Secure Tropos: An agent oriented software engineering methodology for the development of health and social care information systems.

Haralambos Mouratidis

h.mouratidis@uel.ac.uk

School of Computing and Technology, University of East London,

4-6 University Way, Docklands, E16 2RD, London, England

Abstract

A huge amount of health and social care related information needs to be stored and analysed; with the aid of computer systems this can be done faster and more efficiently. As a result, computerised health and social care information systems are gaining popularity. The development of such systems, mostly so far, follows an ad-hoc pattern. However, in order to effectively deal with the characteristics of such systems such as size, security, unpredictability, and openness; appropriate software engineering methodologies and paradigms should be employed for their development. This paper defines a set of requirements that a methodology should demonstrate and it argues for the appropriateness of the Secure Tropos agent oriented methodology for the development of health and social care information systems. The applicability of the methodology is demonstrated with the aid of a real-life case study, the electronic Single Assessment Process system, an information system to support integrated assessment of the health and social care needs of older people. The application of the proposed methodology on the case study indicated that the methodology satisfies the identified requirements.

Keywords: Health and social care information systems development, methodologies, software agents, Secure Tropos

1. Introduction

Almost every nation is facing problems (managerial, administrative, and/or clinical) with the delivery of health and social care to its citizens [1]. It has been argued [2] that information technology can provide the answer to some of these problems.

Consider as an example the problems associated with the health and social care of older people such as duplication of assessments, lack of awareness of key concepts of need and fragmentation of care. To overcome these problems national policy in England is the development of the Single Assessment Process (SAP). The Single Assessment Process [3] aims to create closer working for providing primary health and social care for older people and other groups. In a distributed health care setting different health care professionals such as general practitioners and nurses, must cooperate to provide older persons with appropriate care and they must also work closely with social care professionals, such as social workers, because health and social care needs overlap amongst older people. Moreover, a huge amount of data needs to be available to the appropriate professionals to allow them to take the correct decisions.

It has been recognised [3] that computerising this process is a very important step for the success of the single assessment process. Computer systems will help to automate some of the tasks involved, such as the management of the health and social care teams, and the appointments' procedures; as well as allowing to store and analyse important information faster and more efficiently, thus giving health and social care professionals more time for the actual care of the older people.

Because of the advantages introduced by the use of computerised systems, such as the ones described above, there is a tendency to develop computerised information

systems for every area of the health and social care sector, from systems to link General Practitioners' information to systems which provide support to care assistants.

However, such systems demonstrate an inevitable complexity due to their characteristics, such as size (due to the large amount of data and information that needs to be stored as well as the large amount of processes involved), security (privacy of health and social care information has been widely recognised as one of the important issues for the health and social care sector), unpredictability (the health and social care sector is by nature unpredictable due to its dynamic environment), and openness (due to their wide interconnection with other systems, such as police registers). Therefore, system developers must employ software engineering methodologies to assist them during the development of a system by providing, for instance, methods to elicit the system requirements and develop a design that meets these requirements.

Nevertheless, most of the times, an ad-hoc approach is taken for the development of health and social care information systems, partially because current software engineering methodologies are not well suited for the development of such systems. First of all, they fail to deal effectively with the complexity introduced by these systems. Secondly, they fail to provide a clear ontology, which will allow both system developers and users/stakeholders to *actively* contribute to the development of a system. Thirdly, they fail to consider security as an integral part of the development process.

Therefore, it is important to develop methodologies that will overcome the above limitations, and will allow the development of health and social care information systems in an effective and efficient manner. In doing so, it is important to define the requirements that a methodology for health and social care information systems should demonstrate.

This paper defines such requirements and it presents Secure Tropos, an agent oriented methodology, which is an extension of the Tropos methodology [14]. Tropos demonstrates characteristics that make it ideal for the development of health and social care information systems. Firstly, it allows a deeper understanding of not only the system, but also of the environment in which the system would operate. Secondly, Tropos employs the same concepts and notations throughout the development process leading to a homogeneous development that is easier to understand by non-software engineering experts (such as the health and social care professionals). Thirdly, it employs real life concepts and therefore it narrows the gap between the concepts used by the health and social care professionals and the system developers. Moreover, Secure Tropos adds a fourth important characteristic by considering security issues as an integral part of its development process; therefore leading to the development of secure systems, something very important in the case of health and social care information systems.

The paper is structured as follows. Section 2 defines some important requirements that a software engineering methodology should demonstrate to be suitable for the development of health and social care information systems. Section 3 provides an introduction to the agent oriented software engineering paradigm, necessary for the readers to understand the rest of the paper. Section 4 discusses the appropriateness of the Secure Tropos methodology in the development of health and social care information systems, indicating why Secure Tropos is more suitable, than other methodologies, for the development of these systems. Section 5 presents the methodology, whereas Section 6 demonstrates its applicability with the aid of the electronic Single Assessment Process (eSAP) system case study. Finally, section 7

presents related work and section 8 concludes the paper and describes areas of future work.

2. The requirements of a software engineering methodology for the development of health and social care information systems.

During the development of, health and social care or otherwise, information systems, software engineers employ guidelines, notations and follow structure processes to help them go through the development faster and more efficiently. In other words, they employ methodologies. To emphasise the need of a methodology Birrell and Ould argue "*anyone undertaking software development, on no matter what scale, must be strongly advised to establish a methodology for that development –one or more techniques that, by careful integration and control, will bring order and direction to the production process*" [4].

According to Booch [5] "*a methodology is a collection of methods applied across the software development life cycle and unified by some general, philosophical approach*". Thus a methodology is considered to be a pre-defined series of steps that helps designers to understand a problem and model a solution. Although a methodology must guide through its steps, on the same time it must be flexible and allow creativity.

Moreover, there are requirements that methodologies specifically employed for the development of health and social care information systems, should demonstrate. As argued by Norris [1], "the term healthcare is a deceptively simple one embracing multiple, inherently complex, sets of processes and interactions". As a result, health and social care information systems are, usually, large complex systems. Therefore, the development of such systems implies the decomposition of the main system to smaller

interrelated subsystems, each of which is also decomposed to smaller subsystems, until the lowest element of the system is defined. Therefore:

Requirement 1: A*n appropriate methodology should provide concepts and procedures to decompose effectively the system to smaller, easier to understand and develop components*.

Moreover, due to the dynamic and unpredictable nature of the health and social care infrastructures, the relationships between the subsystems might change over time, and new subsystems might often be added into (or removed from) the main system. Therefore:

Requirement 2: A*n appropriate methodology should allow developers to iterate during the development stages and easily modify/add/delete the various components of the system*.

In addition, the description of health and social care systems usually takes place by identifying the relationships between the stakeholders/users and their interaction with the system. However, there is a fundamental difficulty in eliciting these relationships due to the abstraction gap between the concepts used by health and social care professionals to describe the functionalities of the system, and the software engineers to model such functionalities. Moreover, as a result of the difficulty to understand/been explained the concepts involved, health and social care professionals, usually, are not actively involved in the development of the system under development resulting in the development of a system that does not actually meet the needs of its users. Therefore:

Requirement 3: A*n appropriate methodology should provide concepts that are easily understood not only by the system developers but also from the system's stakeholders/users.*

Most of the developers have limited knowledge of the health and social care sector, and as a result, and due to the limited involvement of health and social care professionals, find it difficult to understand the implications of their analysis and designs. Therefore:

Requirement 4: A*n appropriate methodology should allow developers to model not only the system but also the environment in which the system will be deployed as well as the relationships between the different stakeholders/users of the system*.

Additionally, security is an important issue when developing health and social care information systems as such systems contain sensitive information which should be kept private. It has been argued in the literature [6,7] that in order to identify potential conflicts, which could lead to vulnerabilities, between security and the functionalities of the system; security should be considered from the early stages of the system development. Therefore:

Requirement 5: A*n appropriate methodology should allow developers to consider security issues throughout the development stages by providing concepts and processes customised to security.*

Thus, to demonstrate the suitability of the Secure Tropos methodology in the development of health and social care information systems, we must demonstrate the degree at which Secure Tropos meets the above requirements. However, before doing this, it is important to provide a brief overview of agent oriented software engineering, to enable readers with limited knowledge of it, to understand the paradigm in which Secure Tropos is based. The next section provides such an overview.

3. Agent Oriented Software Engineering

As mentioned above, Tropos is based on the agent oriented paradigm. Agent oriented software engineering is based on the concept of an agent[1]. The term agent derives from the present particle of the Latin verb *agere*, which means to drive, act, lead or do [8]. One of the most widely used definition of an agent, by Wooldridge and Jennings [9], defines it as "…. *A hardware or (more usually) software based computer system that enjoys the following properties*:

1. ***Autonomy***. Agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
2. ***Social ability***. Agents interact with other agents (and possibly humans) via some kind of agent communication language.
3. ***Reactivity***. Agents perceive their environment, (which may be the physical world, such as a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it.
4. ***Pro-activeness.*** Agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviour by taking the initiative".

Agent Oriented Software Engineering introduces an alternative approach, than other software engineering approaches, in analysing and designing complex distributed computerised systems [10,11,12]. According to this, a complex computerised system is viewed as a multiagent system [11] in which a collection of autonomous software agents (subsystems) interact with each other in order to satisfy their design objectives.

[1] In this paper the term agent refers always to a software agent.

Therefore, developers view the system as a society, similar to a human society, consisting of entities that possess characteristics similar to humans such as mobility, intelligence and the capability of communicating.

4. The arguments for the suitability of Secure Tropos for the development of health and social care information systems.

It is important, when arguing for the suitability of the Secure Tropos methodology for the development of health and social care information systems, to discuss agent orientation with respect to the requirements defined in Section 2.

According to Jennings and Wooldridge [13] an agent oriented approach is effective in partitioning the problem space of a complex system. A complex computerised system can be decomposed into smaller components, the same way that a multiagent system can be decomposed into the elements that constitute the system (software agents). As mentioned above, when adopting an agent oriented view of a system, the system is viewed as a society similar to a human society, consisting of entities that possess characteristics similar to humans such as mobility, intelligence and the capability of communicating. This allows decomposing the system easier by mapping the real life counterparts of a system to a software subsystem (agent).

Moreover, agent orientation is well suited for the development of dynamic environments, in which the structure of the system changes. Agent structures provide a variety of stable intermediate forms, meaning that individual agents or organisational grouping can be developed in relative isolation and then added into (or removed from) the system in an incremental manner [13].

In addition, agent oriented software engineering provides high-level abstractions, such as agents, (social) interactions and organisations, in modelling complex systems.

This higher (than other software paradigms) level of abstraction employed in the development of software systems results in modelling a system in terms of autonomous entities with characteristics similar to humans. This introduces a close-to-real-life modelling of the system and therefore makes the development of the software system natural. In turn, this helps to narrow the gap between real life and modelling, by allowing developers to reason about the software system using concepts and mental qualities known to them from the real life. This is particularly true in the case of health and social care information systems, where the high level of abstraction, and the usage of concepts close to real life – such as goals, tasks, actors- leads to a better mutual understanding between the system developers (computer scientists) and system users (health and social care professionals) since system developers can better explain the functionalities of the system, by decomposing it to smaller autonomous entities (agents) that posses characteristics similar to humans, such as mobility and the ability to communicate, and the system users can use the concept of an agent to describe more precisely the needs of the system.

Additionally, the agent oriented software engineering paradigm presents a feasible approach for the integration of security to software engineering. This is mainly due to the appropriateness of agent oriented philosophy for dealing with the security issues that exist in a computer system. Security requirements are mainly obtained by analysing the attitude of the organisation towards security and after studying the security policy of the organisation. As mentioned in [13] agents act on behalf of individuals or companies interacting according to an underlying organisation context. The integration of security within this context will require for the rest of the subsystems (agents) to consider the security requirements, when specifying their objectives and interactions therefore

causing the propagation of security requirements to the rest of the subsystem. In addition, the agent oriented view is perhaps the most natural way of characterising security issues in software systems. Characteristics, such as autonomy, intentionality and sociality, provided by the use of agent orientation allow developers first to model the security requirements in high-level, and then incrementally transform these requirements to security mechanisms.

On the other hand, Secure Tropos in particular demonstrates some key features that make it ideal for the development of health and social care information systems. Firstly, it covers the early stages of requirements analysis [14], and thus allows for a deep understanding of not only the system itself, but also of the environment where the system will operate and as a result it helps to better understand the interactions and the relationships that will occur in the system. This is, as discussed above, very important in the development of health and social care information systems since it allows developers to obtain a clear idea of the environment in which the system will be deployed as well as the interactions between the different health and social care professionals involved and as a result develop a system which is customised to that particular environment and meets the necessities of its users.

Secondly, Secure Tropos employs concepts such as stakeholders, actors, goals, tasks, capabilities, and dependencies which are concepts used in real world situations. This results in introducing during the development process concepts familiar to both users/stakeholders and system developers, and therefore minimise the abstraction gap between the concepts used by health and social care professionals and software engineers when describing a system.

Moreover, Secure Tropos uses the same concepts and notations throughout the development stages. This leads to a uniform development that enables system users (health and social care professional in our case) to be involved throughout the development stages and not only during the requirements elicitation stage.

Another important key feature of Secure Tropos is the consideration of security issues during the development of the system. The security requirements of the various health and social care professionals involved with the system are identified with the aid of security related concepts such as security constraints and secure goals. This enables system developers to understand the security requirements of the system by analysing the security related concepts imposed to the various stakeholders/ users of the system. Then, these requirements can be transformed to (security related) functionalities that the system has to satisfy.

