TRANSACTIONS ON DATA PRIVACY 8 (2015) 273–296

A model driven approach to data privacy verification in E-Health systems

Flora Amato*, **Francesco Moscato****

*DIETI, University of Naples Federico II, Naples, Italy.

**DiSCiPol, Second University of Naples, Caserta, Italy.

E-mail: flora.amato@unina.it, francesco.moscato@unina2.it

Abstract. Last years experienced the growth of new technologies able to remotely monitor health state of persons. This includes both (even complex) Medical devices and all kind of *wearable device*. In addition, with the increasing use of Cloud technologies to manage and store sensitive data from patients, the problem of assuring data privacy is more and more important in E-Health systems. Privacy requirements in Medical domains are not only defined by service providers or users, but Legal rules regulate the whole management and storage processes of health records. The use of Model Driven techniques for E-Health systems is appealing especially if formal verification of privacy requirements is enacted. In this paper we extend the MetaMORP(h)OSY modelling profile in order to explicitly consider privacy requirements for data. A novel model transformation algorithm is described for the application of Model Checking techniques to privacy verification.

Keywords. E-Health, Privacy, Model Driven Engineering, Formal Verification

1 Introduction

In the last years, everybody talked about *E-Health*: from physicians to computer scientists, from national and local institutions leaders to common citizens world wide. Anyway few people have come up with a clear definition of this relatively recent term. Probably it was created with a meaning similar to other *e-things* like e-mail, e-commerce and so on, attempting to convey to health care the principles of Internet communication and distribution of electronic information. The same term has its own meaning in scientific and academic environments, both in medical and computer science domains. Indeed, the term encloses more than the mere technological development: it is an emerging field in the intersection of medical and computer sciences where health services and information are improved, fastened and enhanced through the use of Internet and computer science technologies and methodologies.

E-Health allows for an increase of efficiency in health care, decreasing costs, sharing diagnostics and information about therapeutic interventions. It also improves quality and reduces delay of medical actions enabling fast second opinion requests and supporting for remote suggestions and training from specialists.

Superficially speaking, E-Health seems a Panacea for all problems in Medicine. Indeed, E-Health inherits all problems both from Medical and Computer Science domains. In par-

ticular Data Privacy is an hard problem to deal with for it unfortunately adds legal requirements (and hence the Legal domain) to the others.

It is no more a problem related to the privacy of data contained in (eventually digital) Clinical Records: in the age of Cloud storage and computing, and of *Internet of Things* [1], medical devices and new technologies are becoming more and more linked one to the other. They allow for monitoring health conditions of patients. Thanks to communication systems and, nowadays, to Cloud architecture, Monitoring is a real-time, remote activity. Wearable technologies like smart phones and watches, fitness bands couple with medical (sometimes implanted) electronic probes in order to provide some *data collectors* with biological information.

The main problems with these new technologies is that they trigger several privacy issues because of the large amount of sensitive data on patients' health condition. Data are usually collected on remote, distributed databases and their management must be compliant to privacy laws applicable in the patients' countries.

In this scenario it is really difficult to verify if a computer system respects data privacy requirements because of the heterogeneity of elements to consider. Sensors, monitors of biological probes, distributed storages (usually on Cloud systems), Services for E-Health are usually proactive, concurrent systems based on different hardware and software architectures. In addition, Users' behaviours must be considered since they are active actors of the whole system. In this context, systems designer and analysts can manage laws and directives about data privacy as requirements. Their specification should be managed like Service Level Agreement (SLA) for Cloud systems. Indeed, a SLA generally requests some properties on a system such as availability, dependability or given performances indexes. Of course these properties also apply to E-Health systems. In addition, we think that It is possible to analyse data privacy requirements as state reachability problem. In fact national laws usually define data privacy requirements with sentences like: "The persons who are authorized to access data have to be enabled by patient". This is the same of asserting that the only people having a formal authorization can reach the system state where access to data is granted.

This work shows how Model Driven Engineering (MDE)[2] enables formal, automatic verification of E-Health composite processes with regards to Data Privacy properties. In particular, we will exploit MetaMORP(h)OSY framework in order to provide formal modelling profiles and methodologies in order to define models of E-Health Systems and to analyse privacy requirements with formal methods. Model Driven Engineering extends the scope of OMG's Model Driven Architecture (MDA)[3], MDE is wider in scope than MDA. It combines design, automatic generation of system components and requirements verification during all its life-cycle. MDE focuses on creating running components of a system by their abstract (and formal) representation, leaving verification and (some) implementation issues to automatic translators. Requirements on abstract models are easier to analyse and formal verification can be exploited as well during design phase. In order to reduce the introduction of implementation errors, MDE methodologies try to implement automatically components from design models. In this way, introduction of errors that may lead to unsatisfying requirement can be reduced producing systems *correct by construction*.

In this paper we will use MetaMORP(h)OSY MDE framework, that is based on a Multi-Agent modelling paradigm. As we will describe in the following, this paradigm suites well the complexity of E-Health systems. Multi-Agent Systems[4] Models are promising formalisms for the analysis of complex services, where several actors executes actions proactively. E-Health systems have this properties for their nature of distributed and heterogeneous systems. It defines actors and system components by using a meta-formalism that

extends Unified Modelling Language (UML). The main model is then analysed by means of formal models that are obtained from the UML model with model transformation algorithms.

MetaMORP(h)OSY framework was introduced in [5] and it is based on Papyrus [6] and defines profiles for the definition of a modelling language for real-time MAS description. The language is compliant with Object management Group (OMG) MARTE[7] specification, in order to make *MetaMORP(h)OSY* compatible with other tools supporting the standard. Verification at every life-cycle step is performed by implementing translation algorithms which translate design, simulation and run-time description into formal models [8, 9, 10, 11].

The Framework and the methodology has been successfully used in order to model, analyse and monitor at run time Web Services and Cloud systems [12, 13].

In this paper we extend the MetaMORP(h)OSY modelling profile in order to explicitly consider privacy requirements for data. A novel algorithm is described for implementing a horizontal model transformation [14] for the application of Model Checking techniques to privacy verification.

This paper is organized as follows: section 2 introduces the problem of data protection and data privacy in E-Health systems pointing at legal issues that regulates them; section 3 describes the architecture of the MetaMORP(h)OSY framework, explaining the methodology on which it is based. Section 4 reports the description of the modelling profile used in this work in order model E-Health system and privacy requirement. Section 5 contains the description of a case study where the methodology is applied and the modelling profile is used. It contains a description of the translation algorithm used to verify formally the privacy requirement. Section 6 report some issues about the State of the art and, finally, section 7 contains some concluding remarks.

2 E-Health and data protection issues

Hospitals and healthcare organizations need to manage patient information in efficient and effective manner. At this aim, healthcare facilities are adopting information systems that support health workers in clinical procedures and activities, helping hospitals to be compliant with national and international recommendations. It is necessary for these systems to access an indescribably large amount of medical information and deeply analyze them, in order to give insights to Evidence-based health services and decision-making, in which the doctors contextualize the best available research evidence by integrating it with their individual clinical expertise and their patient's expectations. Although, with the emergence of the HL7[15] or European Health informatics - Electronic Health Record Communication (EN 13606)[16] standards, information is managed in structured way, there are still free fields, where medical staff can enter unstructured information.

Most of the medical data is unstructured and can reside in multiple different places as electronic medical records, results of laboratories, images, medical reports and so on. Accessing to this valuable amount of information and making advanced analytics is decisive, not only to improve the patient care and the outcomes, but also to give insights to Evidence-based health services and decision making, in which the doctors contextualize the best available research evidence by integrating it with their individual clinical expertise and their patients values and expectations.

Safety and quality of Medical care seems to be hot topics also in the international community: the International Joint Commission Organization developed a program of quality

evaluation and control that takes into account both patient oriented operation standards and management standards with the goal of preventing every kind of errors. It is clear that a system able to provide support in every single process or activity would be extremely esteemed by health workers and local health authorities: it would make hospitals compliant with national and international recommendations and observant of patient oriented operation and management standards.

Last years experienced the growth of new technologies able to remotely monitor health state of persons. This includes both (even complex) Medical devices and all kind of *wearable device* able to read and communicate biological information like implanted tools, smart watches, fitness bands etc. Recently, monitoring services, data communication and storage benefit from Cloud services and databases. Using Third parties services for storing information is good for maintainability and availability, since Cloud providers grant services with requested Service Level Agreement, but the other side of the coin is that companies and institutions should asses the legal implications of allowing distributed storing and management of sensitive data.

The exploitation of new technologies and Cloud-based architectures raises some privacy issues for the large amount of persons bio-metrical data collected and transferred to database accessible by practitioners, physicians and third parties that manage and store data. Such data processing is subject to considerable data protection obligations.

As already prescribed by [17], *the company managing the software installed on the device used for the remote monitoring system is subject to the privacy laws of the country where the device/user is located*. The Article applies also to non-European entities managing data from European people in European Country. This means that it is not sufficient, even for simple smartphone apps storing people pulses during a walk, to store data merely asking for privacy consent in a simple web-based form. A data protection notice must be provided that lists all information requested under the relevant privacy law. Therefore, a pop-up message displayed following the download of most apps would not meet the regulatory requirements.

