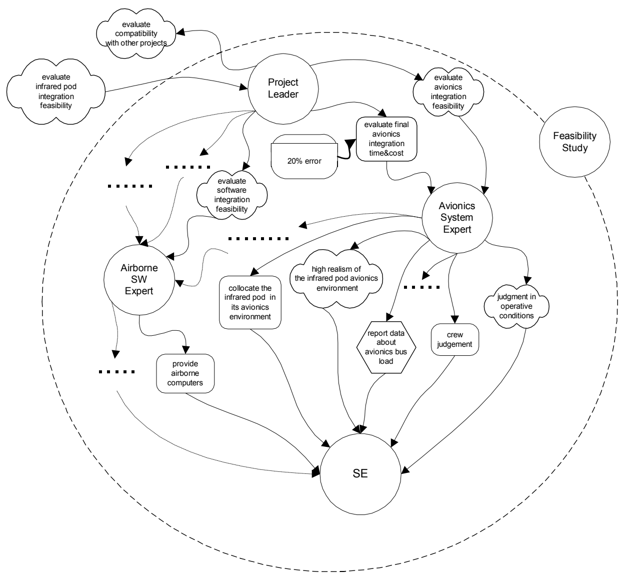

evaluate compatibility with other projects
evaluate infrared pod integration feasibility
Project Leader
evaluate avionics integration feasibility
Feasibility Study
evaluate final avionics integration time&cost
20% error
evaluate software integration feasibility
Avionics System Expert
Airborne SW Expert
collocate the infrared pod in its avionics environment
high realism of the infrared pod avionics environment
judgment in operative conditions
report data about avionics bus load
crew judgement
provide airborne computers
SE

Fig. 6. The even more enriched organisational model to perform the Feasibility Study.

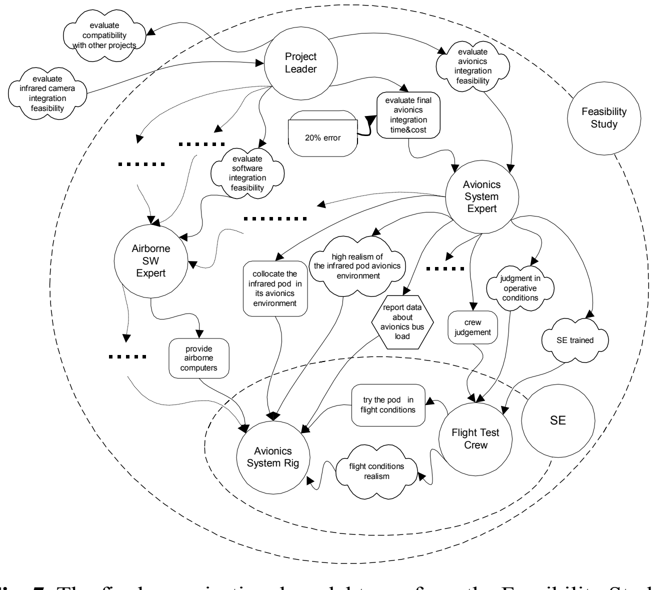

evaluate compatibility with other projects
evaluate infrared camera integration feasibility
Project Leader
evaluate avionics integration feasibility
Feasibility Study
evaluate final avionics integration time&cost
20% error
evaluate software integration feasibility
Avionics System Expert
Airborne SW Expert
collocate the infrared pod in its avionics environment
high realism of the infrared pod avionics environment
judgment in operative conditions
report data about avionics bus load
crew judgement
SE trained
provide airborne computers
try the pod in flight conditions
Flight Test Crew
SE
Avionics System Rig
flight conditions realism

Fig. 7. The final organisational model to perform the Feasibility Study.

As any other agent, also the Flight Test Crew on its turn will be able to generate its own goals. In particular, in order to be able to express its opinion, the *Flight Test Crew* will require the possibility of ‘flying’ the new pod, which places two other goals on the *Avionics System Rig*: a hard goal, “try the pod in flight conditions”, and a soft goal, “flight condition realism”. The resulting, and final organisational model is shown in Figure 7.