To validate our claims, we have applied Secure Tropos to the development of the electronic Single Assessment Process (eSAP) system case study. However, before illustrating this, the next section provides an overview of the concepts and the development stages of the methodology for readers not familiar with it.

5. Description of Secure Tropos

Secure Tropos is an extension of the Tropos methodology. Tropos[2] is a novel agent oriented software engineering methodology tailored to describe both the organisational environment of a multiagent system and the system itself, emphasizing the need to understand not only *what* organisational goals are required, but also *how* and *why* the intended system would meet the organisational goals. This allows for a more refined

[2] The name Tropos derives from the Greek "Τρόπος" which means "way of doing things" but also has the connotation of "easily changeable, easily adaptable".

analysis of the system dependencies, leading to a better treatment not only of the system's functional requirements but also of its non-functional requirements, such as security, reliability, and performance [15].

Tropos supports the idea of building a model of the system that is incrementally refined and extended from a conceptual level to executable artefacts, by means of a sequence of transformational steps [16]. Such transformations allow developers to perform precise inspections of the development process by detailing the higher level notions introduced in the previous stages of the development. In addition, since the methodology employs the same notation throughout the development process, such a refinement process is performed in a more uniform way as compared, for example, to UML-based methodologies where the graphical notation changes from one development step to another (for example, from use cases to class diagrams). Moreover, Tropos allows developers to consider security issues as an integral part of the system development. The security process is one of analysing the security needs of the stakeholders and the system in terms of security constraints imposed to the system and the stakeholders; identify secure entities that guarantee the satisfaction of the security constraints and assign capabilities to the system to help towards the satisfaction of the secure entities. In Secure Tropos, a security constraint is defined as *a restriction related to security issues, such as privacy, integrity and availability, which can influence the analysis and design of a multiagent system under development by restricting some alternative design solutions, by conflicting with some of the requirements of the system, or by refining some of the system's objectives* [17].

A *security constraint* contributes to a higher level of abstraction, meaning that security constraints do not represent specific security protocol restrictions[3], which restrict the design with the use of a particular implementation language. This higher level of abstraction allows for a generalised design free of models biased to particular implementation languages. Regarding the constraint metamodel, a security constraint is captured through a specialisation of constraint into the subclass security constraint.

In addition, *Tropos* adopts the *i** modelling framework [18], which uses the concepts of actors, goals and social dependencies for defining the obligations of actors (dependees) to other actors (dependers), and the novelty of the methodology lays on the fact that those concepts are used to model not just early requirements, but also late requirements as well as architectural and detailed design [19]. Using the same concepts during the development stages of a multiagent system provides the advantage of reducing impedance mismatches between different development stages, and therefore streamlines the development process [19].

The system and its environment are viewed as a set of actors, who depend on other actors to help them fulfil their goals. An actor [18] represents an entity that has intentionality and strategic goals within the multiagent system or within its organisational setting. An actor can be a (social) agent, a position, or a role. Agents can be physical agents, such as a person, or software agents.

A goal [18] represents a condition in the world that an actor would like to achieve. In Tropos, the concept of a hard-goal (simply goal hereafter) is differentiated from the concept of soft-goal. A soft-goal is used to capture non-functional requirements of the

[3] Such security restrictions should be specified during the implementation of the system and not during the analysis and design.

system, and unlike a (hard) goal, it does not have clear criteria for deciding whether it is satisfied or not and therefore it is subject to interpretation [18]. For instance, an example of a soft-goal is "the system should be scalable". On the other hand, a secure goal represents the strategic interests of an actor with respect to security. Secure goals are mainly introduced in order to achieve possible security constraints that are imposed to an actor or exist in the system. However, a secure goal does not particularly define how the security constraints can be achieved, since alternatives can be considered.

A task represents, at an abstract level, a way of doing something [15]. The fulfilment of a task can be a means for satisfying a goal, or for contributing towards the satisficing of a soft-goal. Secure Tropos allows reasoning about the different ways the actors can achieve their goals, by enabling developers to model alternative tasks that actors might employ to achieve their goals. Similarly, a secure task is used to define precisely how a secure goal can be achieved, by representing a particular way of satisfying a secure goal.

A resource [15] presents a physical or informational entity that one of the actors requires. The main concern when dealing with resources is whether the resource is available and who is responsible for its delivery. On the other hand, a secure resource can be defined as an informational entity that is related to the security of the information system. Secure resources can be divided into two main categories. The first category contains secure resources that display some security characteristics, imposed by other entities, such as security constraints, secure goals, secure tasks and secure dependencies. The second category of secure resources involves resources directly associated with the security of the system. For example, consider the authorisation details file of a component of the system.

A dependency [18] between two actors represents that one actor depends on the other to attain some goal, execute a task, or deliver a resource. The depending actor is called the depender and the actor who is depended upon is called the dependee. The type of the dependency – goal, task, resource- describes the nature of an agreement (called dependum) between dependee and depender. By depending on the dependee for the dependum, the depender is able to achieve goals that it is otherwise unable to achieve on their own, or not as easily or not as well [18]. Moreover, a secure dependency [17] introduces security constraint(s) that must be fulfilled for the dependency to be satisfied. Both the depender and the dependee must agree for the fulfilment of the security constraint in order for the secure dependency to be valid. That means the depender expects from the dependee to satisfy the security constraint(s) and also that the dependee will make an effort to deliver the dependum by satisfying the security constraint(s). There are three different types of a secure dependency: Dependee Secure Dependency, in which the depender depends on the dependee and the dependee introduces security constraint(s) for the dependency. The depender must satisfy the security constraints introduced by the dependee in order to help in the achievement of the secure dependency. Depender Secure Dependency, in which the depender depends on the dependee and the depender introduces security constraint(s) for the dependency. The dependee must satisfy the security constraints introduced by the depender otherwise the security of the dependency will be in risk. Double Secure Dependency, in which the depender depends on the dependee and both the depender and the dependee introduce security constraints for the dependency. Both must satisfy the security constraints introduced to achieve the secure dependency.

A capability [15] represents the ability of an actor of defining, choosing and executing a task for the fulfilment of a goal, given certain world conditions and in presence of a specific event. A secure capability represents the ability of an actor/agent to achieve a secure goal, carry out a secure task and/or deliver a secure resource.

A graphical representation of the Tropos concepts is illustrated in Figure 1.

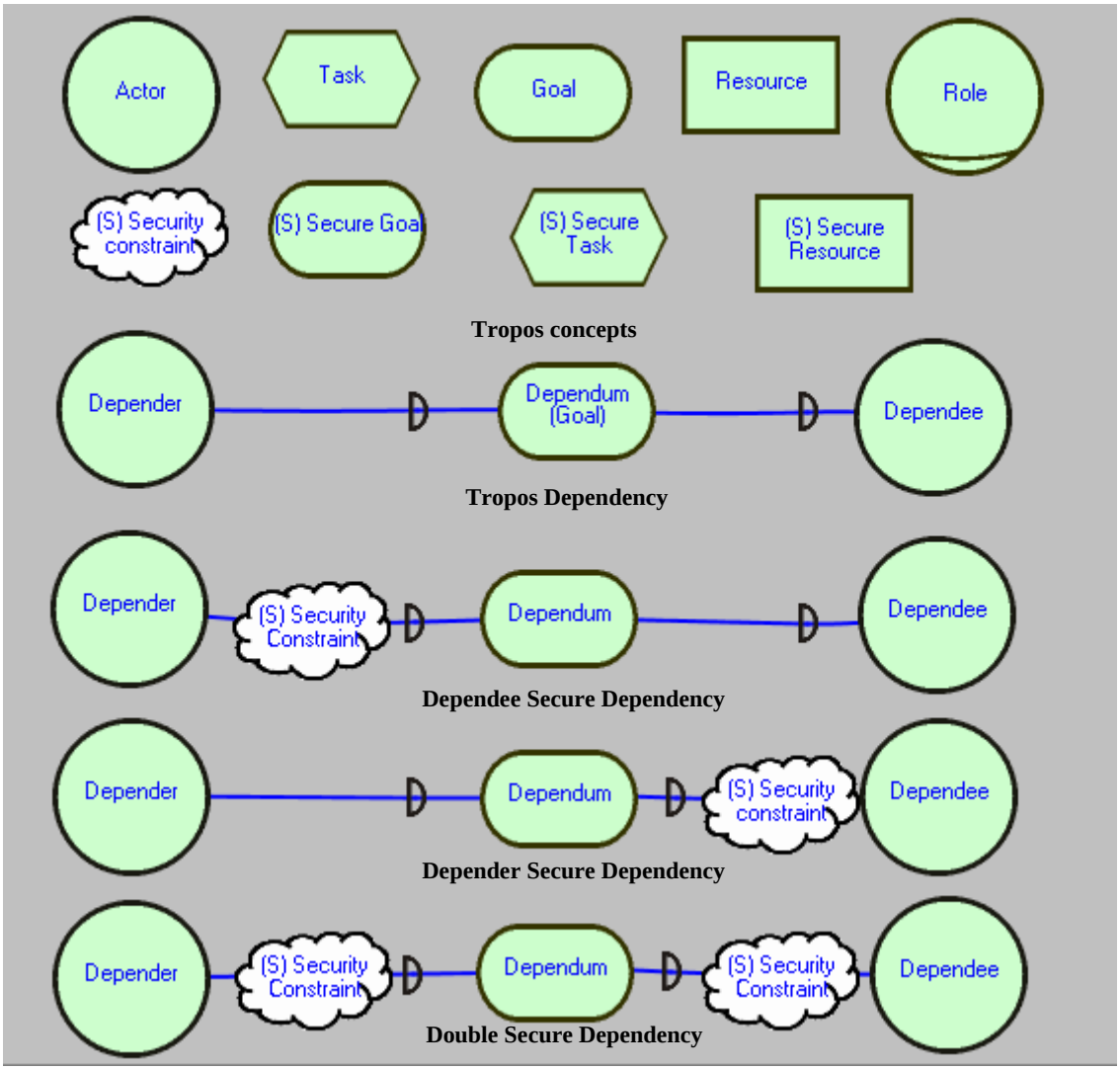

Figure 1: Tropos concepts graphically

Tropos defines its own modelling language [14] in terms of a UML metamodel. The Tropos metamodel is organised into four levels. The meta-metamodel level, which

provides the basis for metamodel extensions; the metamodel level, which provides constructs for modelling knowledge level entities and concepts; the domain level, which contains a representation of entities and concepts of a specific application domain; and the instance level, which contains instances of the domain level. For instance, consider an entity as an example of the meta-metamodel, an actor as an example of the metamodel level, a doctor as an example of the domain level and John as an example of the instance level. In addition, the meta-metamodel level of the language allows the inclusion of constructs for the formal definition of the Tropos concepts. In particular a formal specification language, called Formal Tropos, is under development [20]. Formal Tropos [20,21] offers all the concepts of graphical Tropos, such as actors, goals and dependencies, supplemented with a rich temporal specification language, inspired by KAOS [22].

The Secure Tropos methodology covers four main software development stages. During the early requirements analysis stage, developers are concerned with the understanding of a problem by studying an existing organisational setting. This involves the identification of the domain stakeholders and their modelling as social actors. In particular, developers model the stakeholders as actors, their intentions as goals, and their relationships as dependencies. Through a goal-oriented analysis [16], the actors' goals are decomposed into more precise goals and sometimes into tasks that if performed by the actor, allow for goal achievement. From the security point of view, security constraints are imposed to the stakeholders of the system (by other stakeholders). Such constraints are initially expressed in high level statements, and they are later further analysed and secure goals and entities are introduced to corresponding

actors to satisfy them. The output of this phase is an organisational model, which includes relevant actors and their respective dependencies and security entities.

In the late requirements analysis stage, the system-to-be is specified within its operational environment, together with relevant functions and qualities. This description models the system as an actor, who has a number of dependencies with the actors identified during the previous stage. These dependencies indicate the obligations of the system towards its environment, and therefore define the system's functional and non-functional requirements. Moreover, security constraints are delegated to the system-to-be and appropriate entities are assigned to satisfy them.

During the architectural design stage, the system's global architecture is defined in terms of subsystems, interconnected through data and dependencies. In particular, subsystems are represented as actors and data/control interconnections are represented as (system) actor dependencies. This stage is divided into three steps. The first step includes the definition of the overall architectural organisation by introducing new actors to the system and delegating to them some of the goals of the system. The second step includes the identification of the capabilities needed by the actors to fulfil their goals and tasks and the third step involves the identification of a set of agent types and the assignment of capabilities to those agents. From the security point of view, any possible security constraints and secure entities that new actors might introduce are analysed. Additionally, the architectural style of the multiagent system is defined with respect to the system's security requirements and the requirements are transformed into a design with the aid of security patterns. The final output of this stage is a set of software agents corresponding to the actors of the system, each characterised by its specific capabilities.

In the detailed design stage, each architectural component is defined in further detail in terms of inputs, outputs, control, security and other relevant information. This stage is based on the specifications resulting from the analysis of the previous stages and therefore the reasons for a given element at this stage can be traced back to the early requirements analysis. For this stage, Secure Tropos uses elements of the Agent Unified Modeling Language (AUML) [23] to complement the features of i*.