In particular, privacy issues are more complex since local laws require a *written and signed* privacy consent for processing all sensitive data. In addition, data processing is allowed only within the limits of a written authorization document issued by the data protection authority. These authorities are strict in defining roles and allowed actions for people and systems accessing and processing to sensitive data.

Hospitals are generally considered to be the sole data controllers, with sponsors acting as *data processors* and cloud platform providers acting as *sub-processors*. Sponsors will usually need to be qualified as data controllers in order to have a higher level of discretion in the processing of the data, but data protection authorities have challenged such qualifications in several instances.

In addition, Italian law requires notification of data protection authorities for each use of remote patient monitoring system.

Even when data have been collected, patients must release a privacy consent specific to each purpose. This requirement does not apply in case of emergencies and for aggregated and anonymized data.

Notice that understanding actors of an E-Health system, their roles, actions and the policies for accessing data is important to define a modelling profile for data privacy requirements.

More in detail, for what Wearable devices and, in general, Internet of Things concerns, [17] states that raw data shall deleted as soon as the data necessary for the data processing have been extracted; Users shall be in control of processed data at any time; *Privacy by Design and Privacy by Default principles shall be followed*.

Notice that this last point implicitly promotes MDE techniques for design and validation of Privacy requirements.
Several other laws regulate transfers of data, type of hardware and software in E-Health etc., but they are out the aims of this work.

2.1 Other access policies to health data in Italy

We want here explicitly address some of legal rules (like the ones in [18]) related to the the main access policies to health records in Italy.
First of all [17] rules are considered also in Italian law. In addition:

- Authorization to access data can be granted by patients also for limited time periods;
- Three levels of authorization can be granted:
  - The first level of authorization (a temporary authorization) grants the access to all physicians in a medical structure (an hospital, a clinic etc.) only during the hospitalization period.
  - The second level of authorization grants the access to data to family doctors and to all authorized physician in an Hospital at any time.
  - The third level of authorization grants the access to all physicians during an emergency. Obviously, this authorization should be grant even if the patient is not able to give explicitly his consensus.
- Patients can access to their records at any time, and can hide the information they want also to authorized persons.

3 Model driven engineering and MetaMORP(h)OSY

The software architecture of MetaMORP(h)OSY framework is depicted in Fig.1.
It consists of an editor (to draw design models and to specify requirements); in a set of *Observers* used to evaluate properties (and hence to verify requirements) on MetaMORP(h)OSY models, and a set of *Translators* which enacts model transformation activities on models drawn in the editor. The MetaMORP(h)OSY editor is realized as an Eclipse/Papyrus [6] plugin.
Observers are used in order to evaluate properties both on (abstract) models and (real) running systems.
Translators implement both vertical and horizontal transformation [14]. Horizontal transformations preserve the same level of abstraction and are used to generate analyzable models. Vertical transformation produces models at different level of abstraction. *Abstraction* is enacted when the vertical translation generates a more abstract model from a lesser one; translation which produces finer grained models from other are called *Refinements*. Usually refinements are not fully automated and only stubs can be created. Requirements tracing in vertical transformations requires the creation of proper *Observers* able to verify abstract requirements on finer grained models. On the other hand, *Abstraction* can be used to exploit analyzable models of the design phase during run-time as will be shown in the section 5.
Unified Modeling Language (UML) is the language used for design models definition. MAS systems can be described by extending UML language. UML extension is realized by means of definition of proper meta-models called *modeling profiles*.

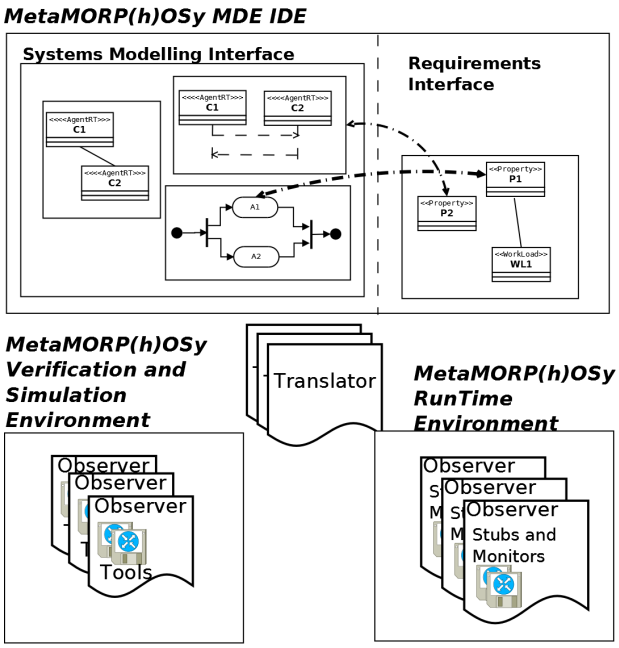

Figure 1: MetaMORP(h)OSY Architecture

Requirements are defined by a modeling profile too. The modeling profile for definition of thermal requirements in MetaMORP(h)OSY is shown in the following.

A deeper description of MetaMORP(h)OSY framework is in [19].

In MetaMORP(h)OSY, formal models drives requirement verification, at both design-time and run-time. At design time, model transformations algorithms translates MAS models into proper formal models. Requirements definition enables the choice of the Observer to execute in order to analyse models. This leads, at design-time, to results related to requirement verification. After validation of Design models, Vertical translations produce stubs for run-time system generation. In order to allow for Requirements Verification at run-time, the framework must create proper monitors and eventually it has to embed them into run-time systems. Run-time monitors for comparing actual behaviour with predicted one, use results from Design-Time analysis. If any difference appears, the framework generates a new formal model for verification purpose, with design-time properties substituted by parameters measured at run-time.

In MetaMORP(h)OSY, the definition of properties is also assured by the definition of a common ontology which describes formally cloud components and services[19]. Proper annotation techniques and models (as the one discussed in [20, 21]) should be coped with agents definition to specify requirements.

The framework allows for the use of the same formalism for system description and requirements specification: it provides a graphical language to users at design phase and requirements specification becomes part of the system model. This language is formally defined by a meta-model[22]. The name of the meta-model, with all the rules used to compose correct models, is (inheriting the UML jargon) *modelling Profile*.

MetaMORP(h)OSY modelling profile must allow for the definition of:

- Structural and behavioural views of the system, as well as interactions among components;
- Properties and Requirements to verify on the system (both at design and run time);
- Methods, Techniques and metrics for analysing or measuring properties and requirements;
- Expected and measured workloads of the systems

In addition, the modelling profile is able to specify temporal behaviour of agents, as well as real-time properties. In fact, the name of the modelling profile is RT-AML (Real-Time Agent modelling Language) since it was originally intended for describing real-time agent.

The modelling Methodology in MetaMORP(h)OSY is based on AML[23], that describes MAS by using a UML-based language. As described in[24], the original support for describing timed behaviours of agents in AML is poor. MetaMORP(h)OSY uses a Beliefs, Desires, Intentions (BDI) logic in order to describe Agents. *Beliefs* of agents, *Goals* they want to achieve and the available *Plans* to reach the goals defines agents structures.

RT-AML profile uses four diagrams for MAS behaviours description: Class diagrams, RT-Agent Diagrams, RT-Activity diagrams and RT-Sequence diagrams. the *Class diagrams*, are the same of UML diagrams. They are used when it is not useful to describe objects as agents (for example, for passive entities). The *RT-Agent diagrams* are the core of the RT-AML profile. They describes agents structures, goals and beliefs. In addition, they declares the actions and the plans they can execute. The *RT-Activity diagrams* allow for description of agents plans. The *RT-Sequence diagrams* describe agents collaborations (and/or competitions). Their main goal is to define exchanges of messages and events (real-time stimula) during execution of agents plans.

The main elements of RT-Agent diagram meta-model are depicted in Fig.2

They are: **AgentRT**, **PlanRT**, **DgoalRT** and **BeliefRT**.

AgentRT stereotype defines agents structures. **PlanRT** stereotype defines agents Plan which in turn define Agents behaviours. They are associated to goals: each plan is intended as a composition of actions aiming at reaching a goal state. **DgloalRT** stereotype defines decidable goals for AgentRT. A decidable goal is a goal whose reachability can be determined under real-time constraints. Finally, *BeliefRT* are used to model agents beliefs in terms of their inner state and of information about other agents and system environment. Stereotypes needed to model agents plans are defined in the RT-Activity diagram profile, that is show in Fig.3.

RT-Activity diagrams inherit all elements from UML activity diagrams, and re-define states and transitions to enable real-time constraint specification. In particular, **ActionStateRT**, **InitialState**, **FinalState**, **TransitionRT**, **SendObjectRT** and **ReceiveObjectRT** are the new stereotypes for this diagram.

InitialState and **FinalState** are the initial and the final states for the Activity Diagram. Usually FinalState refers to a DgoalRT. When starting a new plan, an agent has some Beliefs of its environment. Beliefs also composes its initial state and are specified as properties of InitialState stereotype. The usual time-related properties relate to the final (goal) state.

ActionStateRT is a step that composes the whole agent plan. It inherits the classic UML Activity State, but also GaStep from MARTE profile. An Action State is an activity State, which is subject to real-time constraints. In fact, this stereotype includes the following properties: the *ExecActionTime*, the time needed for the execution of the action and the *TimeOut*.