Finally, by modelling and refining the goals imposed on the Avionics System Rig, its requirements can be obtained. For example, by modelling the soft goals “high realism of the infrared pod avionics environment” and “flight conditions realism”, a clear idea of the needs of the Avionics System Expert and Flight Test Crew agents can be gained and translated into functional and non-functional requirements.

These two goals, in particular, provide the occasion to show how soft goals can act as a useful tool to detect and resolve clashing requirements, as discussed in the next section.

3.2. Soft goals as a mean to detect and resolved clashing requirements

During Soft Goal Modelling the analysts and the stakeholders tend to clarify very early in the project concepts that are usually left blurred until implementation force to make some choice. Soft goals become a knowledge representation vehicle that: 1) encourages the interaction between the analysts and the stakeholders, and among the stakeholders themselves; 2) leads towards a common terminology; 3) supports reasoning about trade-offs; 4) allows freezing temporary solutions, and formalising final decisions.

Soft goal models therefore allow the analysts to early detect clashing requirements, which usually are hidden behind generic and left-implicit assumptions, and, at the same time, provide an operational way to resolve them, by reconciling the different stakeholders’ points of view.

As example of this capability, let’s turn to Figure 7, and analyse the soft goal “flight conditions realism“ placed upon the *Avionics System Rig* by the *Flight Test Crew*. In other words, we have to understand what the *Flight Test Crew* means by “flight conditions realism”, and to see if this is compatible with the view of the other agents, for example the *Avionics System Expert*. The answer may be found by analysing the soft goal models in Figures 8 and 9.

Figure 8 illustrates what the *Flight Test Crew* means when it requires realistic flight conditions. Here, for the sake of the example, only two of the possible related aspects (subordinate soft-goals) are taken into account: the realism of the adopted sensors and aircraft simulation models, and the realism of the flight crew interface. For the first point, the *Flight Test Crew* requires to be able to fly the complete aircraft flight envelope, and to employ quite precise altitude sensors, being aware of the relevance of the new system in low-altitude missions. For the second point, an extensive use of real equipments is required. For example, the crew will require the *Avionics System Rig* to have “real stick, throttle and selectors” and “real displays”, to have a “wrap-around projected display” to show the external environment and to have a motion platform.

Figure 9, instead, illustrates what the *Avionics System Expert* means when he/she requires a realistic infrared pod avionics environment. In this example, only three subordinate sub-goals are analysed. One, the “avionics systems realism”, is matter of concern of only the *Avionics System Expert*, so no conflicts with the *Flight Test Crew* arise. The other two lead instead to problems, as highlighted in both the figures by bold arrows.

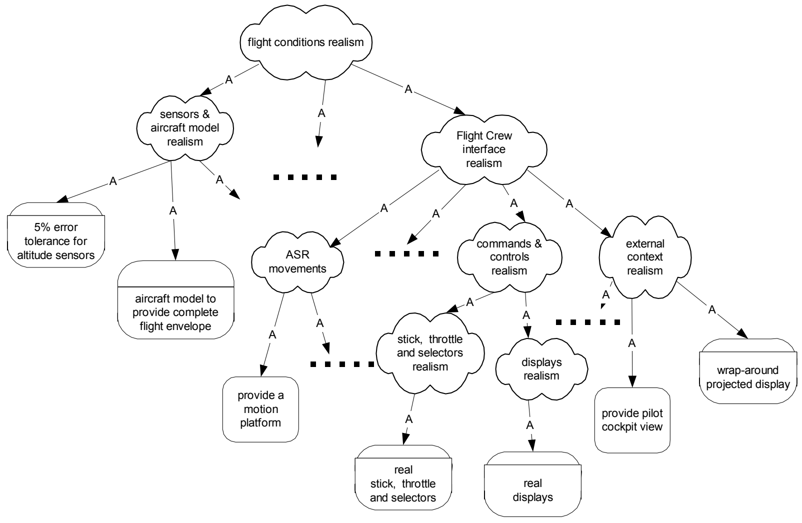

flight conditions realism
A
A
A
sensors & aircraft model realism
Flight Crew interface realism
A
A
A
5% error tolerance for altitude sensors
aircraft model to provide complete flight envelope
A
A
A
A
ASR movements
commands & controls realism
external context realism
A
A
A
A
A
A
A
provide a motion platform
stick, throttle and selectors realism
displays realism
provide pilot cockpit view
wrap-around projected display
A
A
real stick, throttle and selectors
real displays

