Author(s): Mouratidis, Haralambos., Giorgini, Paolo., Philp, Ian., Manson, Gordon.
Article Title: Using Tropos methodology to Model an Integrated Health Assessment System
Year of publication: 2002
Citation: Mouratidis, H., Giorgini, P., Philp, I., Manson, G. (2002) 'Using Tropos methodology to Model an Integrated Health Assessment System' Proceedings 4th International Bi-Conference Workshop on Agent-Oriented Information Systems, International Conference on Advanced Information Systems, Toronto – Canada.
Link to online conference proceedings: http://CEUR-WS.org/Vol-57

Using Tropos Methodology to Model an Integrated Health Assessment System

Haralambos Mouratidis [1,2], Paolo Giorgini[3], Ian Philp[2], Gordon Manson[1]

[1] Department of Computer Science, University of Sheffield, England
{h.mouratidis, g.manson}@dcs.shef.ac.uk
[2] Sheffield Institute for Studies on Ageing, University of Sheffield, England
i.philp@shef.ac.uk
[3] Department of Information and Communication Technology, University of Trento, Italy
pgiorgini@dit.unit.it

Abstract. This paper presents a case study to illustrate the features and the stages of the *Tropos* methodology. *Tropos* is an agent-oriented software engineering methodology that covers four development stages: early and late requirements analysis, architectural design and detail design. The electronic Single Assessment Process (eSAP), and electronic system to deliver the integrated health assessment of health and social care needs of older people is used as the case study throughout the paper. Furthermore, a preliminary analysis on extending *Tropos* to accommodate security concerns is presented.

1 Introduction

Computer systems constitute an inseparable part of our everyday life. Technologies in computer systems advance rapidly and they are used in many different areas of human society. Such an area is the Health Care sector. Health Care information systems are becoming more and more computerised. A huge amount of health related information needs to be stored, analysed and with the aid of computer systems this can be done faster and more efficiently.

One of the areas within the Health Care sector that can take advantage of the computerising of the health care systems, is the area related to the care of older people. In a distributed health care setting different health care professionals, such as general practitioners and nurses, must cooperate together in order to provide older persons with appropriate care and must also work closely with social care professionals, such as social workers, because health and social care needs overlap amongst older people. Computerising this process will help to automate some of the tasks of the health and social care professionals and thus leave the professionals with more time for the care of the older people.

We are developing the electronic Single Assessment Process (eSAP), an electronic system to deliver an integrated assessment of health and social care needs of older people. The project is run jointly between the Computer Science Department of the University of Sheffield and the Sheffield Institute for Studies on Ageing (SISA), and it is funded by the RANK Foundation. Analysing and designing such a system is not an easy task. Apart from the complexity of the system itself, another important factor

is the lack of an existing system, either electronic or "human". Thus, apart from trying to understand the functionality of the system, an understanding of the environment of the system is essential.

From a variety of different analysis and design methodologies for agent-based system, we have identified *Tropos* [1, 2] to proceed in our project. This decision took place because of two important advantages that *Tropos* offers in comparison with other existing methodologies (see for instance [3] for an overview on the state of the art). First, *Tropos* covers the early stages of requirements analysis [4], and thus allows for a deep understanding of not only the system itself, but also of the environment where the system will operate and also helps to better understand the interactions that will occur in the system between the software agents and the humans, something which is very important in the eSAP development. Second, *Tropos* covers the full range of the software development phases from the early analysis to the actual implementation [2, 5].

This paper presents results from applying the *Tropos* methodology in the analysis and design of the electronic Single Assessment Process (eSAP) system. Section 2 describes in detail the Single Assessment Process case, which is used throughout the rest of the paper. In Section 3 the *Tropos* methodology is applied in the analysis and design of the eSAP. The early and late requirements, the architectural design and the detailed design stages of the *Tropos* methodology are illustrated. Section 4 describes an initial exploration on extending *Tropos* to accommodate security concerns of the system-to-be. Finally, Section 5 presents some concluding remarks and presents directions for further research work.

2 The Single Assessment Process Case

National policy in England is to promote the Single Assessment Process (SAP), an integrated assessment of health and social care needs of older people. The Single Assessment Process aims to create closer working for providing primary health and social care for older people and other groups. Ultimately, this might lead to the development of Care Trusts, a single local organization for delivering health and social care. The development of integrated health and social care information systems will support closer working and facilitate the development of Care Trusts. In the SAP setting, different health care professionals, such as general practitioners, nurses and social workers, must cooperate together in order to provide patients with appropriate care.

With closer working, professionals will work in teams that will be responsible for the health and social care of the older person. Each team will demonstrate the following characteristics.

- Each team consists of many different professionals.
- Professionals cooperate between them.
- Professionals share information between them.
- Each professional has some expertise.
- Teams will promote person-centred care.