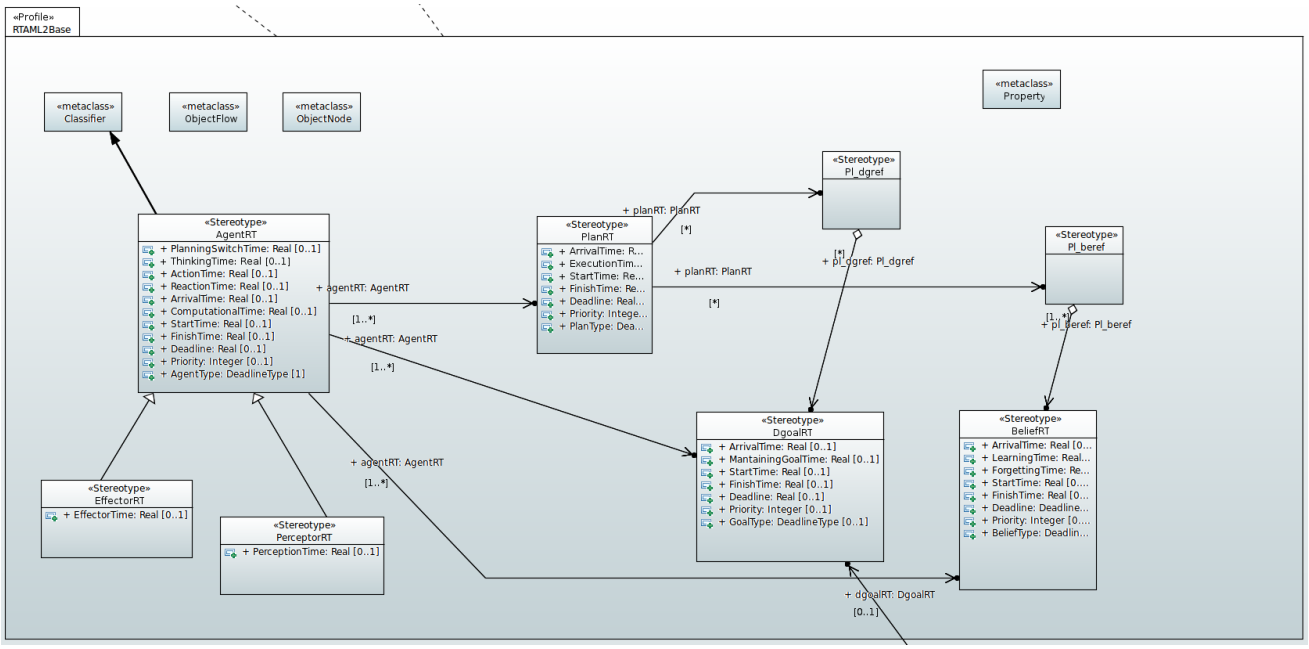

Figure 2: Agent Diagram Profile

Two particular types of ActionStateRT are the **SendObjectRT** and **ReceiveObjectRT**, which are used when messages and events are sent and received to and from other agents.

TransitionRT is a stereotype used to define transitions in activity diagrams with timing constraints. This stereotype too includes Time-related properties.

Finally, the main elements for **Sequence Diagrams** are: **StimulusRT** and **MultiStimulusRT** (see Fig.4).

The **StimulusRT** stereotype is used to define stimula with real time constraints in Sequence diagrams. The main properties for this element are: the *StartTime*, the time when the decision of sending a message is taken by an agent during the execution of a plan; the *SendTime*, the time when the message is sent; the *TransmissionTime*, the time needed for messages transmission; the *StartReceiveTime*, the time when the receiver begins to receive the message; the *StartEndTime*, the time when the receiver ends to receive the message; the *Deadline*, the deadline for transmission; the *MessageSync*, it is used to specify if the message is synchronous or not. **MultiStimulusRT** inherits the *StimulusRT* stereotype and it is used when defining redundancy in messages reception.

In order to define properties and requirement to define and analyse, MetaMORP(h)OSY defines separated modelling profiles. Definition of system models are independent from requirements specification and verification. This allows for requesting different requirements on the same system at different time and this is the reason for defining requirements and properties specification outside Agents stereotypes.

Fig.5 depicts the basic Property and Requirement profile.

A **Requirement** is a list of Properties that Observers will analyse on the models (or at run-time). Properties include **Functional** and **NonFunctional** ones. This profile declares stereotypes for several properties like *Availability, Reliability, Schedulability, Performances etc.*

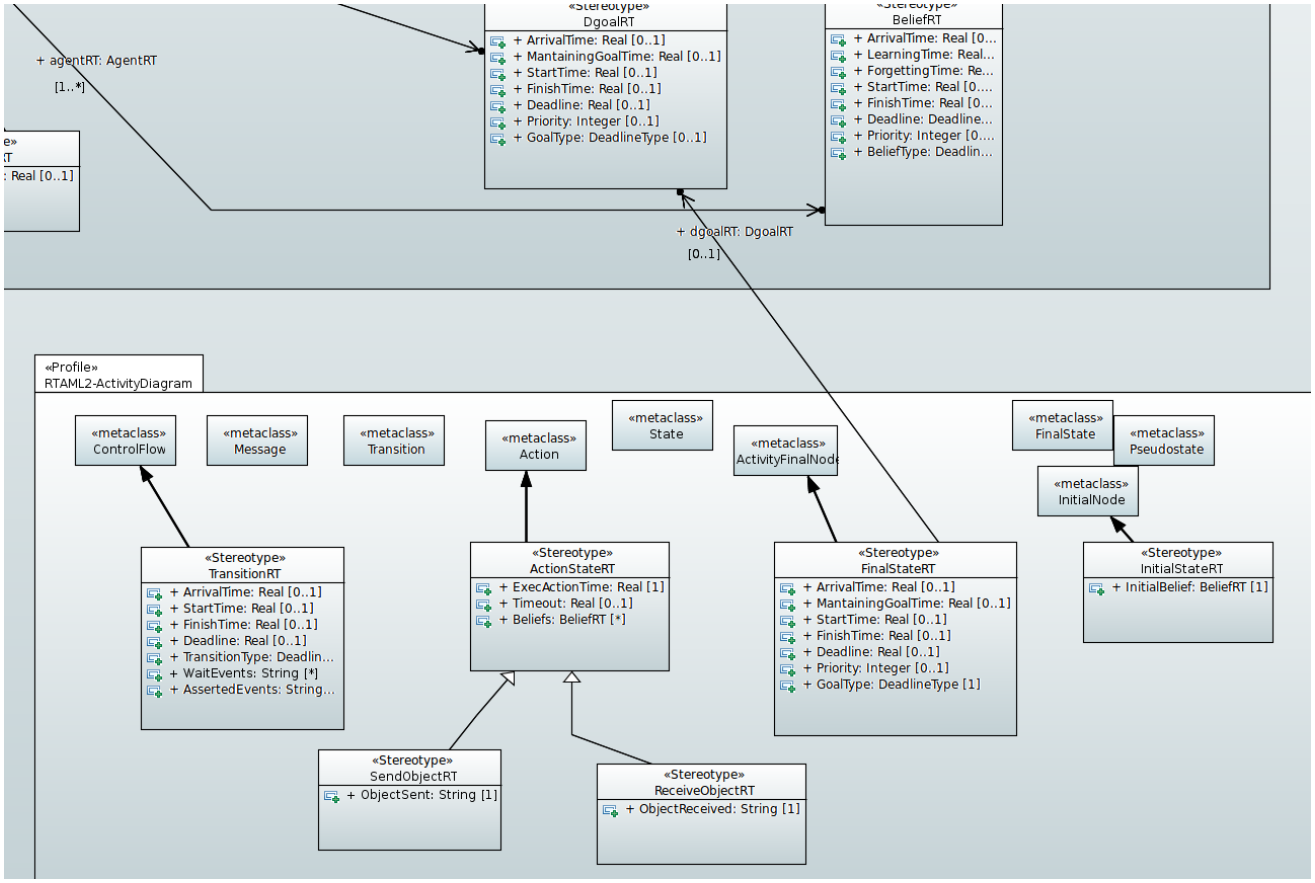

Figure 3: Activity Diagram Profile

Metrics are associated to non functional properties. In a MetaMORP(h)OSY model it is possible to specify the metrics to use for properties definition, evaluation and monitoring. Users can extend the profile if necessary adding more metrics and properties.

The bottom of Fig.5 reports the part of the profile that defines *Observers* stereotypes. The profile divides Observers in $Design\_Time$ and Run_Time Observers. They can be further specialized depending on translation algorithm and analysis methods used to analyse properties and requirements on system components. Observers can be associated to Structural and behavioural components and obviously to the property to analyse or to monitor, as well as to the metrics used to collect and produce results.

4 Data privacy: modelling and verification

In order to specialize MetaMORP(h)OSY methodology to manage verification of data privacy on E-Health systems, we must extend the basic RT-AML modelling profiles (described in section 3 to consider components from E-Health domain and in particular all elements introduced in section 2. A sketch of this profile is reported in Fig.6.