It must be noted that Tropos is not a "laboratory" methodology but it has been motivated by a number of case studies [14,19].

6. Case Study

The assessment of health and social care needs is at the heart of good practice in the care of older people. Older people often have multiple impairments and health problems, and complex support systems involving several health and social care practitioners and family carers. Sharing of assessment information is important to avoid unnecessary repetition and to ensure that all relevant information is available to support effective care planning. Recognition of the need to share assessment information has stimulated standardisation of assessment methods. These in turn have been used to help standardise care planning and referrals following assessment.

In March 2001, the (English) Department of Health published its National Service Framework (NSF) for Older People's Services [3]. The NSF sets national standards for the health and social care of older people.

Standard 2 of the National Service Framework, which refers to person-centred care, includes requirements to establish a single assessment process for integrating the assessment of health and social care needs of older people. Local health and social care communities have to introduce standardised shared systems for assessing needs, with

convergence towards a fully integrated and electronically based national system. The Department issued further guidance in February 2002, listing requirements for contact, overview, specialist and comprehensive assessment, and a range of assessment instruments, which could be used for these types of assessment.

Information technology has the potential to improve efficiency and effectiveness in the collection and sharing of assessment information. An information system, called hereafter, the electronic single assessment process (***eSAP***) system, to support integrated assessment of the health and social care needs of the older person, should therefore build on contact and overview assessment in primary care, with maximum involvement of the older person in prioritising the assessment domains and in care planning.

6.1 A TYPICAL SCENARIO

Our aim in this paper is not to provide a complete analysis and design of the electronic single assessment process (eSAP) system, but rather to illustrate how the Secure Tropos methodology can be effectively used in the development of a health and social care information system. As a result, we focus our illustration to a specific scenario related to the functionality of the electronic single assessment process. This allows us to define well the boundaries of the system, and as a result it makes the understanding of the methodology easier for the readers, since only important issues of the methodology are presented without going into unnecessary details.

The following scenario has been used in this paper for the analysis and design of the electronic single assessment process.

"*An 81 years old lady, widow, lives in her house. Her daughter lives nearby but she has children of her own and therefore she is unable to provide full care to her mother. However, she sees her mother everyday.*

The daughter visits the mother's General Practitioner (GP) to describe her concern about her mother's health. Her mother has become unsteady on her feet and may have had a number of falls. Single assessment process has been introduced, so the GP asks the daughter to complete the EasyCare [24] *contact assessment and the information is entered into the GP's computer. The GP sees the daughter concerned about her mother's health and asks his practice nurse to visit the old lady to perform an overview assessment.*

The old lady's information is transferred to the nurse's computer along with referrals and instructions, e.g. the daughter of the patient is concerned about her mother's health so please perform an overview assessment. The nurse receives the information and arranges to visit the old lady by generating and sending a letter to the old lady and her daughter giving details about visit and ask availability. The daughter replies (also provides her mother's response) that the date/time is suitable.

The nurse visits the old lady and completes most of the EasyCare assessment except from the health promotion module. From the evaluation of the other modules the nurse concludes the old lady has a number of problems with her house, which increase the risk of falls, she needs help with dressing and also she is not getting the appropriate financial benefits. Then the nurse asks the old lady if the information can be shared, and the old lady accepts. The nurse then produces a care plan summarising all the problems identified and the actions to be taken. She also makes two referrals one to a Social Worker (SW) - to check for a care assistant to help the old lady with dressing and to check about financial benefits- and a second to an Occupational Therapist (OT) -to perform a house assessment for need and adaptation. She then forwards the care plan and a summary of the problems to the General Practitioner and the care plan and

contact information to the Social Worker and the Occupational Therapist. In addition, a copy is produced for both the old lady and her daughter and the care plan is signed.

Later, the old lady is visited by the Occupational Therapist who performs the house assessment and decides that the house needs to be adapted to the old lady's needs. The O.T. then makes a referral to the Equipment Services for equipment and also provides the contact information of the old lady. In addition, the O.T. forwards to the GP, nurse and the S.W. a copy of the house problems, needed equipment and informs them that a referral has been made to the Equipment Services.

The Social Worker visits the old lady and identifies that the old lady must apply for financial benefits. A form is produced, filled in, and sent to the Benefits Agency together with old lady's contact and bank information. In addition, the Social Worker agrees to employ a Care Assistant (C.A.) to help the old lady with dressing. A Care Assistant is identified and the Social Worker asks the old lady if she feels comfortable with the identified Care Assistant and the old lady agrees. Contact and overview assessment information is sent to the Care Assistant by the Social Worker. Also, because of the employment of a Care Assistant, the benefits must be adjusted. The social worker informs the benefits agency about it.

While the Care Assistant visits the old lady, she realises that the health promotion module of the overview assessment is not completed. She notifies the nurse and prompts the old lady to fill in the module. When the module is complete, the C.A. sends the information to the General Practitioner, the nurse and the Social Worker.

One of the observations of the Care Assistant was that the old lady didn't have her blood pressure taken for the last 5 years, so she alerts the nurse and the G.P. The General Practitioner receives the alert and makes a referral to the nurse to go and

check the blood pressure of the old lady. The nurse visits the old lady to review all the actions of the care plan and also check the old lady's blood pressure. The care plan is updated and for the time being the old lady gets everything she needs."

6.2 DEVELOPING THE ESAP

The above scenario provides the basis for the development of the system. As mentioned in the previous section the first phase of the Tropos methodology is the early requirements analysis.

6.2.1 Early requirements analysis

During the early requirements analysis the software engineer models the dependencies between the stakeholders (actors). For this reason, the Secure Tropos methodology introduces actor diagrams [14]. In such a diagram, actors (graphically represented as circles[4]) are modelled together with their goals (represented as ovals), soft-goals (represented as bubbles), security constraints (represented as clouds) and their dependencies (represented as links between the actors indicating the dependum). From the scenario presented above, the following actors are derived:

Older Person: The Older Person actor represents patients aged 65 or above, assessed for their health and social care needs. In the presented scenario, the old lady plays this actor. The Older Person must provide information about their health and social care situation, and also receive information such as a summary of their needs and a copy of their care plan. To provide information, the Older Person must undertake assessments. This requires the Older Person to agree with the health and social care professionals on the date/time that the assessments will take place. In addition, the Older Person must understand the procedures clearly, have access to information

[4] For a reminder of the graphical representation of the Tropos concepts please refer to Figure 1.

regarding their care 24 hours every day, and also decide if their information will be shared between the professionals' (and possibly other people) involved in their care. Also, the Older Person must follow the care plan indicated by the health and social care professionals. Therefore, the help of carers (informal-like the daughter- and paid-like the care assistant-) is required.

Nurse: The Nurse performs the overview assessment to the Older Person. To do this, the Nurse must contact the Older Person and arrange a meeting. After performing the assessment the Nurse identifies the care needs of the Older Person, and according to those needs she provides referrals. Also the Nurse must ask for the older person's consent in order to share information with others who may be involved in the care of the Older Person. She generates a care plan and produces a copy of it for the Older Person. In addition, the Nurse informs everyone involved (taking into account the consent of the Older Person) about the care plan and the condition of the Older Person. The Nurse is also responsible for regular review of the care plan.

General Practitioner: The General Practitioner performs the contact assessment, provides referrals to the Nurse to perform an overview (or any different kind she/he thinks is appropriate) assessment, and provides the older person's contact information. In addition, the General Practitioner receives alerts and information regarding the Older Person, such as the care plan, possible referrals, and updates of the care plan.

Social Worker: The Social Worker receives referrals (indicating the problems occurred) and the actions to be performed, and also information about the Older Person such as contact information and a copy of the care plan. According to the referrals, the Social Worker identifies the needs of the Older Person and takes actions. The Social Worker is usually responsible for identifying a suitable care

assistant (if necessary) and also dealing with benefits problems that the Older Person might have. After identifying particular problems the Social Worker provides referrals, informs the other professionals involved in the care of the Older Person and updates the care plan. In addition, the Social Worker manages the care assistant.

Secondary Care Professional: Secondary care professionals (or specialists) undertake assessment and care following referral by primary care professionals. Some secondary care professionals such as community psychiatric nurses work at the interface between primary and secondary care. During the single assessment process, secondary care professionals, usually, do specialist and comprehensive assessments. In the presented scenario, the Occupational Therapist plays this role. The Occupational Therapist receives referrals from the Nurse along with the contact and overview assessment information of the Older Person. She performs a specialist assessment and identifies specialist needs of the Older Person. According to the identified needs, the Occupational Therapist provides referrals, informs the other professionals involved in the care of the Older Person and also updates the care plan.

Care Assistant: The main aim of the Care Assistant is to help the Older Person with everyday needs. The Care Assistant receives information about the Older Person, such as contact and overview assessment, and updates any of those if necessary by providing to the Nurse possible needs of the Older Person. In addition, she informs the General Practitioner, the Social Worker and the Nurse of any updates regarding the older person's information.

Informal Carer: Informal carers include unpaid family members, friends, and neighbours who help meet older persons' needs for care and support, including meeting emotional (visiting and support), financial (help with managing bills), domestic (help

with shopping) and personal (help with dressing) care needs. In the presented scenario the daughter of the old lady plays this role.

Care Manager: A Care Manager, usually a Social Worker or a Nurse, coordinates the delivery of care to the Older Person and plans the work of the care assistants. In the presented scenario, the Social Worker plays the Care Manager.

Benefits Agency: The Benefits Agency actor represents a financial agency that helps older persons financially.

The dependencies, between the above actors are modelled as shown in Figure 2.

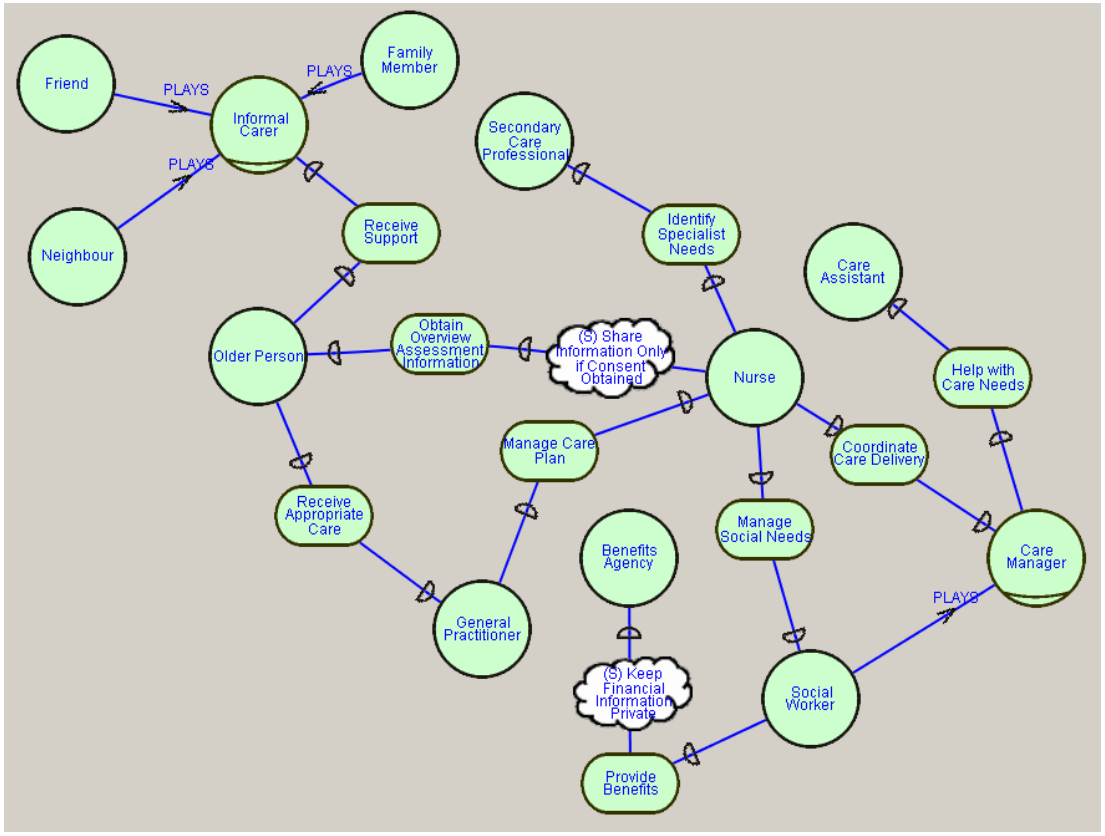

Figure 2: Partial Actor diagram for the eSAP

For instance, the Older Person depends on the General Practitioner to Receive Appropriate Care and on the Informal Carer to Receive Support. On the other hand, the Nurse depends on the Secondary Care Professional to Identify Specialist Needs, on the Care Manager to Coordinate Care Delivery, on the Social Worker

to Identify Social Needs and on the Older Person to Obtain Overview Assessment Information.

However, one of the most important and delicate matters for the Older Person is the privacy of their personal medical information and the sharing of it. Therefore, the Older Person imposes a security constraint (share information only if consent Obtained) on the Nurse for the Obtain Overview Assessment Information dependency to be valid. In addition, the Social Worker imposes a security constraint (Keep Financial Information Private) on the Benefits Agency for the Provide Benefits dependency to be valid.

Modelling the dependencies and the security constraints of the individual actors allows developers to model the functional and security requirements of the system according to the real needs of its stakeholders. For example, in the presented analysis, the lack of identifying the security constraints between the Nurse and the Older Person, or the Social Worker and the Benefits Agency would result in a design that would miss important information regarding the security of the system.