Fig.8. "Flight conditions realism" for the *Flight Test Crew.*

In particular, the *Avionics System Expert* does not assume the availability of a complete flight envelope as relevant for the project, and thinks that a 10% error for the sensors is more than sufficient. Furthermore, he or she is open to various implementation solutions for what concerns the crew interface, at different levels of cost and complexity. For example, for the *Avionics System Expert*, there is no difference in using a real stick or a synthetic one (e.g. a computer joystick).

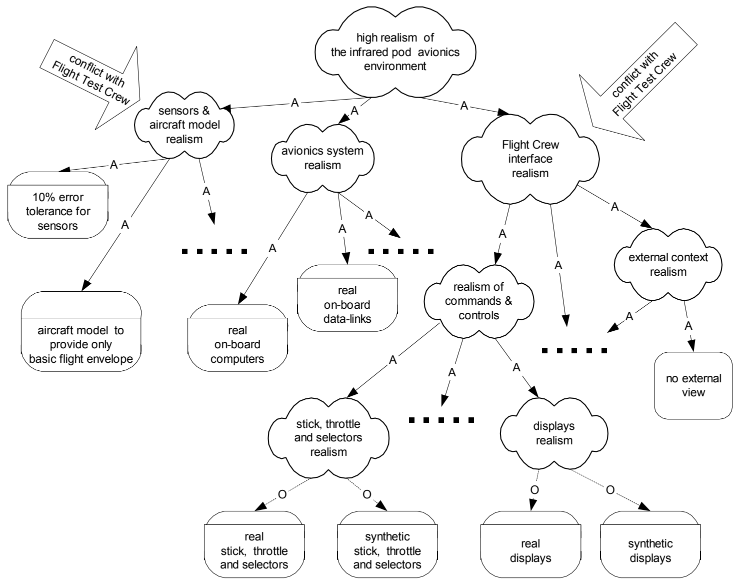

high realism of the infrared pod avionics environment
conflict with Flight Test Crew
conflict with Flight Test Crew
sensors & aircraft model realism
avionics system realism
Flight Crew interface realism
A
A
A
A
10% error tolerance for sensors
aircraft model to provide only basic flight envelope
real on-board computers
real on-board data-links
realism of commands & controls
external context realism
no external view
stick, throttle and selectors realism
displays realism
O
O
O
O
real stick, throttle and selectors
synthetic stick, throttle and selectors
real displays
synthetic displays

Fig.9. "High realism of the infrared pod avionics environment" for the *Flight Test Crew.*

In this situation, it is therefore clear how soft goals models may help in identifying possible conflicts, by highlighting similar problems, issues or concepts that different agents intend to tackle in different ways. In addition, soft goals models also provide a pragmatic way of reasoning about trade-offs [14], and formalising results. The "resolved avionics environment realism" soft goal model is illustrated in Figure 10.

So, for example, for what concerns the crew interface, we can assume that, due to cost limitations (see the initial "perform a quick and lost cost study" soft goal of Figure 2) an interface less complex than the one required by the Flight Test Crew would be implemented: to employ real displays only when they are strictly related with the use of the infrared pod, to use a flat-screen (rather than a wrap-around projected one) to show the external environment, and to avoid proving a motion platform. This choice, although saving on the equipment, would lead to the need of employing only crews specially trained in the use of SEs. Such a need is captured by the soft goal "SE trained" that appears in Figure 7. By further refining this goal, specific constraints and sub-ordinate goals defining the necessary crew skill can be obtained.