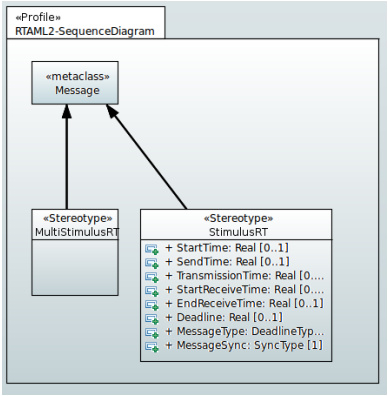

Figure 4: Sequence Diagram Profile

New Agents specialize the AgentRT stereotype:

- **Patient** represents patient agents that possess Electronic Health Record and grant the authorizations for the access to records by other agents;
- **DataController**: users can use this stereotype in order to declare agents like Hospitals that implement the E-Health system ;
- **DataProcessor** stereotype defines providers for records management and storing;
- **DataSubProcessor** usually refers to Cloud Storage providers;
- **Physician** is the entity related to doctors that interacts with patients. In particular, Family doctors and doctors who are eventually involved in emergencies specialize this stereotype.

On the other hands, this new modelling profile specializes some beliefs too. Known Electronic Health Records (**HealthRecord**) and eventually their encrypted versions (**EnCryptedHR**) refers to some Patients and are stored onto DataSubProcessors. Patients may hide some records (**HiddenRecord**)to any agent able to read his data and can grant authorization (**AccessGrant**) to access to data to any other agent. In particular, access to encrypted data must be coupled with the keys to manage encryption (**EncryptKey** and **DecryptKey**).

As introduced in section 1, this work manages the verification of privacy requirements as a reachability problem. This also because MetaMORP(h)OSY includes some Observers for Analysis of timed reachability of Agents Goals. Anyway the basic Observer profile has been specialized as show in Fig.7.

This new Observer, in fact, explicitly considers the Health records and the authorization granted to access records. This because, as we will explain in the next section, verification of Privacy is equivalent to the verification of reachability of some goals with a given configuration of beliefs. In brief: if an agent executes a plan (i.e. if a plan goal is reachable) to access some records without having valid authorization (i.e. with a particular configuration of beliefs), the verification of privacy requirement fails on the model.

Obviously, MetaMORP(h)OSY profile can be extended to manage different properties and architectures related to distributed data problems. For example, E-Health data management may lead to the use of public and private clouds, multi-provider clouds, etc. Requirements for multi-storage architectures can be easily integrated in the model. In particular,

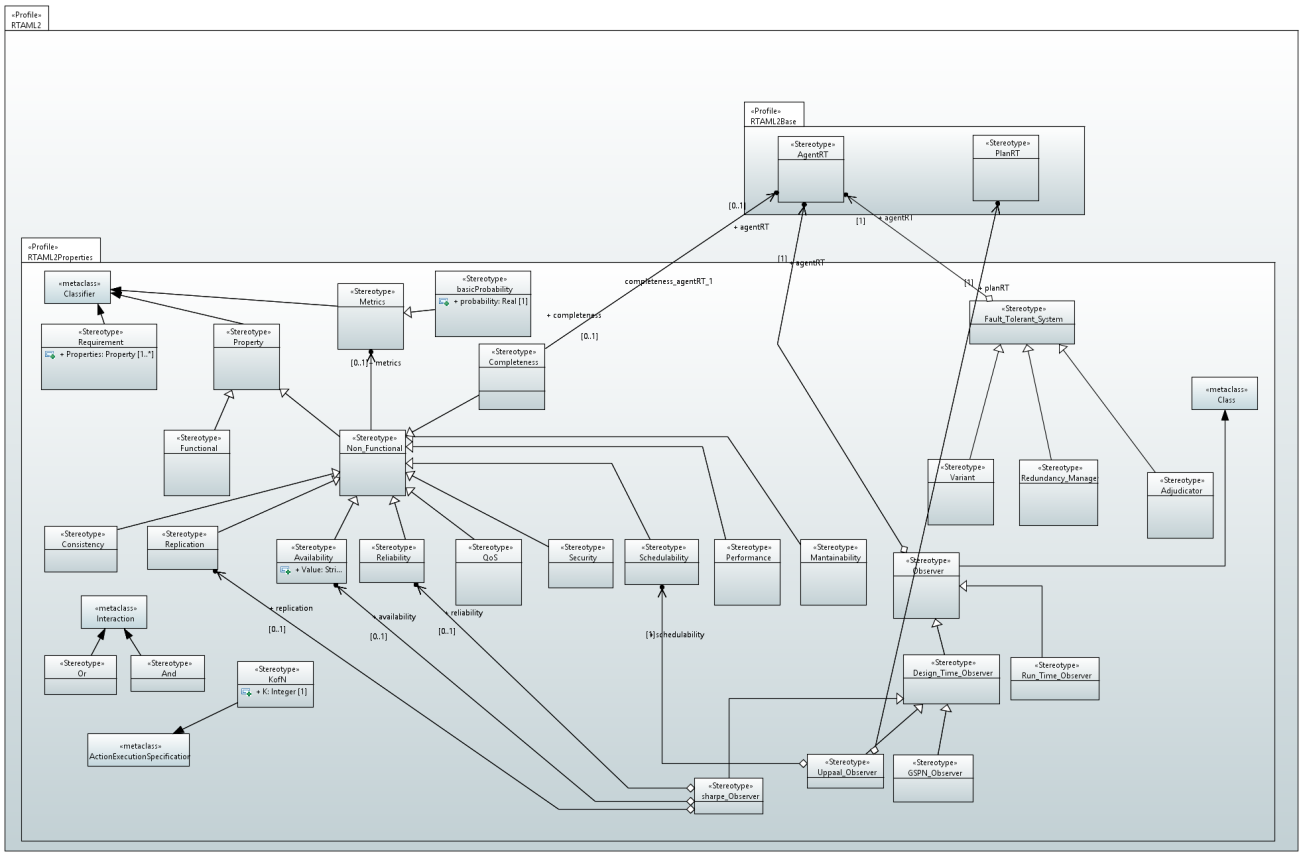

Figure 5: Properties, Requirements and Observer Profile

the application of MetaMORP(h)OSY methodology to the case of multi-storage services from multiple parties, which leads to a composite cloud service, is deeply discussed in [25].

Finally, the main MetaMORP(h)OSY profile was defined for description of real-time system and it includes many elements to define time constraints and timed behaviours of agents. Some Observers (both at design and run time) exists that are able to check temporal behaviours of the systems. Previous works ([5, 24, 26]) describes them. Notice that we can use this part of the profile to model time-variant data-policies and storage behaviours, but this is out of the scope of this work.

5 Case studies

In this section we want to show how it is possible to use MetaMORP(h)OSY methodology and the modelling profile described in the previous section in order to define a protocol for managing and storing E-Health records. We will show how it is possible to define an Observer to evaluate data privacy requirement in the framework.

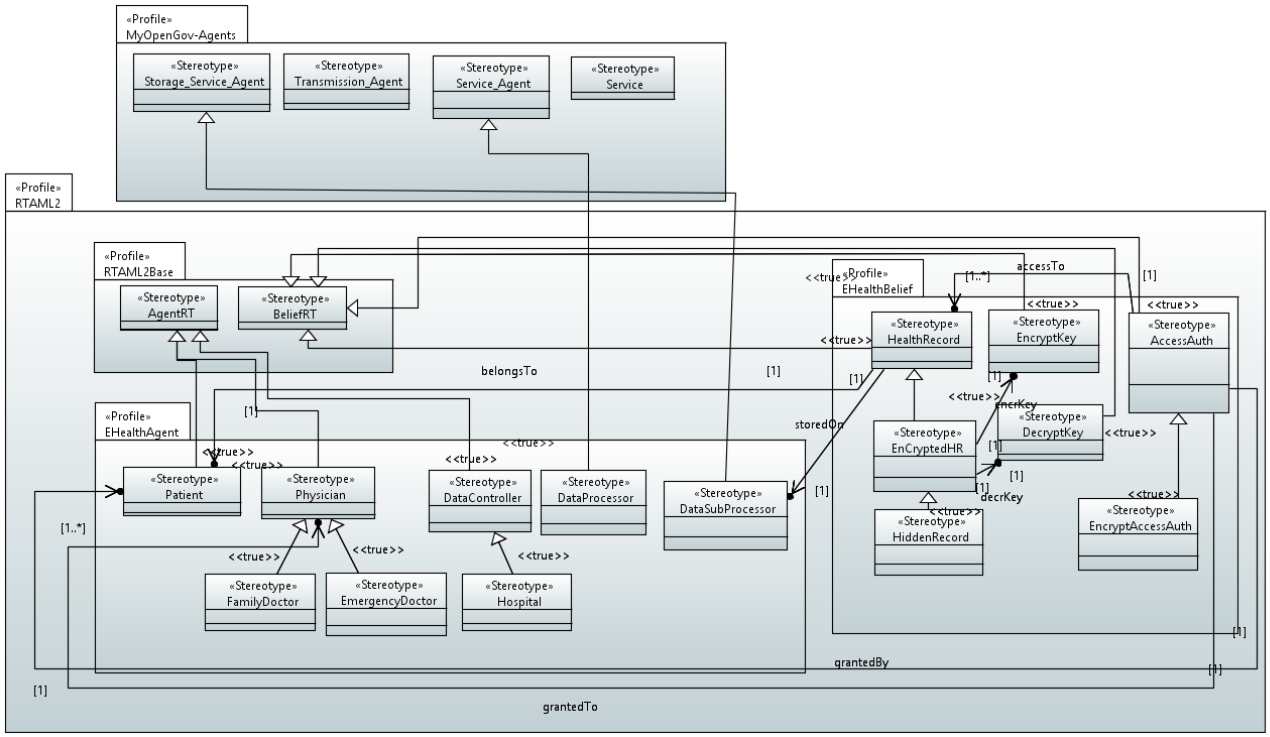

Figure 6: E-Health Modelling Profile

For Simplicity's sake, we will consider several simplifying assumptions. We consider the communication networks secure; we model data controllers, processors and sub-processors as a unique agent; only one health record is considered in the example; encryption exploits asymmetric algorithms (hence an encryption and a decryption will be considered for each agent); key sharing protocols are considered safe and they we will not explicitly consider them.