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. For this reason, Tropos employs goal diagrams. In a goal diagram, each actor is represented as a dashed-line balloon within which the actor's goals, security constraints and dependencies are analysed. The nodes of the diagram represent goals, soft-goals, tasks, security constraints, and secure entities whereas the links identify the different kinds of relationships between those nodes. Moreover, these links can be connected with external dependencies (identified in the actor diagram) when the reasoning of the analysis goes beyond the

actor's boundary [18]. Means-end, contribution and decomposition analysis is used in goal diagrams. Means-end analysis is a form of analysis, consisting of identifying goals, tasks, and/or resources that can provide means for reaching a goal [14]. Contribution analysis is a form of analysis that discovers goals and tasks that can contribute positively or negatively to the achievement of another goal. It is worth mentioning that contributions could be positive or negative [14]. These are graphically represented with plus (+) and minus (-) signs respectively. When a sign is not used, it is assumed the contribution is positive. Decomposition analysis defines the decomposition of a root goal/ task into sub goals /tasks.

As an example of a goal diagram, consider the Nurse actor shown in Figure 3. The main goal of the Nurse[5] is to Manage the Care Plan. To satisfy this goal the Nurse must Generate the Care Plan, Review the Care Plan and Provide Information. However, the Share Information only if Consent Obtained security constraint imposed to the Nurse by the Older Person (see Figure 2) restricts the Share Older Person Information goal of the Nurse. For the Nurse to satisfy this constraint, a secure goal is introduced Obtain Older Person Consent.

Furthermore, the analysis indicates that the Use of eSAP will enable the Nurse actor to work more efficiently, with less effort, convenient and faster. However, an analysis [17] regarding the security of the eSAP system indicates that Authorisation is required for the eSAP system (in order to help towards the Privacy security feature).

[5] To keep the complexity of the figure as minimum as possible, an asterisk * has been used to indicate that the same actor, goal, or entity has been modelled more than once in the figure.

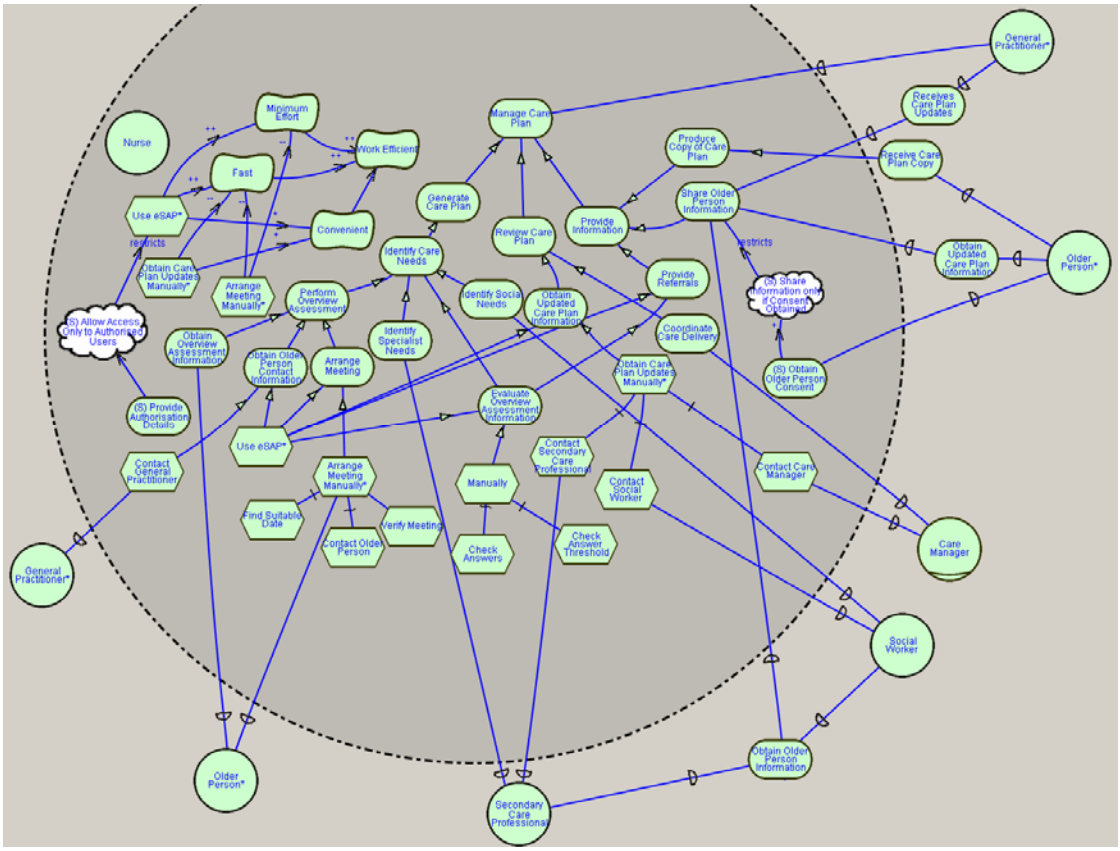

Figure 3: Partial Goal Diagram for the Nurse Actor

Therefore, the security constraint Allow Access Only to Authorised Users, which restricts the Use eSAP task, is imposed to the Nurse actor. To help towards the satisfaction of the imposed security constraint the secure goal Provide Authorisation Details is introduced to the Nurse as indicated in Figure 3.

Modelling the actors internally leads to a more precise definition of the why of the system functionality, and this subsequently helps to verify how the final system implementation matches the stakeholders' real needs.

When all the actors have been further analysed, the actor diagram is refined, as shown in Figure 4, and any possible new dependencies identified during the internal actors' analysis are modelled.

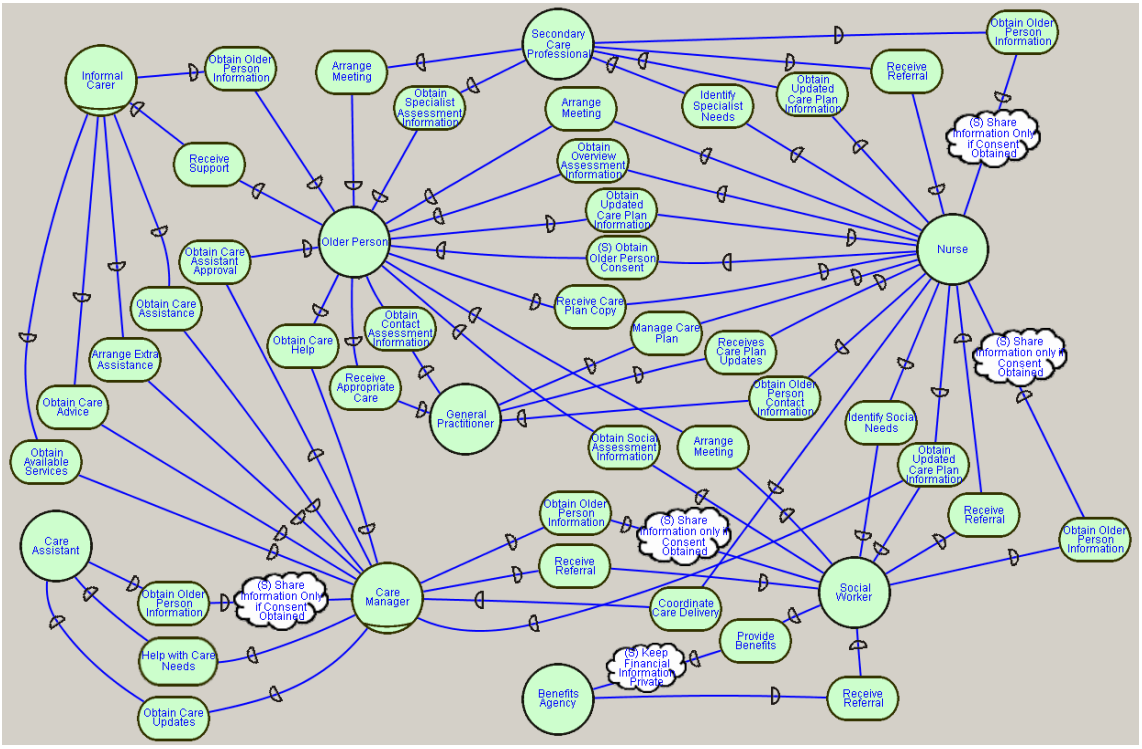

Figure 4: Refined Actor Diagram

This is important since during the actors' internal analysis it is possible that new goals are discovered, which the actors might not be able to satisfy by themselves. Therefore, new dependencies are introduced to enable an actor to delegate to another actor the goals that cannot accomplish on their own. Introducing new dependencies between the actors according to the goals/tasks derived from the internal analysis that takes place in each actor is a common task in Secure Tropos and it is very important since it helps to identify clearly the relationships between the actors and also indicates how the analysis of the goals of one actor can influence the dependencies between this actor and the other actors. Refining the dependencies and the social relationships of the actors this way, leads to a more precise definition of the why of the system functionalities, and as a last result, helps to verify how the final implementation matches the real needs [17]. Also, with regards to security the refinement of the actor diagram is equally important. Refining an actor's goals and dependencies could result in the

redefinition of the security constraints imposed to particular dependencies or the addition of new security constraints. As an example, consider the security constraint Share Information Only if Consent Obtained. This security constraint was imposed to the Nurse, as shown in Figure 2, by the Older Person as part of the Obtain Overview Assessment Information dependency. However, the internal analysis of the Nurse indicated that this security constraint restricts in fact the Share Older Person Information goal of the Nurse. Therefore, in the refined actor diagram, the security constraint has been imposed to all the newly discovered (after the internal analysis of the actors) dependencies that involve the Share Older Person Information goal.

One of the most important issues in the development of the electronic Single Assessment Process system, as in any health and social care information system, is to precisely identify the environment that the system will be situated. This means to identify the intentions of the users, anyone related to the system, and the relationships between those. This is necessary in order to identify the functional and non-functional requirements of the system-to-be. Secure Tropos helps towards this direction during the early requirements analysis stage, since the final output of the early requirements analysis is a clearly defined organisational model, which includes relevant actors (users), their intentions and their respective dependencies (relationships).

6.2.2 Late requirements analysis

During the late requirements analysis the system-to-be, in our case study the electronic Single Assessment Process system, 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 organisation. These

dependencies define all functional and non-functional requirements for the system-to-be.

During the Nurse analysis, modelled in Figure 3, it was identified that the Nurse can achieve some of the goals either manually or by using the electronic single assessment process (eSAP) system. Consider for example, the Arrange Meeting goal of the Nurse actor. This can be fulfilled either by the task Use eSAP or by the task Arrange Meeting Manually. However, the analysis, presented in Figure 3, showed that by using an information system, the eSAP system, the Nurse will be able to work more efficiently, with less effort, faster and more conveniently than trying to achieve the task manually. Similar conclusions were drawn for all the actors of the system [14]. Therefore, it was decided that the use of eSAP provides advantages over the manual achievement of most of the actors' tasks, and as a result the responsibility for the achievement of those tasks was delegated to the eSAP system.

Therefore, in our analysis, the eSAP system has been introduced as another actor that receives the responsibility for the fulfilment of some of the goals identified during the early requirements analysis for the actors of the system. In other words, some goals that the actors of the system cannot fulfil or are better fulfilled by the eSAP system are delegated to the eSAP System.

The actor diagram including the eSAP system and the refined dependencies[6] is shown in Figure 5. For instance, it was identified during the early requirements analysis

[6] It is worth mentioning that the dependencies of the Informal Carer actor are not delegated to the eSAP system, since it is assumed at this point that the Informal Carer does not interact with the system.

that the Arrange Meetings and Obtain Updated Care Plan Information goals of the Older Person will be achieved better with the aid of the eSAP system.