With the single assessment process, a common language of assessment will be used to support information sharing, and the potential to aggregate information to describe the health and social care needs of the local population of older people, which is an extremely useful tool for planning services and monitoring trends in needs and outcomes.

The single assessment process will also provide the older person and their carer with a personal copy of their care plan to support person-centred care. The single assessment process will work in three main stages. The contact, the assessment and the follow-up action. During the first stage contact assessment will provide basic information. In the second phase an overall assessment using a validated assessment instrument, such as Easy-Care [6], will take place. The third stage will provide older people with more care in particular problems (might be different problems for each individual such as house and loneliness problems) and with more detailed assessment if appropriate. The selection of the problems is determined by the results obtained from the Easy-Care assessment instrument.

Computerising this process will help to automate some of the administration tasks, such as the appointments set up between the health and social care professionals, and the management of the health and social care teams, and thus leaving the professionals with more time for the actual care of the older person. Furthermore, it will help older persons to be actively involved in their health and social care, since they will have access to the system.

3 Applying Tropos Methodology

Tropos is a methodology, for building agent-oriented software systems, tailored to describe both the organisational environment of a system and the system itself, employing the same concepts throughout the development stages. *Tropos* adopts the i* modelling framework [7], which uses the concepts of actors, who can be (social) agents (organisational, human or software), positions or roles, goals and social dependencies (such as soft goals, tasks, and resources) for defining the obligations of actors (dependees) to other actors (dependers). This means the system (as well as its environment) can been seen as a set of actors, who depends on other actors to help them fulfil their goals. *Tropos* covers four phases of software development:

Early Requirements, concerned with the understanding of a problem by studying an existing organisational setting; the output of this phase is an organisational model, which includes relevant actors and their respective dependencies;

Late requirements, where the system-to-be is described within its operational environment, along with relevant functions and qualities; this description models the system as a (small) number of actors which have a number of dependencies with actors in their environment; these dependencies define the system's functional and non-functional requirements;

Architectural design, where the system's global architecture is defined in terms of subsystems, interconnected through data and control flows; within the framework, subsystems are represented as actors and data/control interconnections are represented as (system) actor dependencies;

Detailed design, where each architectural component is defined in further detail in terms of inputs, outputs, control, and other relevant information. *Tropos* is using elements of AUML [8] to complement the features of i*.

3.1 Early Requirements Analysis

As it was mentioned above, during the early requirements analysis the analysis engineer models the goals and the dependencies between the stakeholders (actors). For this purpose, *Tropos* introduces actor diagrams. In such a diagram each node represents an actor, and the links between the different actors indicate that one depends on the other to accomplish some goals. The electronic Single Assessment Process (eSAP) project includes, among the others, the following stakeholders:

- Older Person (OP) is the patient that wishes to receive appropriate health and social care.
- Department of Health (DoH) is the government department that must provide older people with health and social care.
- R&D Agencies are agencies that wish to obtain information about older people to help them in their research.

These actors along with their goals are shown in Figure 1. The ***Older Person*** actor has a main goal to receive appropriate health and social care and a soft goal to *Maintain Good health*. The ***Older Person*** depends on the ***DoH*** to accomplish their goal and soft goal. On the other hand, ***DoH*** has a main goal to *Provide Better Health and Social Care to Elderly*. The ***R&D Agency*** has an associated goal to *Obtain Patient Information For Research*.

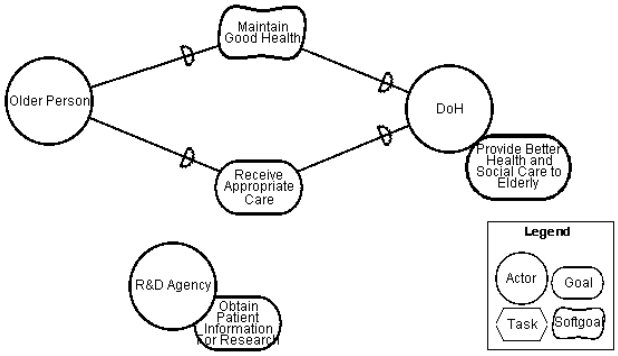

Fig. 1 The Stakeholders of the eSAP

When the stakeholders, their goals and the dependencies between them have been identified, the next step of this phase is to analyse in more depth each goal relative to the stakeholder who is responsible for its fulfilment. In doing so, *Tropos* adopts *i** rationale diagrams for analysing the actors' goals through a means-end analysis. Each rationale diagram is presented as a balloon within which the goals and the

dependencies are analysed. Such an analysis from the perspective of the ***Older Person*** is shown in Figure 2.

As it was mentioned, the ***Older Person*** actor has a goal to *receive appropriate care* and as a soft goal to *maintain good health*. The *receive appropriate care* goal is fulfilled by the tasks *Undertake Assessment, Follow Care Plan, Get Info about Care Plan, Obtain Medical Info*, and *Have Appointment with Professionals*. To perform the last three tasks the ***Older Person*** must use the eSAP system, so the last three tasks are decomposed into the goal *Use eSAP System*. In addition, the *Maintain Good Health* soft goal depends on the *Follow Care Plan* and *Undertake Assessment* tasks to be fulfilled. Furthermore, the ***Older Person*** depends on the ***DoH*** to make the *Technology Infrastructure Available*, and also to make the *eSAP System Available* and Easy-to-Use.