The problem to solve is the storage of sensitive data (the health record) on a third-party storage server (eventually a cloud storage). The same record should be readable by all the actors involved in the system eventually under some conditions. The generic owner of the record is a patient (*thePatient* in the following). It can read the record every time and it will store the the record on the storage service provider (*DataProvider*) only after a proper encryption action. *ThePatient* executes its encryption actions directly on his smart-device. Data outputted from the device is encrypted every time.

The physician (*theDoctor*) can access to *thePatient*'s record only if it is authorized by the patient, but theDoctor cannot own the decription key of thePatient. Hence when thePatient authorizes theDoctor to access to the record, he must produce an encrypted version of the record that theDoctor is able to read. This is possible because, during authorization, the Doctor shares its encryption key with the Patient.

In addition, in order to manage emergencies, an encrypted version of the record has to be stored on the DataProvider too. This implies that another encrypted version of the record must be provided and saved. In this case, a common encryption key is used which is

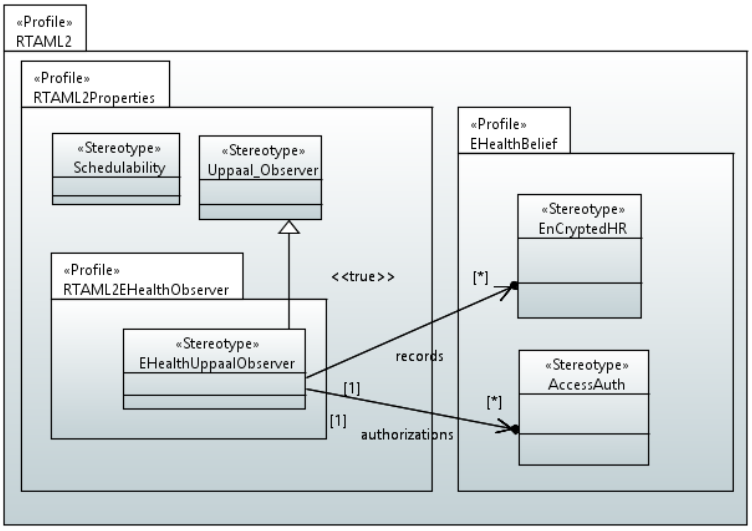

Figure 7: E-Health Observer Profile

accessible only during documented emergencies in hospitals.

Notice that in this case, The DataProvider will never access to raw health record in any case.

In order to produce a MetaMORP(h)OSY model of these agents, we must define an Agent Diagram describing the structural view of the agents. The Agent Diagrams declare the structure, the plans, the goals and the beliefs of each agents. In the following we will omit the inner properties of each element in the diagrams for brevity.

Fig.8 depicts the Agent Diagram for thePatient, whose class has the stereotype *AgentRT* described in section 3. His beliefs contain information about thePatient's, theDoctor's and emergency keys. They are identified by the stereotypes **encryptKey** and **decryptKey**. In addition accessAuth and healthRecord stereotypes denote physicians authorizations and the health record. PlanRT stereotype declares agents plan. In this example we consider only three plans: **Safe** for saving the record on the Data Provider; **Access** to read the record from the data Provider; **Authorize** for granting authorization to doctors to retrieve encrypted records from the Data Provider. Each Plan is associated to a Goal. It is reached when the final action of the plan is executed.

Similarly, Fig.9 shows the Agent Diagran for the Doctor, while Fig.10 depicts the diagram of the DataProvider.

Notice that Beliefs with the same name shares the same information among agents. For example, the *DoctorAuth* in Fig.9 is the same authorization belief of thePatient. The *Authorization* plan helps in sharing this information between the two agents.

In order to describe each plan behaviour, we must define proper Activity Diagrams that are compliant with the previously exposed modelling profile. We report here three plans. The first relates to thePatient's *Save* plan. Fig.11 depicts it.

Notice that plans describe behaviours of *proactive* and generally *interacting* agents. Hence, a plan usually involves synchronizations with plans of other agents. Synchronization is managed by using guards on control flow edges. Guards contains declaration of events

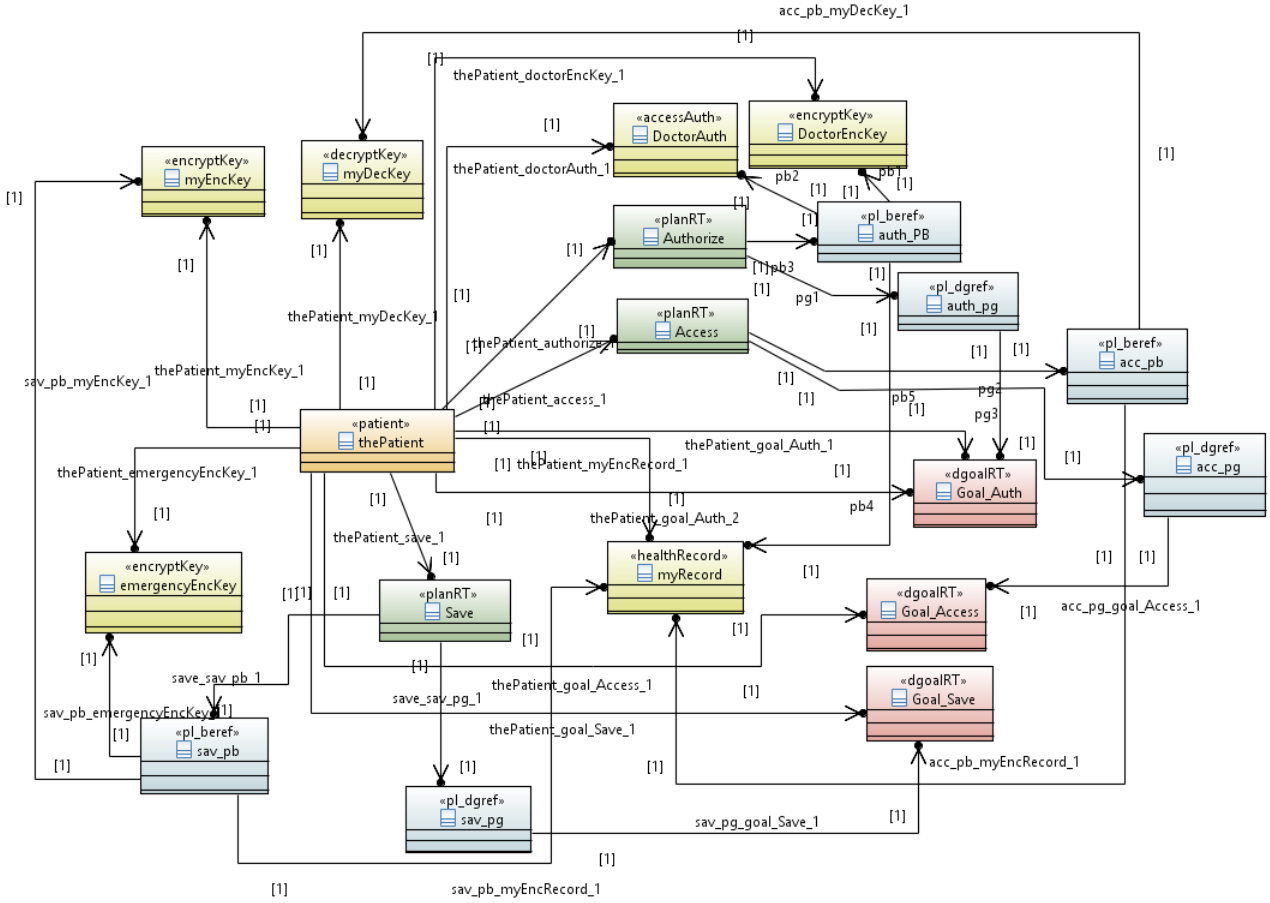

Figure 8: thePatient's Agent Diagram

asserted of awaited during plan execution. In particular, guards ending with exclamation marks are generated into the plan and received by other agents; guards ending with question marks defines that the control flow will continue only when other agents will generate the event with the same name. For example, this is the case of the awaited event *MyRecordSave?*. The control flow will carry on only when anothe agent (in particular the DataProvider) will assert the generation of the event *MyRecordSave!*. It is simple to understand that during the Save Plan, thePatient tries to save its records on the DataProvider. It begins three requests for saving in parallel, trying to save the record encrypted by its encryption key, the record encrypted by the key used for emergencies, and the record for the doctor, if it has been previously authorized to read the record on data Provider.

Fig.12 described the second plan.