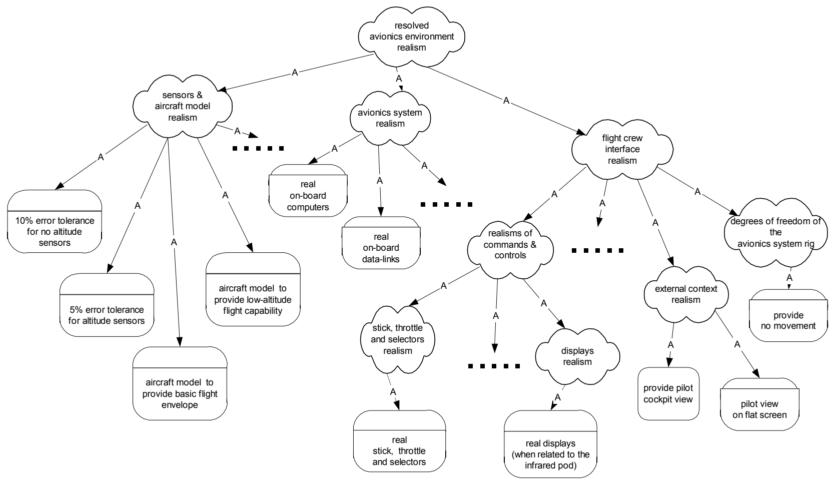

resolved avionics environment realism
sensors & aircraft model realism
avionics system realism
flight crew interface realism
10% error tolerance for no altitude sensors
5% error tolerance for altitude sensors
aircraft model to provide basic flight envelope
aircraft model to provide low-altitude flight capability
real on-board computers
real on-board data-links
realisms of commands & controls
stick, throttle and selectors realism
real stick, throttle and selectors
displays realism
real displays (when related to the infrared pod)
external context realism
provide pilot cockpit view
pilot view on flat screen
degrees of freedom of the avionics system rig
provide no movement
A

Fig.10. The Resolved "Avionics System Rig Realism" Soft Goal Model.

For what concerns the realism of the sensors and of the aircraft model, instead, the importance of performing low-altitude flights expressed by the Flight Test Crew has been recognised by the Avionics System Expert, yielding, for example, to a stricter constraint for altitude sensors (5% error tolerance).

3.3. Further development of the *Avionics System Rig*

The Framework allows the analysts to model the application context at a social-technical level, by providing a systematic approach to deal with agents, soft goals, hard goals, and their incremental refinement. Starting with the high-level organisational goals, through enterprise modelling, the main requirements for the target software system, the Avionics System Rig, can be obtained, expressed as a collection of hard goals, tasks, and constraints placed upon it by other agents of the organisation. Collectively, these hard goals, tasks and constraints form the final result of the analysis process. As example, some of the hard goals and constraints emerged during the analysis in the previous Sections are reported in Figure 11.