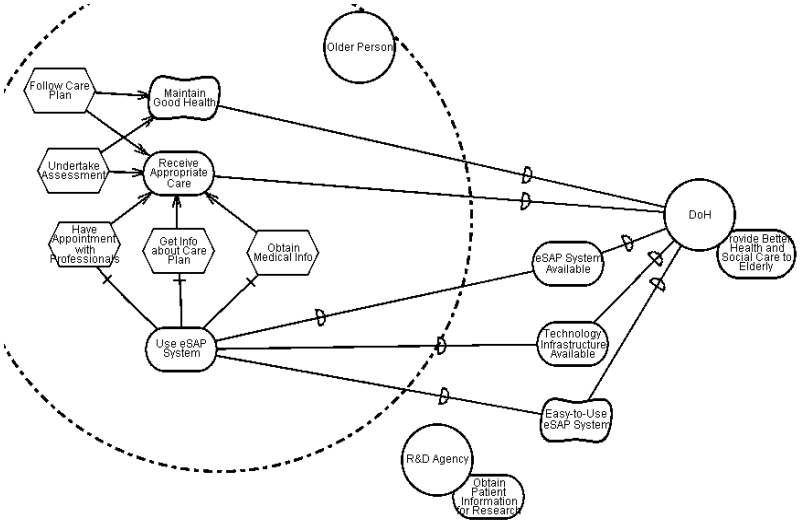

Fig. 2 Means-end analysis for Older Person

Another important stakeholder of the system is the ***DoH***. The rationale diagram for the ***DoH*** is shown in Figure 3. The main goal of the Department of Health is *to Provide Better health and Social Care to Elderly*. To accomplish this goal, ***DoH*** defines a sub goal to *Make Care Person-Centred*. The latter is essential for the ***DoH*** to fulfil its main goal since the ***Older Person*** is the most important participant of the whole procedure, since they know better than anyone their difficulties and when they need health and social care. Thus, the *Make Care Person-Centred* goal is further decomposed into the two sub-goals *Promote SAP* and *Involve Elderly in their Care.* The later sub-goal depends on the task *Provide Guidelines for the Older People* to be fulfilled. The *Promote SAP* sub-goal is decomposed into the *Computerise SAP* goal and the *Provide Guidelines for the SAP* task. The later task is decomposed into two sub-tasks: *Provide Guidelines for the Different Health and Social Care Professionals* and *Provide Guidelines for the Care-Teams*. The first sub-task is further decomposed into four sub-tasks: *Provide Guidelines for GPs, Provide Guidelines for Nurses, Provide Guidelines for Other Professionals* and *Provide Guidelines for Social Workers.* In addition, to fulfil the *Provide Guidelines for the Care-Teams* goal each

locality must comply with the proposed guidelines. *Provide Efficient Care* is another important goal of the ***DoH***. To accomplish this goal the sub goal *Computerise SAP* has been identified. Computerising the SAP will help health and social care professionals to automate some procedures required while caring of the older person and thus help to *Provide Efficient Care*. To accomplish the *Computerise eSAP* sub goal, *Technology Infrastructure* must be provided and the eSAP System must be available. The goal *Build eSAP* is motivated by these two goals since it has no sub-goals.

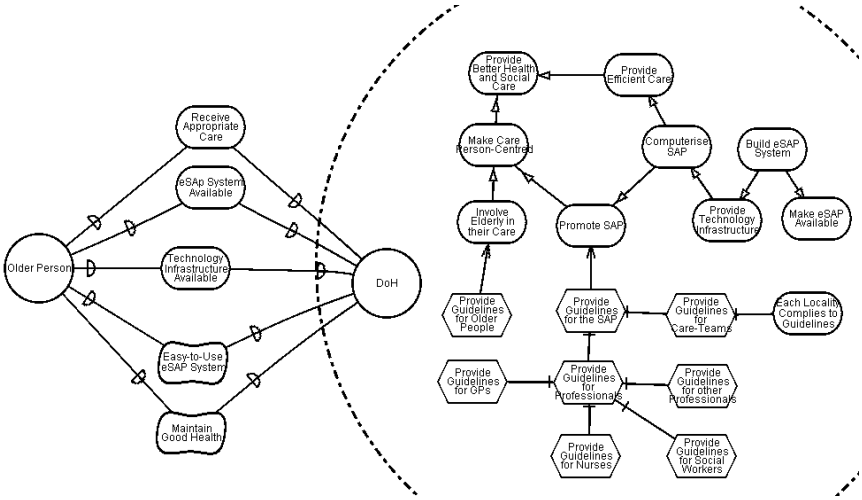

Fig. 3 Means-end analysis for Department of Health (DoH)

3.2 Late Requirements Analysis

During the late requirements analysis the system-to-be (the eSAP system in our case) is described within its operation environment, along with relevant functions and qualities. The system is presented as one or more actors, who have a number of dependencies with the other actors of the organization. These dependencies define all functional and non-functional requirements for the system-to-be.