It reports the activities of another thePatient's plan, the one related to the authorization of a Doctor. Notice that The object parameters on the edge of the activities reports the input object (on the left) and the output object (on the right) involved in the plan. Usually they

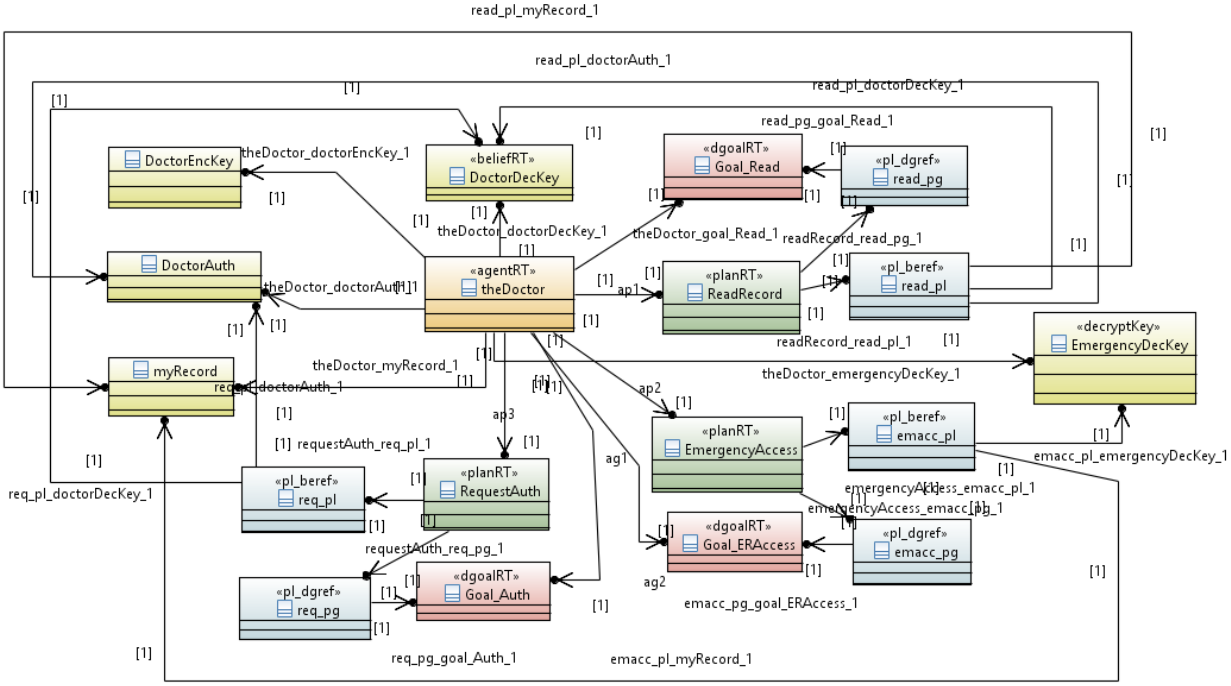

Figure 9: theDoctor's Agent Diagram

represent some Agents beliefs. In particular, in this activity, the Plan creates the Authorization for theDoctor and it stores the belief in the storage provider.

The third plan (Fig.13) shows the actions executed by theDoctor when it requests a normal authorization to read the health record. Notice that two final states are available here. The lower is related to the goal of the Read plan, the upper is a final state where the plan fails reaching its goal. This latter state is reached when theDoctor tries to read the records without proper authorization. On the other hand, if authorized, theDoctor accesses to the raw health record after decryption.

Finally, Sequence diagrams constitute a mean to orchestrate agents execution in use cases. They simply declares events that agents assert during their execution.

For example, Fig.14 shows the sequence related to the events exchanged between thePatient and DataProvider during a saving operation when theDoctor has the authorization to access to the health record.

At the end of structural and behavioural description of the system, Users must prepare an Observer Diagram to declare the type of the analyses and of monitoring activities they want to perform on the system. For Brevity's sake we do not report the figure with the Observer diagram used to enact privacy verification.

We have defined an *ad-hoc* Observer algorithm in order to verify privacy requirements on models compliant with the modelling profile previously described. It translates the design model into Timed Automata. The temporal stem in timed automata is unitary and no deadlines are reported on the automata. The algorithm implemented by the Observer is

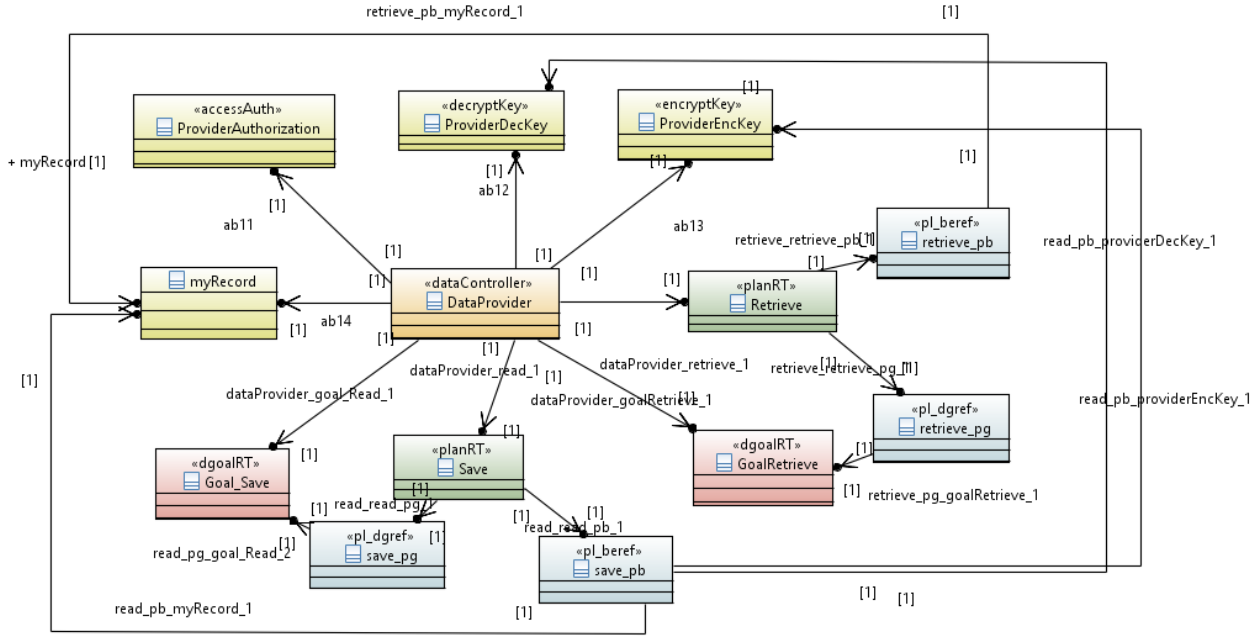

Figure 10: DataProvider Agent Diagram

described in the following:

for all $plan \in$ Agents Plans **do**
1.*Generate a Timed Automaton* $TA\_Pl[plan]$ *depending on Action States and Guards definition*
2.*Generate a Timed Automaton* $TA\_Bel[plan]$ *considering Input and Output Beliefs of* $plan$.
3.*find states in* $TA\_Pl[plan]$ *related to Actions that modify Beliefs in* $plan$
4.*create proper state transitions in* $TA\_Bel[plan]$ *depending on actions found at step* **3.**
end for
for all $seq \in$ Sequence Diagrams **do**
generate a Timed Automaton $TA\_Sq[seq]$ *for other TAs interactions;*
end for
Build the Product Automaton:
$PTA = \prod_{p \in Plans} TA\_Pl[p] \times \prod_{p \in Plans} TA\_Bel[p] \times \prod_{s \in Sequences} TA\_Sq[s]$;

The Observer enact an horizontal model transformation[14] translating an RT-AML model into a Timed Automaton. This allows for the use of tool for the analysis of Timed Automata in order to verify the needed privacy requirements.

The last thing to do is the translation of Privacy Requirements in a property analysable by a Timed Automata Model Checker (we use Uppaal[27] in order to analyse Timed Automata).

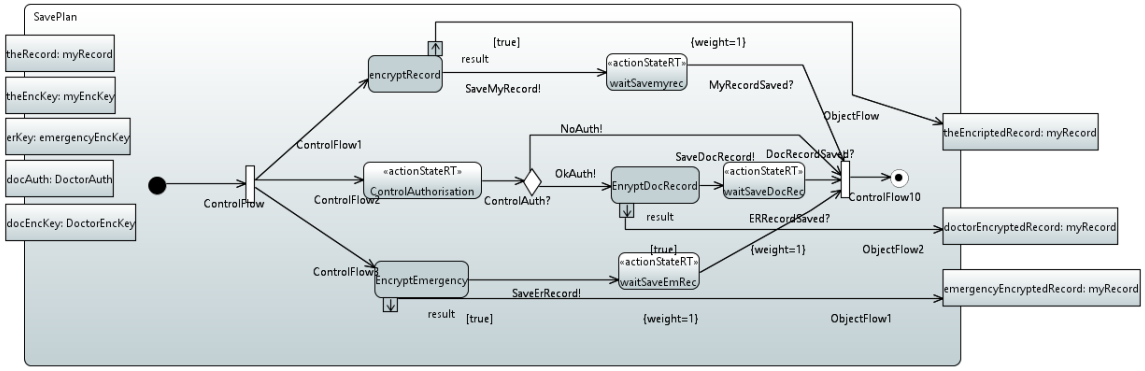

Figure 11: Save Plan

A way to express a privacy requirement is in terms of state reachability. In particular it is possible to assert that privacy is verified if the system grants doctors no access (i.e. Read_Goal is not reachable) to records except during emergencies or after being authorized by thePatient.