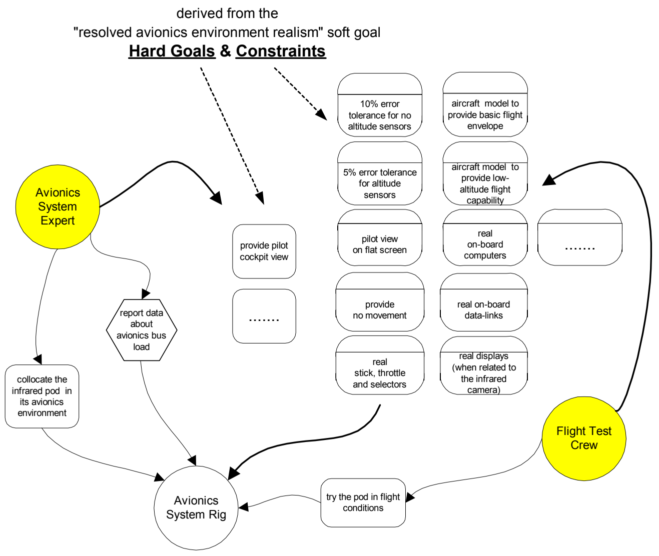

derived from the
"resolved avionics environment realism" soft goal
Hard Goals & Constraints
10% error tolerance for no altitude sensors
aircraft model to provide basic flight envelope
5% error tolerance for altitude sensors
aircraft model to provide low-altitude flight capability
Avionics System Expert
provide pilot cockpit view
pilot view on flat screen
real on-board computers
....
report data about avionics bus load
....
provide no movement
real on-board data-links
collocate the infrared pod in its avionics environment
real stick, throttle and selectors
real displays (when related to the infrared camera)
Flight Test Crew
Avionics System Rig
try the pod in flight conditions

Fig.11. The RE Framework outcome: a Set of Hard Goals, Task and Constraints.

At this point, the analysts have to deal with a purely technical system: the *Avionics System Rig* agent. Although it would be possible to carry forward the Framework social-technical modelling approach within it, which would eventually result in a network of agents performing a set of tasks, the possibility of incorporating techniques more suitable for dealing with the final, and purely technical, system requirements has been investigated.

Initial results show in fact that the Framework can be usefully applied as a forerunner to UML-based [15] approaches. The Framework output, in fact, as set of hard goals and constraints placed upon the target software system by well-defined agents, not only results to be entirely consistent with "use-case modelling" (i.e. the front-end activity of any UML-based approach), but also addresses the need, clearly recognised by both the research community and the business world, that use-case analysis can benefit substantially from supporting methods, that help to identify both the actors, which interact with the system, and their motivating goals.

Although details can be found in [5,16], two main points about the suggested approach are worth of being noticed here. First, UML use-cases offer a systematic and intuitive means of capturing functional-requirements, so *UML use-case modelling results a very effective hard goals refining strategy*, to translate hard goals placed upon the system into functionalities it has to provide. A hard goal can in fact lead to one or more use-cases, where a use-case is a collection of possible sequences of interactions (scenarios) between the actor and the system. Importantly, the possibility of delaying hard goal refinement at the Framework stage, through use-cases at this level, helps to reduce the detail that must be captured at the social-technical level, thereby providing greater clarity and abstraction. Second, the *constraints emerging from the Framework allow to enrich and complete use cases*, by providing not only a systematic and intuitive means of capturing and formalising non-functional requirements, but also a very effective means for combining them with functional requirements. Enriched with constraints and clearly linked to goals and their originators, UML use-cases models result therefore into conceptual models suitable to combine functional and non-functional system aspects, and to better support subsequent development stages.

Some extracts from the *Avionics System Rig* use-case diagram, developed by using the SELECT CASE tool, are illustrated in Figure 12, where only the *Avionics System Expert* agent and his or her related goals and constraints have been taken into account. Here, by super-imposing the Framework notation (dotted lines) on to the UML notation, we can illustrate how use-cases are related to hard goals and constraints. In particular, dotted boxes are used to group together use-cases contributing to the same hard goal (linked by a ‘spark’), dotted arrows are used to link constraints to use-cases within which they should be dealt with, and tasks are simply put close to the use cases that will take their place.