For the eSAP system the following, amongst others, three requirements have been defined:

1. The professionals must be able to customize their software agents through an easy-to-use interface.
2. The system must be developed with mobility in mind since many of the professionals will use it whilst in the older person's house.
3. The system must be secure.

The eSAP system is introduced as another actor, as shown in Figure 4. The ***DoH*** depends on the eSAP to *Provide Efficient Care*, and also to *Make Care Person-Centred*, in order to fulfil its main goal, which is to *Provide Better Health and Social*

Care. In our example we concentrate on the analysis of the goals *Provide Efficient Care, Make Care Person-Centred,* and *Usable eSAP System*.

The goal *Provide Efficient Care* aims to allow health and social care professionals more time for the actual health and social care of the older person. Plenty of time is currently spent from the health and social care professionals for administrating procedures, such as appointments set up, and communication with colleagues. eSAP will minimise the time spend on these procedures by automating most of the tasks and providing services, such as access to medical libraries, and medical records. Thus, the *Provide Efficient Care* goal is decomposed into the *Provide Services* sub-goal, which is further decomposed into four sub-goals, *Access to Medical Libraries, Access to Medical Records, Schedule Appointments* and *Access to Care Plans*. The later goal is further decomposed into the sub-goal *access care plan info,* which is decomposed into four further sub-goals *Future Care Plan Appointments, Professionals Involved, Previous Assessments,* and *Future Actions*.

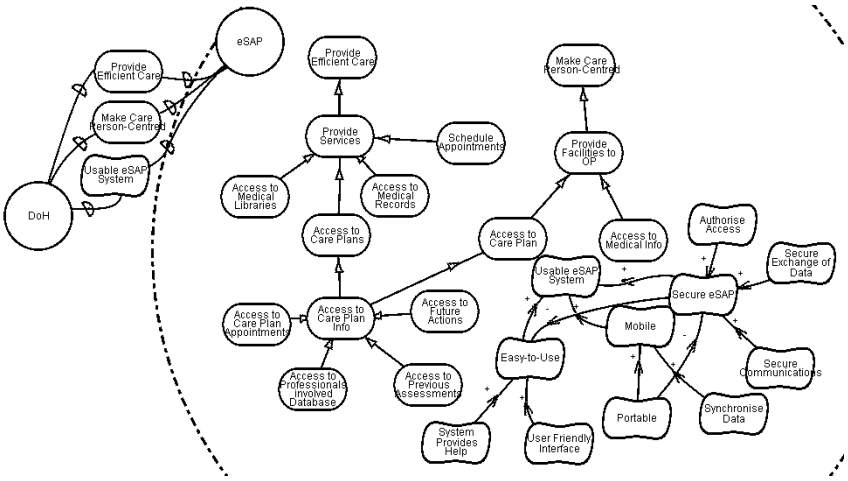

Fig. 4 Means-end analysis for eSAP System

Another important goal of the eSAP system is to promote person-centred care. To fulfil this goal, eSAP must *Provide Facilities to Older People*. The later is decomposed into two further goals *Access to Care Plan* and *Access to Medical Info*. The soft goal *Usable eSAP System* has three positive (+) contributions from the *Easy-to-Use* soft goal, which contributes positively because the system must be Easy-to-Use to be usable, from the *Mobile* soft goal because the system must be mobile to be usable, and also from the *Secure eSAP* soft goal, which contributes positively since it makes the system secure.

The *Easy-to-Use* soft goal has two positive contributions from the *System Provides Help* and the *User Friendly Interface* soft goals. The former contributes positively since the system must help the user to be easy-to-use, and the latter contributes positively because the system must have a *User Friendly Interface* to be easy-to-use. In addition the *Easy-to-Use* soft goal has a negative (-) contribution from the *Secure eSAP* soft goal, since usually trying to make the system secure make it more difficult to be used.

The *Mobile* soft goal accepts two positive contributions from the *Portable* and the *Synchronise Data* soft goals. The former contributes positively because the system must be portable to be mobile, and the latter because the system must be able to synchronise data in order to be mobile.

Furthermore, the *Secure eSAP* soft goal receives three positive contributions. The first positive contribution comes from the *Authorise Access* soft goal, which contributes positively because the system must be able to *Authorise Access* to be secure. The other two positive contributions come from the *Secure Exchange of Data* and the *Secure Communications* soft goals. The former acts positively because the exchange of data must be secured for the system to be secure, and the latter because communications must be secured for the eSAP system to be secure.