Since Uppaal supports Timed - Computational Tree Logics(CTL)[28] for queries specification and model checking, It is possible to translate privacy requirements into a CTL formula. CTL formulas contains both logical and temporal operators.

Logical Operators are the well known operators from propositional logics: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$, along with boolean constants *true* and *false*.

Temporal Operators quantifies over paths. Let ϕ and ψ two propositions, we consider here the following operators and their combination:

- Quantifiers over Paths:
  - **A** ϕ All: ϕ has to hold on all paths starting from the current state.
  - **E** ϕ Exists: there exists at least one path starting from the current state where ϕ holds.
- Path-Specific quantifiers:
  - **X** ϕ Next: ϕ has to hold at the next state (this operator is sometimes noted N instead of X).
  - **G** ϕ Globally: ϕ has to hold on the entire subsequent path.
  - **F** ϕ Finally: ϕ eventually has to hold (somewhere on the subsequent path).
  - ϕ **U** ψ Until: ϕ has to hold at least until at some position ψ holds. This implies that ψ will be verified in the future.

Valid CTL formulas contains only Path Quantifiers coupled with Path-specific quantifiers (for example: **AG**ψ,**EF**ψ etc.)

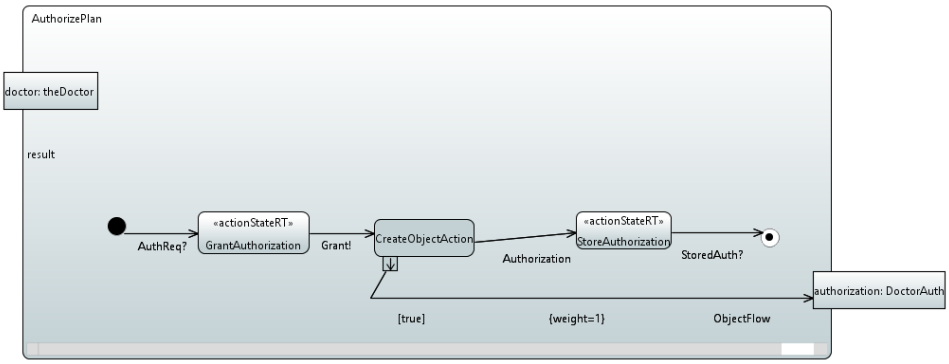

Figure 12: Authorize Plan

Furthermore, the privacy Observer maintains traceability with the RT-AML model. This means that states in the translated model (i.e. the product timed automaton) maintains the same name of the action states and of the beliefs in the original model. The Observer is able to identify the states of the translated timed automata by using a *dotted notation*.

theDoctor.ReadRecord.ReadEncRecord identifies the state $ReadEncRecord$ in Fig.13, *theDoctor.EmergencyAccess.initialActivity* addresses the initial activity of the $EmergencyAccess$ plan etc. Beliefs are identified by the value they assume. For example, the proposition *theDoctor.DoctorAuth == true* is true in each state where authorization is granted to the doctor.

With this notation, it is possible to verify CTL formulas by means of the Observer. For example:

$$theDoctor.EmergencyAccess.initialActivity \Rightarrow EGtheDoctor.Read\_Goal$$

Is true if in the case of Emergency, in any case the $Read\_Goal$ is reachable and hence the doctor can, in any case, read the record in case of emergency.

The Observer checked this formula and it assure that our model satisfies this formula.

Privacy properties can be verified for example, by checking the following two formulas:

$$((AG\neg theDoctor.EmergencyAccess.initialActivity)$$

$$\wedge(AG\neg theDoctor.DoctorAuth == true)) \Rightarrow AG\neg theDoctor.Read\_Goal$$

or

$$AG(\neg theDoctor.Read\_Goal\mathbf{U}$$

$$(theDoctor.EmergencyAccess.initialActivity \vee theDoctor.DoctorAuth == true))$$

The formulas are quite different. The first one asserts that If there *never* is an emergency and if theDoctor is *never* authorized then the record cannot be *ever* red. This seems to be a good way to define privacy in this model. Anyway, the second formula assert that in any case, the record cannot be red by theDoctor until an emergency occurs or the doctor is authorized. This second formula is the right formula to verify privacy in our model. Anyway the Observer assures that the model satisties both formulas.

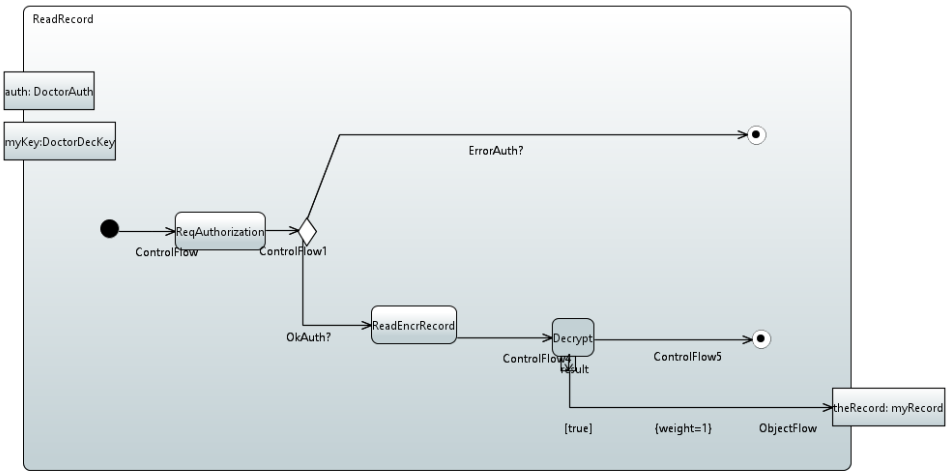

Figure 13: theDoctor's Read Plan

6 State of the art

The use of MDE techniques in modelling and verification of complex and critical systems is growing more and more.

Usually UML could be considered as the standard modelling language to use, but it is not the best tool for modelling agent based systems[29]. The problem in standard UML-based languages for multi-agent systems, is that agents pro-activity is hard to describe. This is why several languages for Agent Based models have been developed like Agent-UML [30], Agent Modeling Language (AML)[31] (a semi-formal visual modelling language based on the UML 2.0 superstructure) or the one defined for the Prometheus methodology [32]. The Prometheus methodology has been coped with INGENIAS[33]: it consists of a meta-model describing MAS from different perspectives like environment, goals and tasks, specifying the behavior of each agent. Anyway INGENIAS focuses only on the generation of executable software and no verification issues are addressed especially for critical systems.

A tool that uses an approach similar to MetaMORP(h)OSY is SCADE[34]. It enacts model checking to verify properties at design time and implements them onto embedded systems. It is nowadays at a mature implementation stage, but unlike MetaMORP(h)OSY, it implements only verification of state reachability properties (by mean of Model Checking) and it focuses on embedded and control system. Model transformation focused on a single domain is common for MDE tools and methodologies. For example, another framework which focuses on the generation of executable embedded code in TOPCASED [35].

The MetaMORP(h)OSY modelling methodology is oriented towards requirements verification. Differently from other proposed approaches, it uses both vertical and horizontal transformation in order to follow requirement verification during all life cycle of the system. The design model (DSML) is based on UML and it is an extension of AML with time support. It also allows to specify properties to evaluate by means of definition of proper Observers. Model Transformation algorithms can be coped with different Observers by including proper plugins in the framework. Model transformations are used both for the

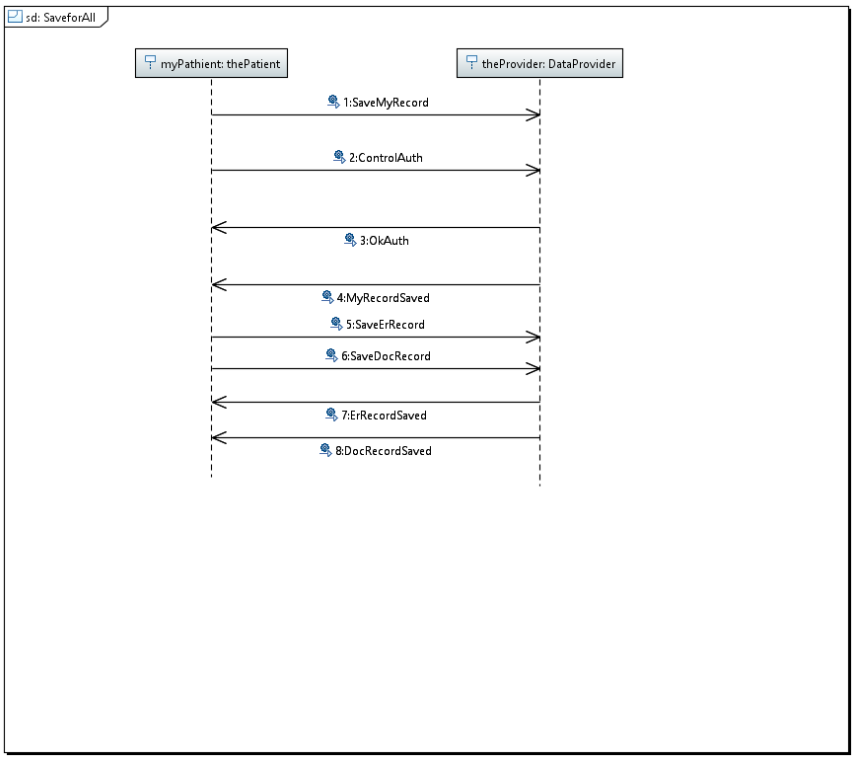

Figure 14: Save Sequence

analyses of design models, and for the implementation of run-time components. Hence MetaMORP(h)OSY is a more versatile framework and at the state of the art it has been used both for verification of properties and generation of running embedded systems (like in this work), and also to provide support for design and provisions of cloud services[13, 36]. It also supports different verification techniques (and new ones can be plugged in) including multi-formalisms and multi-solution approaches [11].