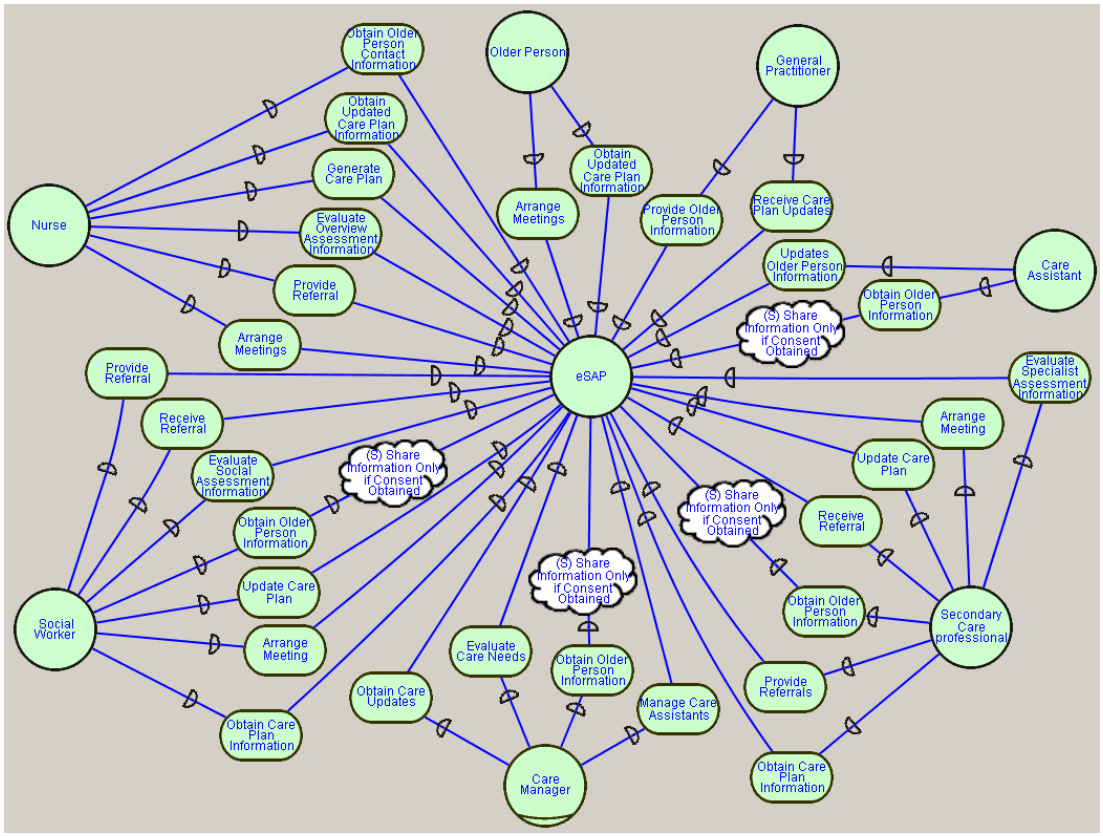

Figure 5: Actor diagram including the eSAP actor

Therefore to indicate this, two goal dependencies have been added from the Older Person to the eSAP system actors. It is worth mentioning that along with dependencies, security constraints are also delegated. For example, before the introduction of the eSAP system, the Social Worker was depending on the Nurse to Obtain Older Person Information. However, this secure dependency involves the security constraint (restricting the Nurse) Share Information Only if Consent Obtained. With the introduction of the eSAP system, the Social Worker actor depends on the eSAP for the satisfaction of the Obtain Older Person Information secure dependency. Therefore, the satisfaction for this dependency is delegated from the Social Worker to the eSAP actor. However, this secure dependency imposes the

satisfaction of the Share Information Only if Consent Obtained security constraint, and as a result the eSAP becomes responsible for satisfying it.

To satisfy all the delegated dependencies, the main goal of the eSAP system has been identified as to Automate Care. By performing a means-end analysis, presented in Figure 6, it was identified that for the eSAP System to fulfil the Automate Care goal, the following sub-goals must be accomplished: Assist with Assessment Procedures, Provide Older Person Information, Manage Care Plans and Schedule Meetings. Each of these sub-goals can be furthered analysed by employing means-end analysis. For example, the Manage Care Plans goal can be accomplished with the fulfilment of the Generate Care Plan, Manage Care Plan Updates, Provide Care Plan Information, Manage Referrals and Identify Care Assistants sub-goals.

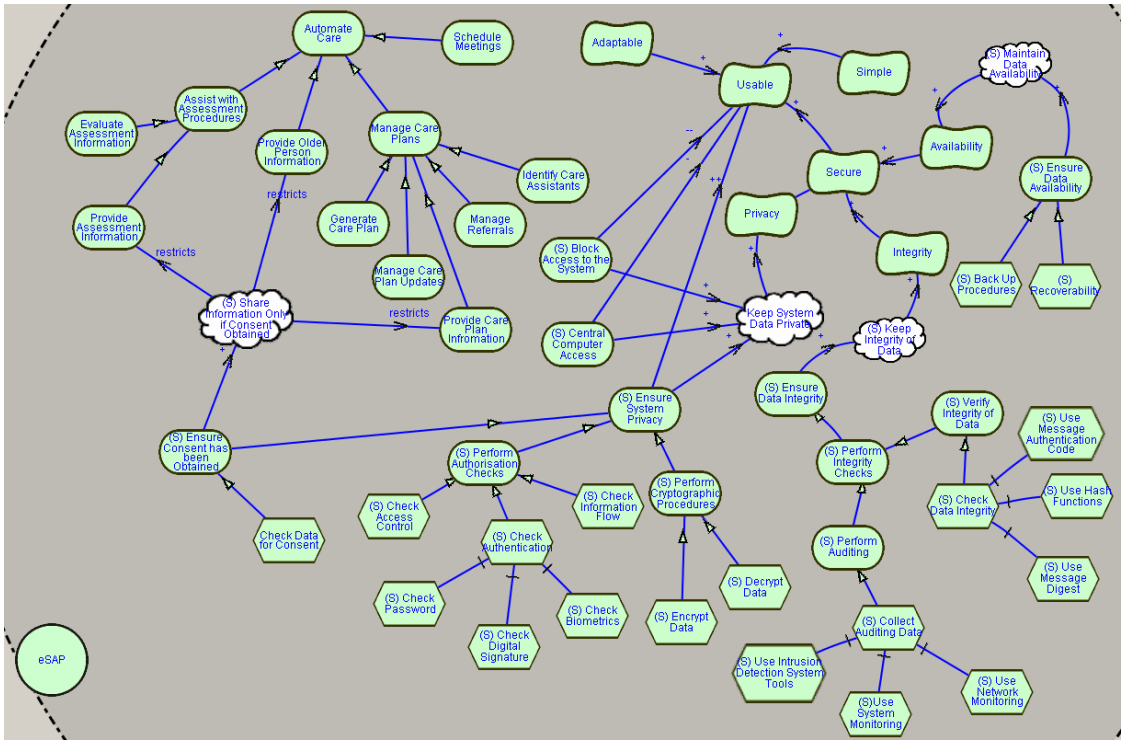

Figure 6: Goal Diagram for the eSAP actor

From the security point of view, three main constraints imposed, by the desired security features of the system, Privacy, Integrity and Availability, to the eSAP's main goal. These are Keep System Data Private, Keep Integrity of the Data and Maintain Data Availability. In addition, the eSAP system must satisfy the Share Information Only if Consent Obtained security constraint imposed to the eSAP by the secure dependencies delegated by the other actors. Each of these secure constraints can be satisfied with the aid of one or more secure goals. For example, the Keep System Data Private security constraint can be fulfilled by blocking access to the system, by allowing access only from a central computer, or by ensuring system privacy. However, the first two contribute negatively to the usability of the system, i.e. the system will be secure but it will not be used. On the other hand, the Ensure System Privacy secure goal is considered the best solution since it provides security to the system and it doesn't affect (dramatically) its usability.

Thus, for the eSAP to satisfy its security constraints the following secure goals have been identified as shown in Figure 6: Ensure System Privacy, Ensure Data Integrity, Ensure Data Availability and Ensure Consent has been Obtained. These can be furthered analysed. For example, the Ensure System Privacy goal is further analysed into the Perform Authorisation Checks and Perform Cryptographic Procedures secure goals. Both of those goals must be fulfilled for the Ensure System Privacy goal to be satisfied.

An important issue at this point is to check whether the goals assigned to the eSAP system satisfy all the goals delegated to the system by the other actors. Thirty (30) goals were delegated to the eSAP system as shown in Figure 5. From these goals, fifteen of them are satisfied by the Manage Care Plans goal (and its sub-goals), six of them are

satisfied by the Provide Older Person Information goal, five are satisfied by the Assist with Assessment Procedures goal (and its sub-goals), and four of them are satisfied by the Schedule Meetings goal.

As it can be seen from the analysis presented in this section, the late requirements analysis stage follows the same analysis techniques used in the early requirements analysis. The main difference is the idea of introducing the system as another actor. Such an approach is very important and provides advantages since it helps to identify clearly the relationships and the dependencies between the system and the environment that the system will be situated. Medical Information Systems, such as the electronic Single Assessment Process, more often are introduced to environments in which non or very little computer expertise is found. Defining clearly the roles and the dependencies of the actors and the system helps to identify the functional and non-functional requirements of the system-to-be according to the real needs of the stakeholders. Also, by analysing the system within its operational environment helps to delegate responsibility for the achievement of goals to the system (by other stakeholders) and also identify new dependencies between the system and its stakeholders. This leads to the definition of functional and non-functional requirements for the system, which would be very difficult to identify otherwise. In addition, the way the system is analysed within the late requirements stage, provides developers with the ability to consider different alternatives for satisfying the system's goals and decide, by checking for example if the alternative contributes positively or negatively to other goals of the system, which of those alternatives is the best solution.

6.2.3 Architectural design

When the system goals have been identified, the next step of the development cycle involves the definition of the system's global architecture in terms of subsystems (actors) interconnected through data and control flows (dependencies). To achieve this, Secure Tropos employs different processes during the architectural design stage, such as the *Addition of new actors*, in which new actors (and sub-actors) are added to make the system interact with the external actors, to take responsibility to fulfil one or more goals of the system and to contribute positively to the fulfilment of some non-functional requirements; the *Capabilities identification*, in which the capabilities needed by the actors to fulfil their goals and tasks are identified; and the *Agents assignment*, in which a set of agent types is defined assigning to each agent one or more different capabilities. The first step on the architectural design is the choice of the architectural style to be used for the system. For the eSAP system a client/server architectural style has been chosen [17], as opposed to a mobile agents style. Mainly, this decision took place because the client/server style contributes more to the security of the eSAP system than the mobile agents style [17].

When the architectural style has been chosen, the next step of the architectural design stage aims to decompose the system in order to identify internal actors who will satisfy the system's goals. In the presented example, the eSAP actor is decomposed, as shown in Figure 7, to internal actors and the responsibility for the fulfilment of the eSAP's goals is delegated to these actors.

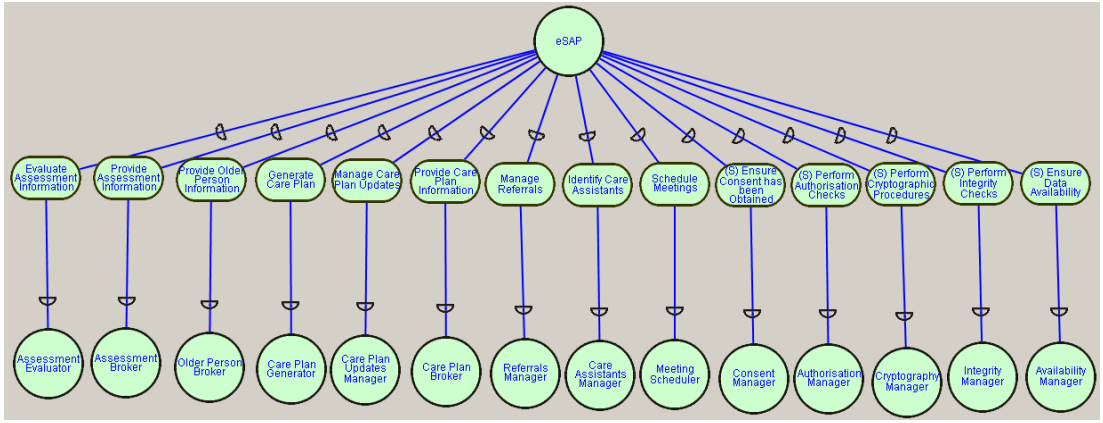

Figure 7: Partial Decomposition of the eSAP system

For instance, the Evaluate Assessment Information goal is delegated to the Assessment Evaluator, whereas the Provide Assessment Information goal is delegated to the Assessment Broker. In addition, the Older Person Broker and the Consent Manager actors have been introduced to the eSAP system to fulfil the responsibility (identified during the late requirements analysis – see Figure 5)of the eSAP system to satisfy the secure dependency Obtain Older Person Information together with the Share Information Only if Consent Obtained security constraint.

However, the new introduced actors must be furthered analysed and their dependencies with the other (existing and new) actors must be furthered investigated. Such an analysis is important since it helps developers to identify dependencies between new and existing actors, introduce new actors to the system-to-be and, as a result of this, refine the goals of the system or even possibly introduce new goals to the system, which would be very hard to identify otherwise.

An important point at this stage of the development is the identification of the actors to deliver the security goals of the system. This is important because, as opposed to the identification of actors to deliver the functional goals of the system, the identification of the "security" actors is a difficult task, especially for developers with minimum knowledge of security (this is the case for most of the information system

developers). To help developers with this task, Secure Tropos employs a security pattern language [17]. Security patterns can greatly help to identify the required actors in a structured manner, which does not endanger the security of the system, by providing a solution customised to the problem.

For example, from the internal analysis of the eSAP, presented in Figure 6, it was concluded that Information Flow, Authentication and Access Control checks must be performed in order for the eSAP system to satisfy the secure goal Ensure System Privacy. In the case of the information flow secure task, the eSAP should be able to control how information flows within the system, and between the system and other actors. For example, the system should be able to control who requires access to the system and, by considering the security policy, to grant or deny access to the system. With respect to the Authentication checks, the system should be able to authenticate any agents that send a request to access information of the system, and in the case of the access control, the system should be able to control access to its resources.

The pattern language[7] proposed in [17] can be used to fulfil the above-mentioned secure goals of the eSAP system. Three main patterns are employed by the language: The Agency Guard, which provides a single non-bypassable point of access, the Agent Authenticator which provides authentication services to an agency, and the Access Controller, which provides access to resources according to a security policy. Therefore, in the case of the eSAP, the Agency Guard pattern can be used to check grant/deny access to the system according to the security policy, the Agent Authenticator pattern can be used to provide authentication checks and the Access

[7] Although different patterns might be employed, the chosen language has been developed with agent orientation in mind –as opposed to many object oriented languages- and as such it provides concepts similar to the Secure Tropos concepts. This results in a unified modelling process.