In addition, the *Secure eSAP* soft goal has a negative contribution from the portable soft goal because a portable system is more difficult to be secured.

When the system goals and soft goals have been identified, new actors and sub-actors are introduced. Each of the new actors takes the responsibility to fulfil one or more goals of the system. Figure 5 shows a partial decomposition of the actors and sub-actors of the eSAP system, along with their dependencies with respect to the eSAP system.

The eSAP system depends on the ***Medical Library Manager*** to provide *Access to Medical Libraries*, on the ***Medical Record Manager*** to provide *Access to Medical Records*, on the ***Appointments Manager*** to *Schedule Appointments*, and on the ***Care Plan Manager*** to provide *Access to Care Plans*. The ***Care Plan Manager*** depends on the ***Care Plan Appointments Manager*** to Access C*are Plan Appointments*, on the ***Professionals Manager*** to provide information about the professionals involved in the care plan, on the ***Assessments Manager*** to manage *Previous Assessments* and on the ***Future Actions Manager*** to manage the *Future Actions* required by the care plan. Furthermore, the eSAP depends on ***Security Manager*** to fulfil the *Secure eSAP System* goal. The ***Security Manager*** depends on the ***Authorisation Manager*** to Authorise Access to the System, on the ***Confidentiality Manager*** to assure the confidentiality of the data of the system, and on the ***Integrity Manager*** to assure the integrity of the data.

3.3 Architectural Design

The architectural design includes the following four steps:

- *Addition of new actors*, in which new actors are added to make the system interact with the external actors as well as to contribute positively to the fulfilment of some non-functional requirements. *Tropos* introduces the extended actor diagram in which the new actors and their dependences with the other actors are presented.
- *Actor decomposition*, in which each actor is described in detail with respect to its goals and tasks.
- *Capabilities identification*, in which the capabilities needed by the actors to fulfil their goals and tasks are identified, by analysing the extended actor diagram. Each dependency relationship can give place to one or more capabilities triggered by external events.

– *Agents assignment*, in which a set of agent types is defined assigning to each agent one or more different capabilities.

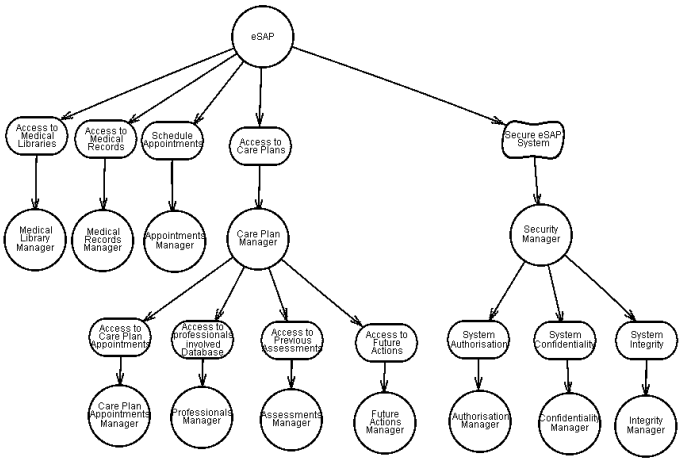

Fig. 5 Sub-Actors Decomposition for the eSAP System

In the eSAP system, the software agents will act on behalf of professionals. Each professional will have his/her "own" software agent, which will be customised according to his/her needs. The agent will have enough information about the professional, such as personal information and professional commitments, and it will be intelligent enough (capable of analysing the information and take decisions) that will enable it to act on his/her behalf, and also negotiate for the interest of the professional. From the above we decided for the following architectural choices:

1. The system will consists of software agents as well as human professionals.
2. Each professional will have his/her software agent.
3. Professionals will be able to customize the software agent according to their needs.
4. The software agent will be capable of analysing information and take decisions. Also, it will have information about the professional (personal and professional) that will be able to act on his/her behalf.
5. Software Agents in the system will be able to communicate between themselves as well as with the human professionals.

Figure 6 shows the extended actor diagram with respect to the ***Authorisation Manager***. The ***Authorisation Manager*** is responsible for authorising access to the ***Older Person (O.P)*** when trying to perform the *Get Info About Care Plan* task. When authorisation is granted the ***O.P.*** interacts with the *Care Plan Manager* in order to obtain available Care Plan Information. Then depending on the kind of information the ***O.P.*** wishes to access, he/she interacts with the appropriate manager. Again the ***Authorisation Manager*** checks for authorisation permissions and grants access to the ***Care Plan Data Manager***, which manages a repository for the care plan information. To represent the interactions of the health and social care professionals within the

system we have introduced the ***Professional*** actor. The ***Professional*** interacts with the ***Authorisation Manager*** to gain the authorisation to interact with the ***Medical Library***, ***Medical Records***, and ***Appointment*** managers. In addition, the ***Authorisation Manager*** interacts with the ***R&D Agency*** actor to grant authorisation to the ***Care Plan Data Manager***.