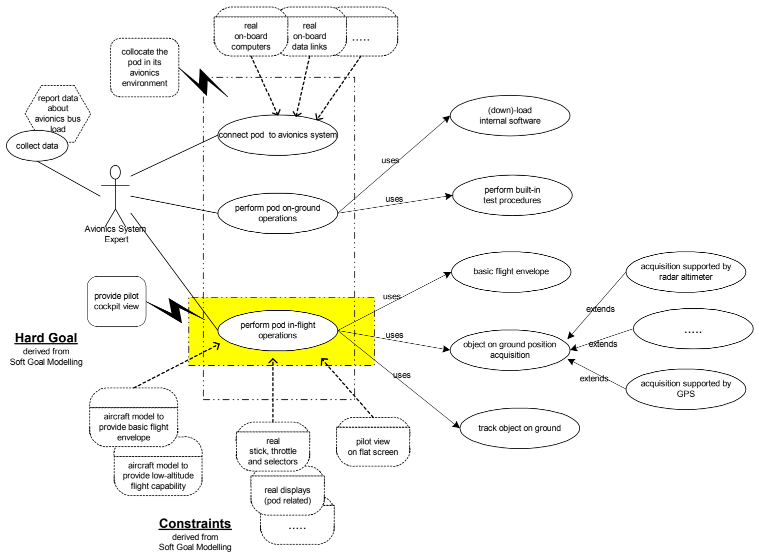

real on-board computers
real on-board data links
.....
collocate the pod in its avionics environment
report data about avionics bus load
collect data
connect pod to avionics system
(down)-load internal software
uses
perform built-in test procedures
uses
perform pod on-ground operations
Avionics System Expert
basic flight envelope
acquisition supported by radar altimeter
provide pilot cockpit view
uses
extends
perform pod in-flight operations
.....
uses
object on ground position acquisition
extends
Hard Goal
derived from Soft Goal Modelling
uses
extends
acquisition supported by GPS
track object on ground
aircraft model to provide basic flight envelope
aircraft model to provide low-altitude flight capability
real stick, throttle and selectors
pilot view on flat screen
real displays (pod related)
.....
Constraints
derived from Soft Goal Modelling

Fig.12. The Use-case Diagram for the *Avionics System Expert*.

First, Figure 12 shows how the whole set of use-cases “connect pod to the avionics sub-systems”, “perform pod on-ground operations”, and “perform pod in-flight operations” contribute to the achievement of the *Avionics System Expert* hard goal “collocate the pod in its avionics environment”. Then, how the use-case “perform pod in-flight operations”, on its own, aims to achieve the hard goal “provide pilot cockpit view”, and how the use-case “collect data” is simply another view of the task “report data about avionics bus load”. Finally, Figure 12 illustrates how the constraints are associated with the various use-cases. For example, the constraints “real on-board computers” and “real on-board data links” will affect the way in which the use-case “connect pod to avionics system” will be performed.

Use-cases may be described in a tabular form [15]. For example, Table 1 gives the descriptions for one of the use-cases introduced in Figure 12. The rows with a dark background contain the basic use-case information; that is, the use-case name, the actor, the description, and the specific intent behind it. Here, for the sake of brevity, only very short descriptions have been given. The other rows represent additional information, including a specification of the hard goal that the use-case contributes to achieving, and the non-functional requirements derived from the constraints.

Table 1. Tabular Description of the Use-case "connect pod to avionics system".

Use Case Feature	Feature Description
Name	Connect pod to avionics system
Actor	*Avionics System Expert*
Description	• Connect power cable • Connect discrete-signal links • Connect bus-data link
Intent	To connect the pod to the avionics system
High-level Goal	*Collocate the Pod in its avionics environment*
Pre-condition	Pod available
Post-condition	Pod connected to avionics system
Non-functional Requirements	• ASR to employ real on-board computers • ASR to employ real data/power cables • <…..>

The further system development according to UML is outside the scope of this work. However more details on how UML can be used as a modelling technique to produce a conceptual (analysis) model of the *Avionics System Rig* are in [16]

4. Conclusions and Future Work

The described application example demonstrates the feasibility of the suggested approach, and the benefits it offers during the early phases of requirements engineering, when the analysts and the stakeholders have to cooperate to understand and reason about the organisational context within which the new system has to function, in order to identify and formalise not only the system requirements, but also the organisational setting that better exploits the new system's capabilities.