Controller pattern to perform access control checks. The use of these patterns not only satisfies the fulfilment of the secure goals of the system but also guarantees the validity of the solution.

To apply a pattern, the developer must carefully consider the problem to be solved and the consequences that the application of each particular pattern will have on the system. Figure 8 illustrates a possible use of the Agency Guard, Agent Authenticator and Access Controller patterns in the eSAP system.

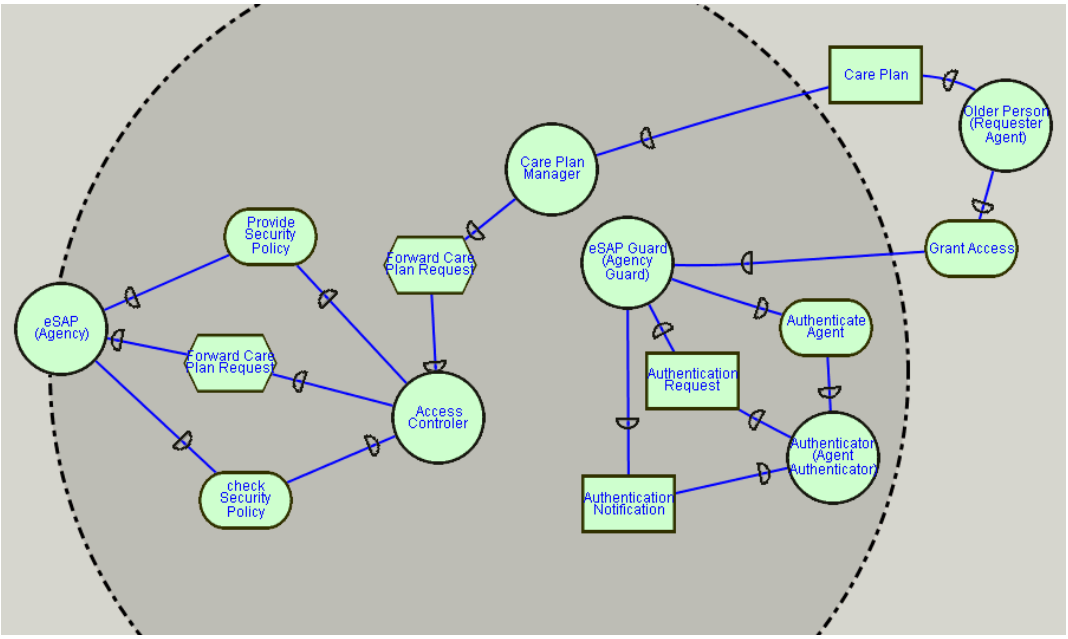

Figure 8: Using the AGENCY GUARD, the AGENT AUTHENTICATOR and the ACCESS CONTROLLER patterns in the development of the eSAP

In particular it shows how the secure goals Check Information Flow (problem), Check Authentication (problem) and Check Access Control (problem) can be satisfied. The Agency Guard satisfies the goal by providing a single non-bypassable point of access to the system (solution), the Agent Authenticator satisfies the goal by authenticating each agent that tries to access the system (solution) and the Access Controller controls access to the resources of the system (solution). The use of the patterns helps developers to delegate the responsibilities of particular system security

goals to particular actors defined by the patterns. In addition, the developer knows the consequences that each pattern introduces to the eSAP system.

The application of the Agency Guard means that only the Agency Guard must be tested for correct enforcement of the agency's security policy (consequence), the application of the Agent Authenticator means that during implementation only the Agent Authenticator must be checked for assurance (consequence), whereas the application of the Access Controller means that different policies can be used for accessing different resources (consequence).

Therefore, as derived from the application of the pattern language, the eSAP delegates responsibility for the fulfilment of the Perform Authorisation Checks security goal to three new actors, the eSAP Guard (delegated the Check Information Flow secure task), the Authenticator (delegated the Check Authentication secure task), and the Access Controller (delegated the Check Access Control secure task) as shown in Figure 9.

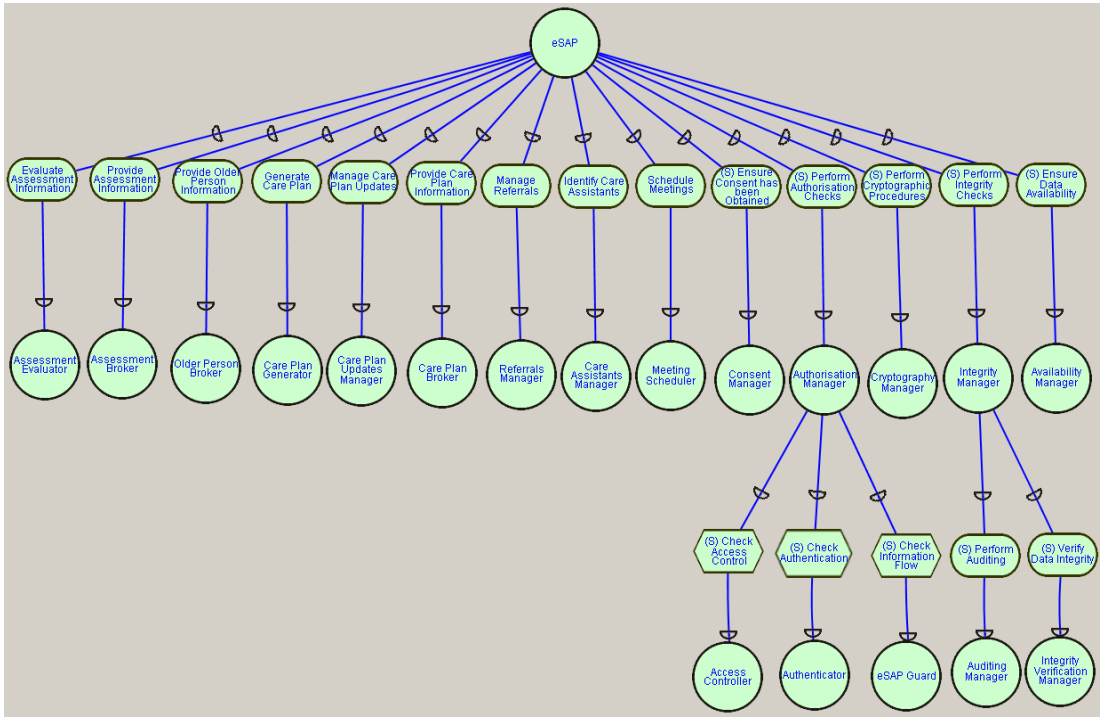

Figure 9: Decomposition of the authorisation and integrity managers

In addition, the Secure Tropos methodology introduces extended actor diagrams, in which the new actors and their dependencies with the other actors are presented. As an example, consider the extended diagram depicted in Figure 10.

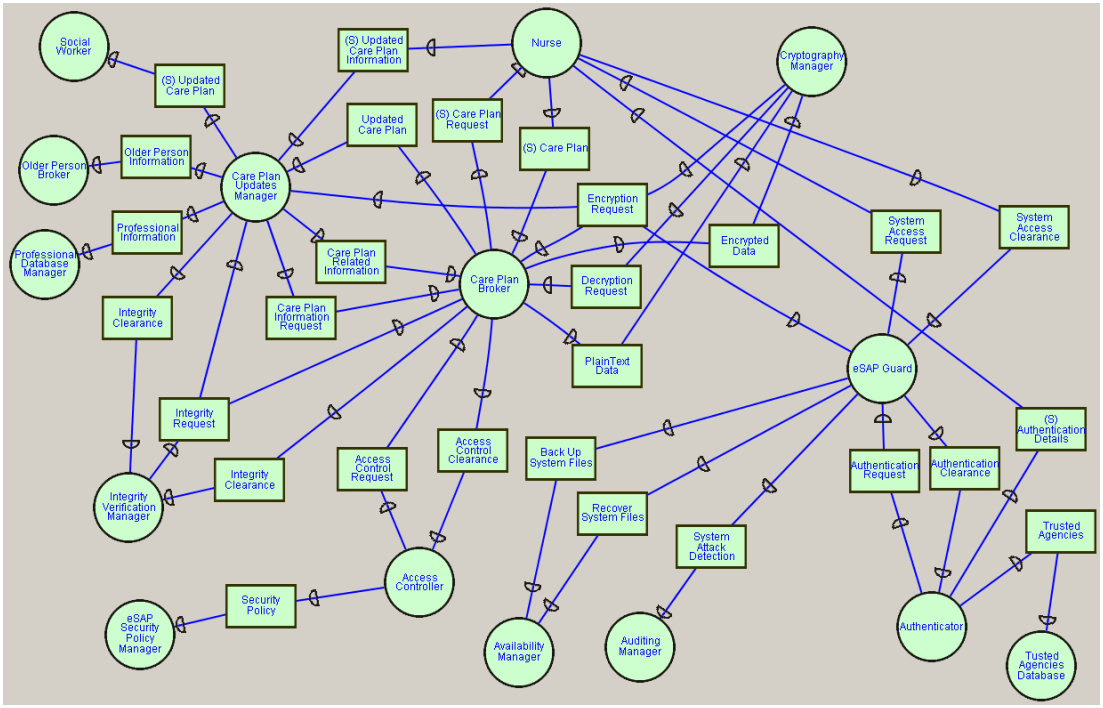

Figure 10: Extended diagram for the eSAP

In this diagram[8] the resource dependencies between the Social Worker, the Older Person Broker, the Care Plan Updates Manager, the Nurse, the Cryptography Manager, the Care Plan Broker, the eSAP Guard, the Access Controller, the Availability Manager, the Auditing Manager, the Integrity Verification Manager, and the Authenticator are modelled. An important point to consider is the addition of new actors, such as the Professional Database Manager, the eSAP Security Policy Manager, and the Trusted Agencies Database as derived from the analysis of the other actors in order to fulfil the delivery of specific resources such as the Professional Information, or the system's security policy.

[8] In order to keep the diagram simple, only some of the actors of the eSAP system have been included in this diagram.

In addition, the extended diagram can be further analysed in order to model more precisely the actors. Consider for instance, the extended diagram with respect to the Assessment Evaluator actor, as depicted in Figure 11. The Assessment Evaluator has been delegated the responsibility to satisfy the goal Evaluate Assessment Information. To fulfil this goal, the Assessment Evaluator depends on two internal actors, the Assessment Analyser and the Evaluation Synthesiser. The first is responsible for obtaining the Assessment Information secure resource, identify the problems of the Older Person according to the Assessment Information and provide the Problems to the Evaluation Synthesiser. The latter is responsible for obtaining the Evaluation Request, and the Problems and providing the Assessment Evaluation secure resource to the actor requesting the information (in the presented analysis to the Social Worker) after considering the Problems, the Available Professionals, the Required Skills and the Proposed Actions resources.

In addition, at this stage, the capabilities identification, in which the capabilities needed by each actor to fulfil their goals and tasks are modelled. Each actor's capabilities can be identified with the aid of the extended actor diagram, since each resource dependency relationship can give place to one or more capabilities triggered by external events. For example the resource Evaluation Request, shown in Figure 11, calls for the capability Obtain Evaluation Request for the Evaluation Synthesiser actor and Provide Evaluation Request for the Social Worker actor.

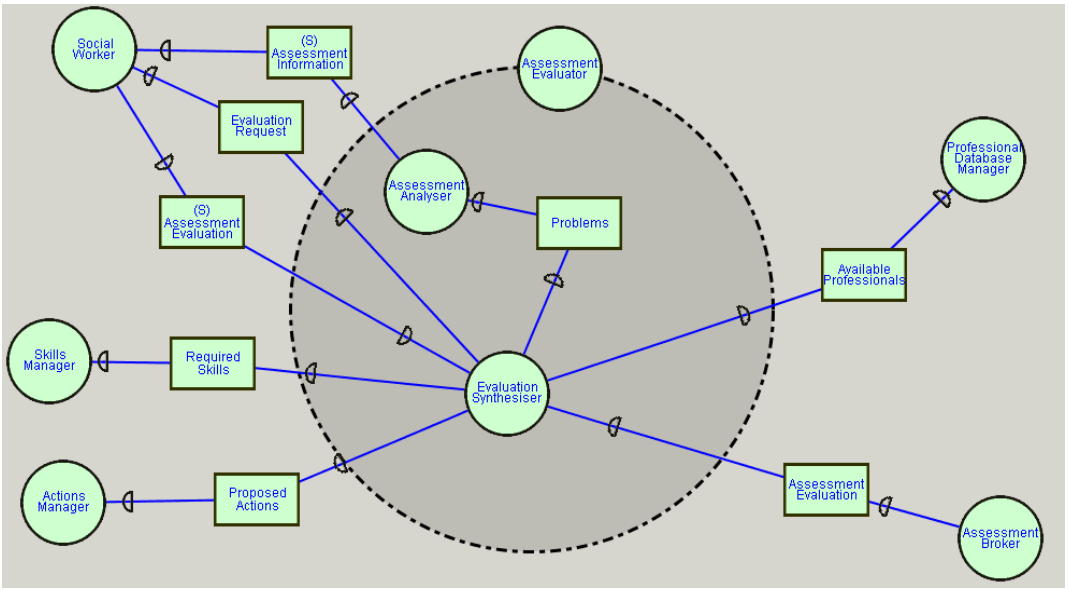

Figure 11: Extended actor diagram with respect to the Assessment Evaluator

In addition, capabilities are identified taking into account the resources of the extended actor diagram. For example, as identified in the early requirements analysis, for the eSAP system to satisfy the Ensure System Privacy secure goal, only encrypted data transfers across the network should be allowed. Therefore, the Assessment Information resource sent from the Social Worker to the Assessment Analyser must be encrypted. Because of this the Social Worker actor should be provided with capabilities to encrypt and decrypt data. Later in the detailed design, each agent's capabilities are further specified and then coded during the implementation phase. Table 1 reports the actors of Figure 11 and their capabilities as derived from the dependencies that exist between them.