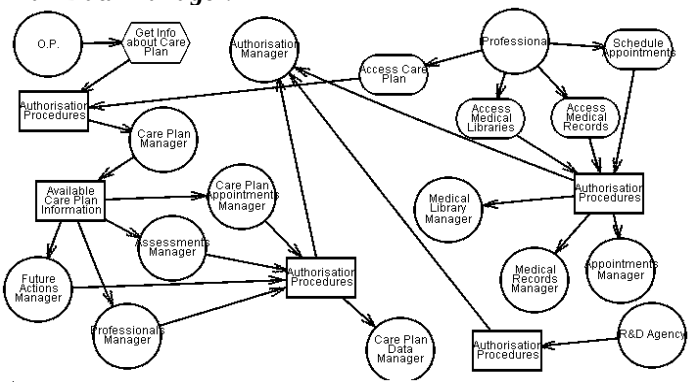

Fig.6 Extended Actor Diagram with respect to the Authorisation Manager

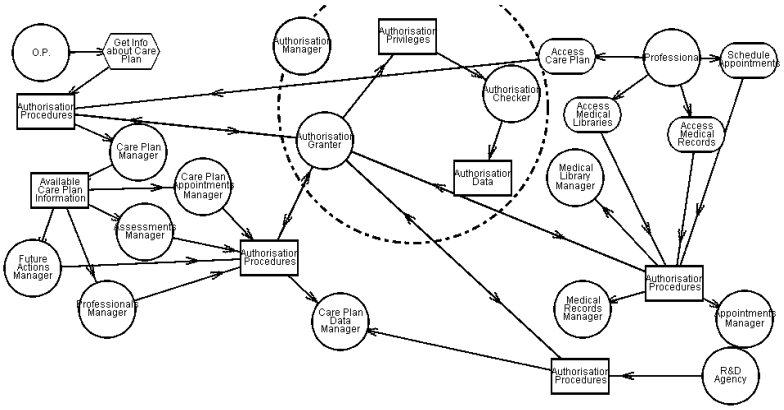

Fig. 7 Extended Diagram with respect to the Authorisation Manager – Internal Decomposition

The second step of the architectural design is the decomposition of actors in sub-actors aiming to expand in details each actor with respect to its goals and tasks. Figure 7 shows the decomposition of the ***Authorisation Manager*** Actor with respect to the *get information about care plan* task. The ***Authorisation Manager*** is decomposed in two sub-actors: the ***Authorisation Granter*** and the ***Authorisation Checker***. The former is responsible for checking the users' authorisation details and grant access to services, while the latter is responsible for checking if the user has access and in what services. The ***Authorisation Granter*** depends on the ***Authorisation Checker*** to obtain the *Authorisation Privileges* of the user. The ***Authorisation Checker*** interacts with the *Authorisation Data* in order to obtain information about the *Authorisation Status* of the user.

The next step of the architectural design is the capabilities identification, in which the capabilities needed by each actor to fulfil their goals and tasks are modelled. The extended actor diagram is used to identify the capabilities, since each dependency relationship can give place to one or more capabilities triggered by external events.

The last step of the architectural design is the agents' assignment. During this step each agent is assigned one or more different capabilities identified in the previous step. Table 1 illustrates the agents along with the capabilities assigned to each one of them with respect to the task *Get Info about Care Plan*, shown in Figure 7.

Table 1 Agent Types and their Capabilities w.r.t. Extended Diagram of Figure 8.

Agent	**Capabilities**
Authorisation Granter	Get User Authorisation Details Get User Authorisation Privileges Allow Access to Services Deny Access to Services
Authorisation Checker	Provide Info about Users' Authorisation Privileges Check User Authorisation Status
Older Person	Provide Information about Older Person Provide Authorisation Details of Older Person Provide Service Description Get Older Person query Act on Behalf of Older Person
Care Planner	Provide service description Provide available Care Plan Information Re-direct User to Appropriate service Manager
Future Actions Agent	Get Older Person query about Future Actions Get Query Results Provide Info about Future Actions
Professionals Agent	Get query about Professionals Get Query results Provide info about Professionals
Previous Assessments Agent	Get Query about Previous Assessments Get Query results Provide Info about Previous Assessments
Appointments Agent	Get Query about Appointments Get query results Provide Info about Appointments
Professional	Act on Behalf of Professional Provide Service Description Provide Info about the Professional Provide Authorisation Details of the Professional Get Professional's query
Services Facilitator	Give access to Medical Libraries Give access to Medical Records Obtain request for Appointment Provide Service Description Get request for Service

3.4 Detailed Design

Detailed design stage aims at specifying agent capabilities and interactions. Thus, during this stage internal and external events that trigger plan and the beliefs involved

in agent reasoning are modelled. In our approach we have adapt a subset of the AUML diagrams proposed in [8]. These are:

– *Capability Diagrams*. We use AUML activity diagrams to model a capability or a set of capabilities for a specific actor. In each capability diagram, the starting state is represented by external events, activity nodes model plans, transition arcs model events, and beliefs are modelled as objects. An example of a capability diagram is shown in Figure 8, in which the *allow (or deny) access to services* capability of the Authorisation Granter Agent is illustrated. The ***Authorisation Granter*** initially accepts an authorisation request from the user. It compares the user's authorisation details with the user authorisation status that exist in the system, and either allow or deny access to services.