In particular, benefits can be observed in terms of requirements discovery and early validation. Discovery and validation are improved because of the visibility of decisions made by the stakeholders, resulting from explicit organisation and goal modelling. Each type of the framework model provides in fact a specific knowledge representation vehicle that the analyst can use to interact with the stakeholders, in order to capture requirements, reason about them and, eventually, reach an accepted formulation. In other terms, soft goal models force the stakeholders to reason about their own concepts of quality (for example, the concept of "realism" as in Figure 10), hard goal models allow the stakeholders to understand and validate their role within the organisation, whereas organisation models provide the management with a clear view of how the business process will be changed, or affected, by the introduction of the new system (see Figure 7). The resulting models also suggest that the Framework offers potential benefits in the post-deployment phase. The clear links established between organisational goals and system requirements, in fact, allow the analysts to quickly identify the influence of organisational changes on the final system requirements, supporting both system maintenance and re-use in different application contexts.

Finally, the general principles upon which the Framework is based allow it to be deployed for a larger class of computer-based information systems, beyond SEs. For example, in [17], it has been applied to define the requirements for a workflow-based document management system, whereas in [18] it has been adopted to analyse the organisational impact and advantages of introducing a corporate smart card as enabling platform for accessing and using different e-services delivered by the organisational IT system (e.g. digital signature, certified e-mail, documents management systems, etc.).

References

1. 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.
2. Yu, E. "Why Agent-Oriented Requirements Engineering" Proceedings of 3rd Workshop on Requirements Engineering For Software Quality, Barcelona, Catalonia, June 1997.
3. Loucopulos, P., Karakostas, V. System Requirements Engineering. McGraw Hill, UK, (1995).
4. 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.
5. Donzelli, P., Moulding M.R. "A Unified Approach to the Verification, Validation and Accreditation of Synthetic Environments: A Requirements Engineering Framework", Cranfield University Technical Report SE027E/TR1 Version 1, December 1999.
6. Hammer, M., Champy, J. "Reengineering the Corporation: A Manifesto for Business Revolution", Nicholas Brealey Publishing, London, (1995).
7. Dardenne, A., Van Lamsweerde, A., Fickas, S. "Goal-Directed Requirements Acquisition" Science of Computer Programming, Vol. 20, North Holland, 1993.
8. 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.
9. Dardenne, A., Van Lamsweerde, A., Fickas, S. "Goal-Directed Requirements Acquisition" Science of Computer Programming, Vol. 20, North Holland, (1993).
10. Basili, V. R., Caldiera, G., and Rombach, H. D., "The Goal Question Metric Approach", Encyclopedia of Software Engineering, Wiley&Sons Inc., 1994.
11. Cantone, G., Donzelli P. "Goal-oriented Software Measurement Models", European Software Control and Metrics Conference, Herstmonceux Castle, East Sussex, UK, April 1999.
12. Chung, L., Nixon, B., Yu, E., Mylopoulos, J. Non-Functional Requirements in Software Engineering, Kluwer Academic Publishers, (2000).
13. Donzelli, P., Marozza R. "Laser Designation Pod on the Italian Air Force AMX aircraft: a prototype integration", Proceedings of the NATO/RTO SCI Joint Symposium on Advances in Vehicle Systems Concepts and Integration, Ankara, Turkey, April 1999.
14. Van Lamsweerde, A., Darimont, R., Letier, E. "Managing Conflicts in Goal-Driven Requirements Engineering", IEEE Transactions on Software Engineering, Vol. 24, N. 11, November 1998.
15. Rumbaugh, J., Jacobson I., Booch, G. The Unified Modelling Language Reference Manual. Rational Software Corporation, Addison Wesley, UK, 1999.
16. 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.
17. Antonelli, C., P. Donzelli, Mastroianni, R. Setola, S. Vinti, S. Tucci *A Web-based Workflow Solution to support the Italian Government Agenda Definition Process*, Proceedings of the Italian Automatic Computation Association (AICA) 2000 Conference - Le Tecnologie dell'Informazione e della Comunicazione come motore di sviluppo del Paese, Taormina, Italy, 27-30 September 2000.
18. Donzelli P., Setola R., "Putting the customer at the center of the IT system – a case study" , Proceedings of the EuroWeb 2001 Conference – The Web in the Public Administration, Pisa, Italy, 18-20 December 2001.