When all the actors and their capabilities have been identified, the next step of the architectural design is the agents' assignment. During this step a set of agents are defined and each agent is assigned one or more different capabilities, as shown in Table 2. The capabilities are assigned according to the actors that the agent represents.

Table 1: Actors and their capabilities with respect to the extended diagram of Figure 11

Actor	Capability	Capability Id.
Assessment Analyser	Get Assessment Information	1
	Provide Problems	2
Evaluation synthesizer	Get Problems	3
	Get Evaluation Request	4
	Provide Assessment Evaluation	5
	Get Required Skills	6
	Get Available Professionals	7
	Get Proposed Actions	8
Skills Manager	Provide Required Skills	9
Professional Database Manager	Provide Available Professionals	10
Actions Manager	Provide Proposed Actions	11
Assessment Broker	Get Assessment Evaluation	12
Social Worker	Provide Assessment Information	13
	Provide Evaluation Request	14
	Get Assessment Evaluation	15
	Encrypt Data	16
	Decrypt Data	17

Table 2: Agent types and their capabilities

Agent	Capabilities
Assessment Evaluator	1,2,3,4,5,6,7,8
Skills Manager	9
Professional Database Manager	10
Actions Manager	11
Assessment Broker	12
Social Worker	13,14,15,16,17

6.2.4 Detailed design

The next step of the development process involves the specification of the system's components.

Detailed design stage aims at specifying agent capabilities, plans, and interactions and it is intended to introduce additional detail for each architectural component of the system. In Secure Tropos the detailed design stage is based on the specifications resulting from the architectural design phase and the reasons for a given element, designed at this level, can be traced back to early requirements analysis, a very important advantage of the methodology. Another important advantage of the methodology is that it is well integrated with existing work. Thus, for the detailed design stage, Secure Tropos adapts a subset of the Agent Unified Modelling Language (AUML) diagrams proposed in [23]. 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. Consider for example, the Receive Care Plan Request secure capability of the Care Plan Broker. This can be depicted as shown in Figure 12.

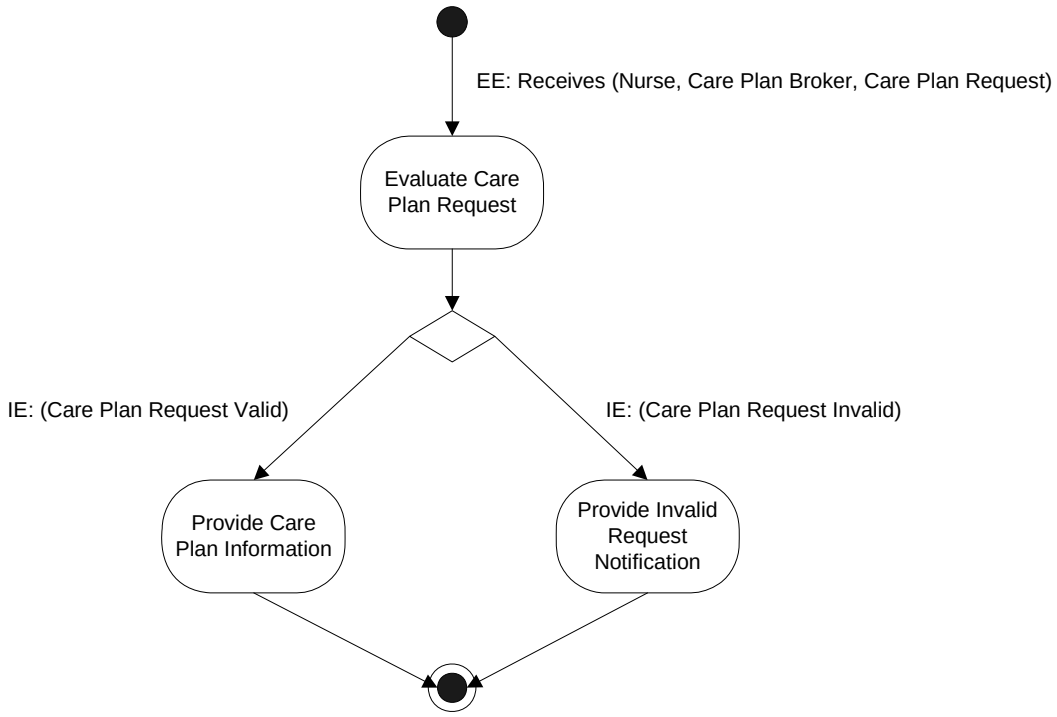

Figure 12: Capability diagram for the Receive Care Plan Request **capability of the** Care Plan Broker **agent**

The Care Plan Broker initially accepts a Care Plan Request from the Nurse. Then the Care Plan Broker evaluates the request and either provides the requested information or notifies the requester (the Nurse in this case) that the request is invalid.

Plan Diagrams. Plan Diagrams are used to further specify each plan node of a capability diagram. The starting point of a plan diagram is the initiation of a plan and the end point is the termination of a plan. The different actions required by the plan are modelled as activity nodes together with the transitions (modelled as arcs) from one actions to a subsequent one. For instance, the Evaluate Care Plan Request plan depicted in Figure 12, is modelled as shown in Figure 13. The Care Plan Broker firstly tries to decrypt the incoming request. If the request is not encrypted then the agent categorises the request as not valid (all the incoming requests must be

encrypted) and the plan is terminated. If the request is successfully decrypted the next step involves the integrity check of the request. In case the integrity of the request is not verified the agent categorises the request as not valid and the plan is terminated. The last step involves reading the request in order for the agent to respond to it. It must be noticed that every incoming request follows a specific format, in order for the agent to be able to read it. If the request is readable the Care Plan Broker categorises it as valid request and the plan terminates, in any other case the request is categorised as invalid and the plan terminates.

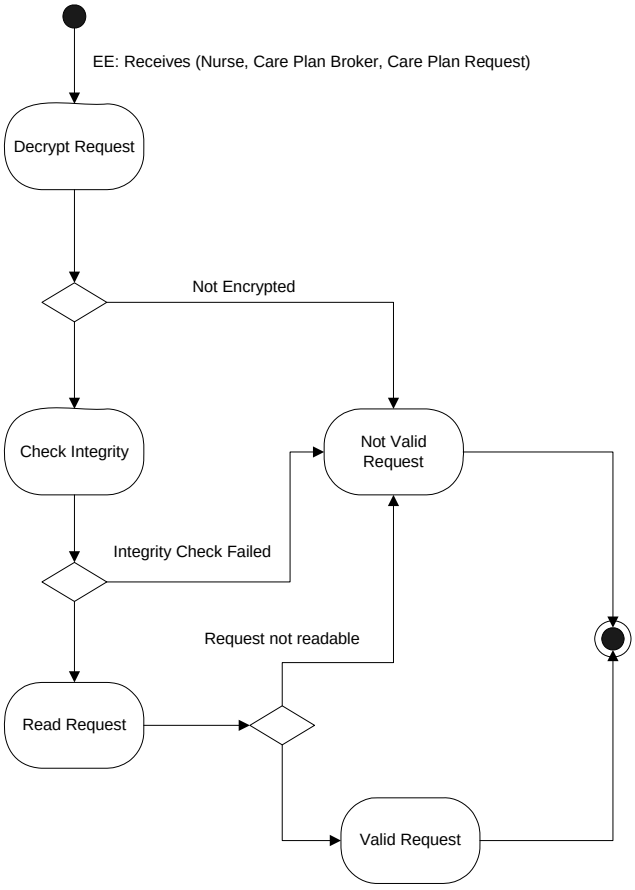

Figure 13: Plan diagram for the evaluate care plan request plan

Agent Interaction Diagrams. We apply in our case sequence diagrams modelling agent Interaction Protocols as proposed by [25]. Agent Interaction Diagrams capture the structural patterns of interactions between the agents of the system by emphasising the

chronological sequence of communications. Consider for example, the interactions that take place when the Social Worker tries to obtain access to the system as shown in Figure 14. The Social Worker sends an encrypted message to the eSAP Guard requesting access to the system. The eSAP Guard forwards the request to the Cryptography Manager for decryption. After the Cryptography Manager decrypts the request it forwards it plain text to the eSAP Guard. Then the eSAP Guard checks the authentication privileges of the Social Worker with the aid of the Authenticator. Then the Authenticator requests from the Social Worker to send their authentication details. When the Authenticator receives the authentication details of the Social Worker either provides an authentication clearance or rejects the authentication of the Social Worker. After the authentication clearance has been granted, the eSAP Guard provides system access clearance to the Social Worker.

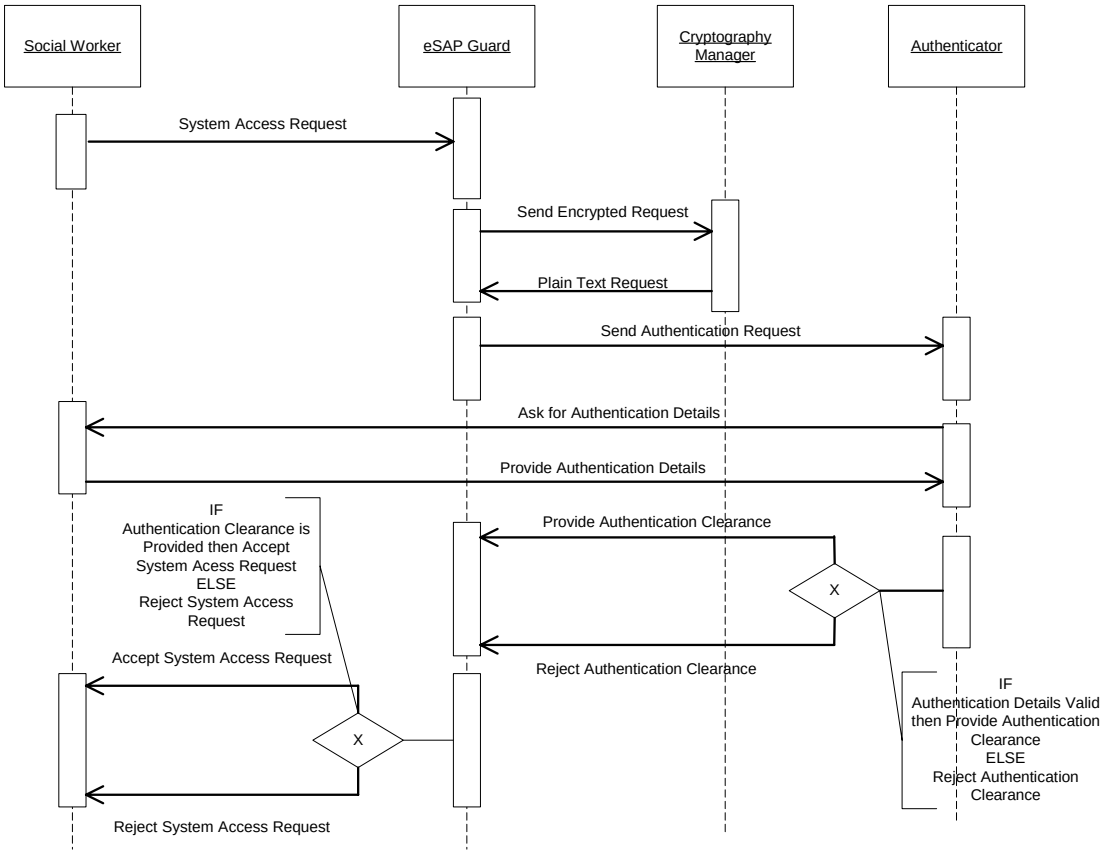

Figure 14: Interaction diagram for the Social Worker system access clearance

The detailed design stage takes place until all the components of the system identified during the architectural design have been specified in depth with the aid of the methods and diagrams presented in this section.

6.3 DISCUSSION REGARDING THE CASE STUDY

As a result of the application of Secure Tropos to the above case study, we are able to draw some important conclusions regarding the degree in which Secure Tropos satisfies the requirements identified in Section 2.

Requirement 1: A*n appropriate methodology should provide concepts and procedures to decompose effectively the system to smaller, easier to understand and develop components*.

The concepts that Secure Tropos employs throughout the development stages, allow developers to decompose easily the system to smaller components by mapping the system development concepts to entities of the real environment in which the system will be situated. For example, initially the system was modelled as one actor. By analysing the goals of the system, and assigning sub-actors to satisfy these goals, the decomposition of the system becomes easier. Moreover, the developers can verify whether the decomposed system meets the initial requirements by analysing if the new actors meet the delegated goals. In addition, the application of patterns during the architectural design of the Tropos methodology provides developers with an additional "tool" to help them to decompose the system effectively. On the other hand, Secure Tropos defines three different processes (Addition of new actors, Capabilities identification and agent assignment – for a reminder please see Section 5) to allow developers with less experience in decomposing a system to successfully achieve decomposition.