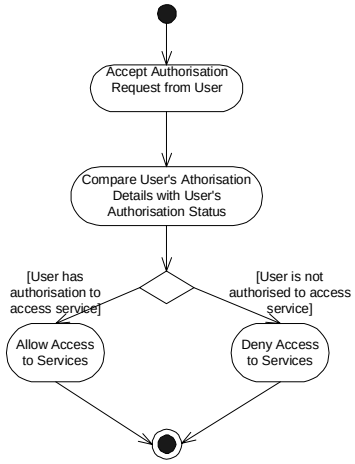

Fig. 8: Capability Diagram for the *Allow or Deny Access* capability

– *Plan Diagrams*. Plan Diagrams are used to further specify each plan node of a capability diagram. Figure 9 illustrates a plan diagram for the *Accept Authorisation Request from User* plan. The ***Older Person*** agent sends an authorisation request to the ***Authorisation Granter***. The ***Authorisation Granter*** checks the authorisation request integrity, by interacting with the ***Integrity Manager***, and if the integrity information is valid the ***Authorisation Granter*** accepts the authorisation request from the user and acknowledges the authorisation request, otherwise it rejects the authorisation request and notify user about the rejection.

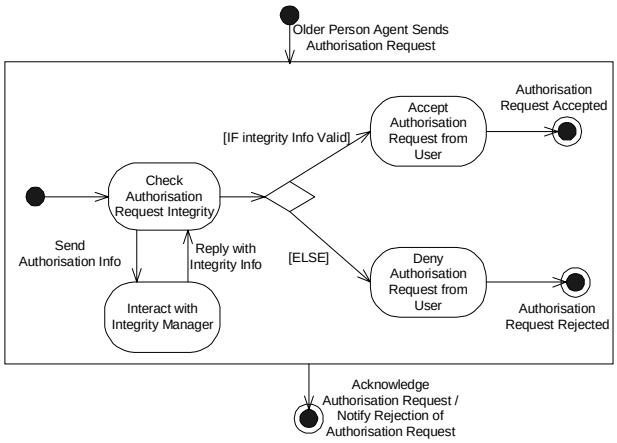

Fig. 9 Plan Diagram for the Accept Authorisation Request from the User

– *Agent Interaction Diagrams*. We apply in our case sequence diagrams modelling agent Interaction Protocols as proposed by [9]. An example of an Agent Interaction Diagram is shown in figure 10. The ***Older Person*** sends an ***Authorisation Request*** to the ***Authorisation Granter*** who acknowledges the request. Then the ***Older Person*** sends the user's authorisation details to the ***Authorisation Granter*** who forwards them to the ***Authorisation Checker***. The ***Authorisation Checker*** checks the user's ***Authorisation Status*** and reply to the ***Authorisation Granter***, who grants or refuses access.

4 Initial Exploration on Extending Tropos to Accommodate Security

For integrated health and social care systems, security is a major concern. In our case study we tried to analyse the system inside its environment taking into consideration some security attributes. We considered security in terms of actor dependences, goals and soft goals, and we assigned capabilities to the agents of the system in order to achieve the security goals and soft goals of the system. Although, the exploration was very encouraging more work is required towards this direction. The need to introduce concepts and notation in the *Tropos* methodology in order to capture some security aspects of the system-to-be is essential. Trying to extend *Tropos* to accommodate security, two main questions must be answered:

1. What extra concepts and notations we need?
2. How we can integrate the process of identifying security in the existing stages of the methodology?

Trying to give answers to the first question, we have identified that the concepts of secure condition, secure dependency, secure goal, secure task and secure resource must be introduced. A secure condition, generally speaking, is a condition that must be satisfied for the system to be secure. A secure dependency means the dependency introduces some security conditions for either the depender or the dependee to satisfy, while a secure goal is a goal that introduces some security conditions to the actors of the system and also to the system itself. However a secure goal does not define how a security condition can be achieved, since (as in the definition of goal) alternatives can be considered. Through a secure task we can identify security conditions but also ways of achieving them. This is possible because (differently than a goal) a task specifies a particular way of achieving something. A secure resource will also introduce some security conditions in the system, related to the resources of the system.