Requirement 2: A*n appropriate methodology should allow developers to iterate during the development stages and easily modify/add/delete the various components of the system*.

Although, for the sake of simplicity we presented the stages and processes of the Secure Tropos methodology in a sequential way, the development process is actually an iterative one, in which developers and users can move back and forward on the development to add/modify different components that might be needed by the system. For example, during the development of the eSAP case study, initially the goals of the care manager role had been assigned to the Social Worker actor. However, as it was concluded after discussions and input from the users (health and social care professionals), this is not always the case. It is possible a health professional will assume responsibility for the satisfaction of these goals. Therefore, we modified the actor diagram to include the role of the care manager, which in our case study is played by the Social Worker.

Requirement 3: A*n appropriate methodology should provide concepts that are easily understood not only by the system developers but also from the system's stakeholders/users.*

Although, we did not contact a detailed survey as to whether the concepts of the methodology are easy to understand and use by health and social care professionals, their involvement throughout the development process indicated they were feeling confident with them. After an initial "learning" period the users of the system (mainly health professionals) started to provide feedback about the system using the same concepts and notations as the developers. This we believe, made the analysis and design faster and helped greatly to establish a mutual understanding.

Requirement 4: A*n appropriate methodology should allow developers to model not only the system but also the environment in which the system will be deployed as well as the relationships between the different stakeholders/users of the system*.

One of the novelties of the Secure Tropos methodology is that it starts the development process from the early requirements analysis, in which the environment of the system is modelled. Regarding the eSAP case study, neglecting any reference to an electronic system during the early requirements phase allowed us to precisely identify the environment of the Single Assessment Process, and moreover identify the reasons for introducing an electronic system (see for example the Goal Diagram for the Nurse actor in Figure 3). In addition, the usage of concepts such as goals and dependencies allowed us to precisely identify all the relationships and the delegations of goals between the actors of the case study.

Requirement 5: A*n appropriate methodology should allow developers to consider security issues throughout the development stages by providing concepts and processes customised to security.*

By modelling the security constraints of the individual actors, developers are able to model the security requirements of the system according to the real security needs of its stakeholders. For example, during the eSAP system analysis, the lack of identifying the security constraints between the Nurse and the Older Person, or the Social Worker and the Benefits Agency would result in a design that would miss important information regarding the security of the system. Furthermore, by imposing security constraints, and differentiate between security-related and non-security-related goals and entities, developers can define together security and other functional and non-functional requirements of a system and at the same time provide a clear distinction between them.

This distinction helps towards the detection of possible conflicts between security and other requirements, and therefore allows developers to analyse those conflicts and propose possible ways towards a design that will overcome them, leading to the development of a more secure system.

7. Related Work

This paper proposes the Secure Tropos agent oriented software engineering methodology as a suitable methodology for the development of health and social care information systems.

Most of the methodologies used so far for the development of health and social care information systems are based on the object oriented software engineering paradigm [5], according to which an information system is decomposed to smaller components which communicate in terms of messages. The object orientation paradigm uses concepts such as object, classes, attributes, and methods, which are well known amongst the system developers. Beer et al. [26] employ object orientation techniques such as use case diagrams and collaboration diagrams to develop an information system for managing the provision of care. Lee and Asllani [27] propose the SCOPE IT object oriented methodology for designing patient service scheduling systems in health care. Krol and Reich [28] define three models, the object model that describes the hierarchical structure of objects in a system, the dynamic model that represents the sequence of operations of the objects on the system, and the functional diagram to represent the transformation of data within a system by means of data flow diagrams. On the other hand, Blobel [29] argues for the suitability of a component-based approach for the analysis and design of advanced health system architectures.

These approaches, although valuable, demonstrate some important limitations with regard to the development of health and social care information systems. First of all, we feel the detail on which they can model a complex information system, such as a health and social care information system, is restricted. As Booch, one of the "fathers" of object orientation, states [5, page 34] " for complex systems we find that objects, classes and models provide essential yet insufficient means of abstraction". Moreover, the concepts - such as objects, modules, components, and methods - and the modelling languages used by these approaches although very well known and understood within the software engineering community, can be quite confusing for someone not familiar with software engineering methods such as the health and social care professionals. This usually leads to misunderstandings during the crucial stages of the system requirements elicitation and analysis. This misunderstanding might propagate to errors during the design, which are usually cost a large amount of money and time to be corrected when they are discovered, if they are discovered before the implementation. Moreover, current object oriented methodologies fail to adequately consider security during the development of information systems [6], an important limitation, as discussed earlier, when developing information systems for the health and social care sector.

On the other hand, Secure Tropos is not the only agent oriented software engineering methodology. In the last few years many software development methodologies for multi-agent systems have been developed [30], such as MaSE [31], MESSAGE [32], and GAIA [33] amongst others. These approaches start the development process either from the late requirements (system analysis) or the design stages (system design). Such an approach however, is not desirable for the development of health and social care information systems, since the environment and the

interactions of the stakeholders play an important – possibly the most important – role when defining the system's functionalities. In contrast, Secure Tropos covers the full range of software development starting from the early requirements (system's environment analysis) and thus allow developers to capture not only the system-to-be but also analyse the environment that the system will be situated and therefore define more precisely the interactions of the system with its stakeholders/ users and as a result the functionalities of the system.

Moreover, unlike other modelling languages used by many agent oriented software engineering methodologies, the Secure Tropos graphical notation and the concepts used in the requirements and architectural design stages (stages that users need to provide feedback and their opinion) are more intuitive and comprehensible for people that are not experts in software engineering [18], such as health and social care professionals. Therefore, the methodology provides an effective means of interaction between the system developers and the users of the system.

Moreover, throughout the paper, we have indicated the necessity to consider security issues during the analysis and design of the electronic Single Assessment Process system. Secure Tropos is the only agent oriented methodology that provides a well defined process for identifying the security requirements of the system, and allow developers to develop a design that satisfies them.

8. Conclusions and Future Work

The number of health and social care information systems under development increases constantly. An important consideration for such systems is the involvement of the users/stakeholders (health and social care professionals) in the development process. Therefore, it is important that software engineering methodologies exist to assist not

only developers but also stakeholders and users. Such methodologies must provide adequate concepts that are understood by the system developers as well as the stakeholders/users of the system, enabling the latter ones to actively involved in the development of the system, and therefore guarantee the correct identification of the system's requirements.

This paper argues for the suitability of the Secure Tropos methodology for the development of health and social care information systems. The argument is mainly based on the degree in which Secure Tropos satisfies the requirements that a methodology for the development of health and social care system should demonstrate as well as on the Secure Tropos key features, such as the use of the same concepts and notations throughout the development process, and the consideration of security as an integral part of the development process. The applicability of the methodology has been demonstrated by applying it at the electronic Single Assessment Process system case study.

However, this work is not finished. Future work involves the application of the methodology to more health information systems, mainly from different domains, such as cancer information systems, in order to identify whether different health information systems require different modelling techniques or a unique methodology can be used for the development of all health information systems.

References

[1] A. C. Norris, Current Trends and Challenges in Health Informatics, in Proceedings of the 7th International Symposium on Health Information Management Research, Sheffield, June 2002.

[2] C. Dowd, B. Eaglestone, P. Bath, P. Procter (editors), Proceedings of the 7th International Symposium on Health Information Management Research, University of Sheffield, 2002.

[3] Department of Health, Single Assessment Process for Older People, http://www.dh.gov.uk/PolicyAndGuidance/HealthAndSocialCareTopics/SocialCare/SingleAssessmentProcess/fs/en, Last Accessed 11/04/05

[4] N. D. Birrell, M.A. Ould, A practical handbook for software development, Cambridge University Press, 1986

[5] G. Booch, Object-oriented analysis and design – with applications, The Benjamin / Cummings Publishing Company, 1994.

[6] T. Tryfonas, E. Kiountouzis, A. Poulymenakou, Embedding security practices in contemporary information systems development approaches, Information Management & Computer Security, Vol 9 Issue 4,pp 183-197, 2001

[7] L. Liu, E. Yu, J. Mylopoulos, Analyzing Security Requirements as Relationships Among Strategic Actors, in the Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS'02), Raleigh, North Carolina, October 2002

[8] M. Bradshaw, Software Agents, American Association Artificial Intelligence Publication, 1997

[9] M. Wooldridge, N. R. Jennings, Agent Theories, Architectures, and Languages: A Survey, Intelligent Agents, Wooldridge, Jennings (eds.), Springer-Verlag, pp 1-22, 1995

[10] N. R. Jennings, An agent-based approach for building complex software systems, Communications of the ACM, Vol. 44, No 4, April 2001

[11] M. Wooldridge, P. Ciancarini, Agent-Oriented Software Engineering: The State of the Art, In P. Ciancarini and M. Wooldridge (eds.), Agent-Oriented Software Engineering. Springer-Verlag, Lecture Notes in AI Volume 1957, January 2001

[12] C. Iglesias, M. Garijo, J. Gonzales, A survey of agent-oriented methodologies, Intelligent Agents IV, Lecture Notes in Computer Science, Springer-Verlag 1555, 1999

[13] N. R. Jennings, M. Wooldridge, Agent–Oriented Software Engineering, in the Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99), Valencia, Spain, 1999

[14] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos A. Perini, TROPOS: An Agent Oriented Software Development Methodology. Journal of of Autonomous Agents and Multi-Agent Systems. Kluwer Academic Publishers Volume 8, Issue 3, Pages 203 - 236, May 2004

[15] F. Giunchiglia, J. Mylopoulos, A. Perini, The Tropos Software Development Methodology: Processes, Models and Diagrams, Lecture Notes in Computer Science 2585, pp 162-173, Springer 2003

[16] P. Bresciani, P. Giorgini, The Tropos Analysis Process as Graph Transformation System, In Proceedings of the Workshop on Agent-oriented methodologies, at OOPSLA 2002, Seattle, WA, USA, Nov, 2002

[17] H. Mouratidis, A Security Oriented Approach in the Development of Multiagent Systems: Applied to the Management of the Health and Social Care Needs of Older People In England, PhD thesis, University of Sheffield, 2004.

[18] E. Yu, Modelling Strategic Relationships for Process Reengineering, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995

[19] J. Castro, M. Kolp, J. Mylopoulos, Towards Requirements-Driven Information Systems Engineering: The Tropos project, In Information Systems (27), pp 365-389, Elsevier, Amsterdam - The Netherlands, 2002

[20] A. Fuxman, L. Liu, M. Pistore, M. Roveri, J. Mylopoulos, Specifying and Analyzing Early Requirements: Some Experimental Results. In Proceedings of the 11th IEEE International Requirements Engineering Conference, 8th-12th September 2003, Monterey Bay, California U.S.A

[21] A. Fuxman, Formal Analysis of Early Requirements Specifications, MSc Thesis, University of Toronto, Canada, 2001

[22] P. Bertrand, R. Darimont, E. Delor, P. Massonet, A. Van Lamsweerde. GRAIL/KAOS: an environment for goal driven requirements engineering, In Proceedings of the 20th International Conference on Software Engineering (ICSE'98), IEEE-ACM, Kyoto, April 98

[23] 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.), Lecture Notes in Computer Science, pp. 91-103, Springer, Berlin, 2001

[24] I. Philp, Can a medical and social assessment be combined? Journal of the Royal Society of Medicine, 90(32), pp 11-13,1997

[25] J. Odell, 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"

[26] M.D. Beer, R. Hill, W. Huang, A. Sixsmith, An agent-based architecture for managing the provision of care – the INCA (Intelligent Community Alarm) experience, Proceedings of the workshop on Agents Applied in Health Care, at the 15th European Conference on Artificial Intelligence, France-Lyon, 2002

[27] S. M. Lee, A. A. Asllani, S. Trim, An Object-Oriented Approach for Designing Service Scheduling Support Systems, in International Journal on Services and Standards, Vol. 1, No. 1, 2004

[28] M. Krol, D.L. Reich, Object-Oriented Model of a Health Care System, in the Proceedings of the 11th International Symposium on Computer Based Medical Systems (CBMS'98), TX-USA, 1998.

[29] B. Blobel, Application of the component paradigm for analysis and design of advanced health system architectures, in International Journal of Medical Informatics 60(2000), pp. 281-301.

[30] J. Odell, P. Giorgini, J. P. Muller (eds), Proceedings of the Fifth International Workshop on Agent Oriented Software Systems (AOSE'04), N.Y. –USA, July 2004.

[31] Scott A. DeLoach, Modeling Organizational Rules in the Multiagent Systems Engineering Methodology, Proceedings of the 15th Canadian Conference on Artificial Intelligence (AI'2002), Calgary, Alberta, Canada. May 27-29, 2002.

[32] R. Evans, P. Kearney, J. Stark, G. Caire, F. J. Carijo, J. J. Gomez Sanz, J. Pavon, F. Leal, P. Chainho, and P. Massonet. MESSAGE: Methodology for Engineering Systems of Software Agents, AgentLink Publication, September 2001

[33] F. Zambonelli, N. R. Jennings, M. Wooldridge, Organisational Abstractions for the Analysis and Design of Multi-Agent Systems, P. Ciancarini and M. Wooldridge (eds.), Agent-Oriented Software Engineering, Springer-Verlag, Lecture Notes in AI, Vol. 1957, January 2001