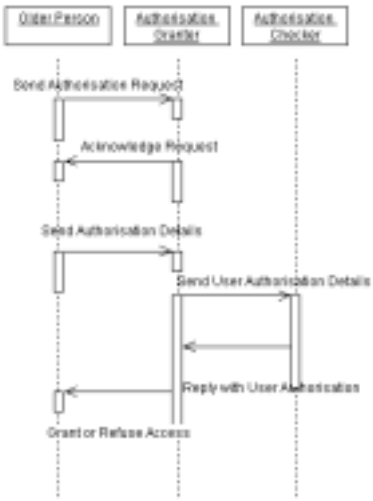

Fig. 10 Agent Interaction Diagram for the Authorisation Procedure

Trying to answer the second question, during the early and late requirements stages, the security conditions derived from the analysis of the stakeholders of the system as well as from the system-to-be can be identified. In the architectural design we identify the security conditions that the new actors introduce and also during the actor decomposition we identify security sub-conditions. Furthermore, along with the capabilities we identify the security conditions that each actor must satisfy in order for the system to be secured. Then along with the agent assignment capabilities we assign the security conditions and sub-conditions that each agent must satisfy. Finally, during the detailed design we specify the agent capabilities and interactions taking into account the security aspects as well. In doing so we are using AUML notation in which we introduce the tag of security rules. This is similar to the business rules that UML has for defining constraints on the diagrams.

Plenty of work is still required so that security can be integrated into *Tropos*. First, the security concepts described in the previous paragraphs must be defined more precisely, and any other necessary concepts and notations must be added. Also, the integration of the security stages together with the *Tropos* stages must be examined more in detail. In addition, formal *Tropos* must be extended to accommodate the new security concepts. Finally, examples of real life complex systems must be used to justify the correctness of the approach.

5 Conclusions

In this paper we applied the *Tropos* methodology in the analysis and design of the electronic Single Assessment Process (eSAP), an electronic system to deliver the integrated health assessment of health and social care needs of older people. The main conclusions derived from this attempt were really encouraging.

Tropos methodology coped well with the challenge of analysing and designing the eSAP system. Although, it was the first time that the first author of this paper applied *Tropos*, the way the methodology is developed (main concepts) and the separation of the stages make it easy-to-use. Also, although it guides developer throughout the analysis and design stages, it also allows freedom for creativity design. Furthermore, starting the analysis from the environment of the system, helps to actually identify easier the main actors involved in the system, and thus the agents (software or humans) involved in it. In addition, it describes the system very well both in a high (environment) and a low (functionality) level.

One of the main difficulties in analysing the eSAP is the fact that there is not a similar system, either computerised or manual, in existence. Because of that, the capture of the requirements and also the roles (of the humans and the agents of the system) is a difficult task. *Tropos* with early requirements analysis, allows us to make explicit the reasons (why) beyond the system requirements and then to decide which is the best solution.

As it was pointed in the previous section, future work will take place so that *Tropos* can also accommodate security concerns of the system-to-be. Another important point for future work is the support of mobile agents. In an integrated health and social care system, the adoption of mobile agents may be an effective choice. Extensions must take place in *Tropos* in order to be able to capture the concept of mobile agents. Trying to give a solution to this problem, questions must be answered as to why/when/where and how an agent moves from one platform to another.

References

1. J. Castro, M. Kolp and J. Mylopoulos. “A Requirements-Driven Development Methodology,” In *Proc. of the 13th Int. Conf. On Advanced Information Systems Engineering (CAiSE’01)*, Interlaken, Switzerland, June 2001.
2. P. Giorgini, A. Perini, J. Mylopoulos, F. Giunchiglia and P. Bresciani. “Agent-Oriented Software Development: A Case Study”. In *Proceedings of*

the 13th International Conference on Software Engineering & Knowledge Engineering (SEKE01), Buenos Aires, Argentina, June 2001.

3. P. Ciancarini and M. Wooldridge, editors. *Agent-Oriented Software Engineering*. Springer-Verlag Lecture Notes in AI Volume 1957, March 2001.
4. J. Mylopoulos and J. Castro. Tropos: A Framework for Requirements-Driven Software Development. In J. Brinkkemper and A. Solvberg (eds.), *Information Systems Engineering: State of the Art and Research Themes*, Lecture Notes in Computer Science, Springer-Verlag, p. 261-273, June 2000.
5. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, J. Mylopoulos. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. To appear in *Proc. of the Fifth International Conference on Autonomous Agents*, Montreal, Canada, 28 May - 1 June 2001.
6. Philp. Can a medical and social assessment be combined?. Journal of the Royal Society of Medicine, 90(32), pp 11-13,1997.
7. E. Yu. Modelling *Strategic Relationships for Process Reengineering*, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.
8. B. Bauer, J. Müller, J. Odell, Agent UML: A Formalism for Specifying Multiagent Interaction. In *Agent-Oriented Software Engineering*, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, pp. 91-103, 2001.
9. J. Odell and C. Bock. Suggested UML extensions for agents. Technical report, OMG, December 1999. Submitted to the OMG's Analysis and Design Task Force in response to the Request for Information enti-tled "UML2.0 RFI".