For what Data Privacy verification concerns, recent years have seen many proposals that incorporate security in the software engineering process. Approaches such as [37, 38, 39] use existing modelling and analysis framework such as UML or Tropos in order to model in that framework security requirements. The main differences with MeraMORP(h)OSY are in the way requirements are modelled: while our framework uses the same (UML based) formalism in order to model requirements and properties too, These approaches require a different language for property specification. In addition, no model transformation is enacted at all.

Other approaches [40, 41, 42, 43] adopt a requirements engineering framework and enhance it with novel constructs specific to security. For such approaches, formal analysis techniques and implementation guidelines need to be revised and/or extended to accommodate the new concepts. No Model Transformation approaches have been introduced here and again properties and requirements are difficult to define since they require the use of DSML languages.

Finally, notice that the need for a methodology for modelling (possibly in a formal way) Legal Requirements, is discussed in [42, 44]. Some attempt to apply Model Driven techniques to privacy problems in E-Health have been made recently [45]. Anyway, this work is one of the first in the best of our knowledge which copes Legal Requirements modelling with MDE methodologies and techniques for E-Health systems.

7 Conclusions

In this paper we have extet the MetaMORP(h)OSY modelling profile in order to explicitly consider privacy requirements for data. A novel algorithm is described for implementing a horizontal model transformation for the application of Model Checking techniques to privacy verification and to express privacy in terms of a stete reachability problem following an MDE approach.

As future works we are considering the introduction of further model transformation algorithms to support more complicated data privacy requirements and models. For example, the approach reported in [46] uses models based on Bayesan Models. Since MetaMORP(h)OSY already embeds Bayesan Observers, we will try to enhance our modelling and verification techniques to include more and more privacy models.

References

[1] Hermann Kopetz. Internet of things. In *Real-Time Systems*, pages 307–323. Springer, 2011.

[2] Douglas C Schmidt. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25, 2006.

[3] Richard Soley et al. Model driven architecture. *OMG white paper*, 308:308, 2000.

[4] Michael Wooldridge. Agent-based software engineering. In *IEE Proceedings on Software Engineering*, pages 26–37, 1997.

[5] Francesco Moscato, Salvatore Venticinque, Rocco Aversa, and Beniamino Di Martino. Formal modeling and verification of real-time multi-agent systems: The remm framework. *Studies in Computational Intelligence*, 162:187–196, 2008.

[6] Papyrus uml: http://www.papyrusuml.org.

[7] Madeleine Faugere, Thimothee Bourbeau, Robert de Simone, and Sebastien Gerard. Marte: Also an uml profile for modeling aadl applications. *Engineering of Complex Computer Systems, IEEE International Conference on*, 0:359–364, 2007.

[8] Giuliana Franceschinis, Marco Gribaudo, Mauro Iacono, Stefano Marrone, Francesco Moscato, and Valeria Vittorini. Interfaces and binding in component based development of formal models. In *IEEE proc. of VALUETOOLS 09 conference*, page 44, 2009.

[9] Giusy Di Lorenzo, Nicola Mazzocca, Francesco Moscato, and Valeria Vittorini. Towards semantics driven generation of executable web services compositions. *International Journal of Software, JSW*, 2(5):1–15, 2007.

[10] Giusy Di Lorenzo, Francesco Moscato, Nicola Mazzocca, and Valeria Vittorini. Automatic analysis of control flow in web services composition processes. In *PDP*, pages 299–306, 2007.

[11] Francesco Moscato, Valeria Vittorini, Flora Amato, Antonino Mazzeo, and Nicola Mazzocca. Solution workflows for model-based analysis of complex systems. *IEEE T. Automation Science and Engineering*, 9(1):83–95, 2012.

[12] Francesco Moscato, Rocco Aversa, and Alba Amato. Describing cloud use case in metamorp(h)osy. *Proceedings - 2012 6th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2012*, pages 793–798, 2012.

[13] Francesco Moscato, Beniamino Di Martino, and Rocco Aversa. Enabling model driven engineering of cloud services by using mosaic ontology. *Scalable Computing: Practice and Experience*, 13(1), 2012.

[14] Tom Mens and Pieter Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152(0):125 – 142, 2006. Proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005), Graph and Model Transformation 2005.

[15] Robert H Dolin, Liora Alschuler, Sandy Boyer, Calvin Beebe, Fred M Behlen, Paul V Biron, and Amnon Shabo Shvo. Hl7 clinical document architecture, release 2. *Journal of the American Medical Informatics Association*, 13(1):30–39, 2006.

[16] Dipak Kalra. Electronic health record standards. 2006.

[17] European Commission. Article 29 Working Party. `http://ec.europa.eu/justice/data-protection/article-29/index_en.htm`, 2013. [Online; Last update: 06/08/2013].

[18] Flora Amato, Antonino Mazzeo, Antonio Penta, and Antonio Picariello. Knowledge representation and management for e-government documents. *IFIP International Federation for Information Processing*, 280:31–40, 2008.

[19] Francesco Moscato, Rocco Aversa, and Alba Amato. Describing cloud use case in metamorp(h)osy. In *IEEE Proc. of CISIS 2012 conference*, pages 793–798, 2012.

[20] Flora Amato, Valentina Casola, Nicola Mazzocca, and Sara Romano. A semantic approach for fine-grain access control of e-health documents. *Logic Journal of IGPL*, 21(4):692–701, 2013.

[21] Flora Amato, Antonino Mazzeo, Vincenzo Moscato, and Antonio Picariello. Exploiting cloud technologies and context information for recommending touristic paths. *Studies in Computational Intelligence*, 511:281–287, 2014.

[22] Anneke G Kleppe, Jos Warmer, Wim Bast, and MDA Explained. The model driven architecture: practice and promise, 2003.

[23] Radovan Červenka, Ivan Trenčanskỳ, Monique Calisti, and Dominic Greenwood. Aml: Agent modeling language toward industry-grade agent-based modeling. In *Agent-Oriented Software Engineering V*, pages 31–46. Springer, 2005.

[24] Francesco Moscato, Salvatore Venticinque, Rocco Aversa, and Beniamino Di Martino. Formal modeling and verification of real-time multi-agent systems: The remm framework. In Costin Badica, Giuseppe Mangioni, Vincenza Carchiolo, and Dumitru Burdescu, editors, *Intelligent Distributed Computing, Systems and Applications*, volume 162 of *Studies in Computational Intelligence*, pages 187–196. Springer Berlin / Heidelberg, 2008.

[25] Francesco Moscato. Model driven engineering and verification of composite cloud services in metamorp(h)osy. In *Intelligent Networking and Collaborative Systems (INCoS), 2014 International Conference on*, pages 635–640, Sept 2014.

[26] Rocco Aversa, Beniamino Di Martino, and Francesco Moscato. Critical systems verification in metamorp (h) osy. In *Computer Safety, Reliability, and Security*, pages 119–129. Springer, 2014.

[27] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. Developing uppaal over 15 years. *Software: Practice and Experience*, 41(2):133–142, 2011.

[28] Thilo Hafer and Wolfgang Thomas. Computation tree logic ctl* and path quantifiers in the monadic theory of the binary tree. In *Automata, Languages and Programming*, pages 269–279. Springer, 1987.

[29] Bernhard Bauer. Uml class diagrams revisited in the context of agent-based systems. In *Agent-Oriented Software Engineering II*, pages 101–118. Springer, 2002.

[30] Bernhard Bauer, Jrg P. Mller, and James Odell. Agent uml: A formalism for specifying multia-

gent software systems. *Int. Journal of Software Engineering and Knowledge Engineering*, 11:91–103, 2000.

[31] Ivan Trencansky and Radovan Cervenka. Agent modeling language (aml): A comprehensive approach to modeling mas. *Whitestein Series in Software Agent Technologies and Autonomic Computing*, 29:391–400, 2005.

[32] Jos M. Gascuea, Elena Navarro, and Antonio Fernndez-Caballero. Model-driven engineering techniques for the development of multi-agent systems. *Engineering Applications of Artificial Intelligence*, 25(1):159 – 173, 2012.

[33] Antonio Fernndez-Caballero and JosM. Gascuea. Developing multi-agent systems through integrating prometheus, ingenias and icaro-t. In Joaquim Filipe, Ana Fred, and Bernadette Sharp, editors, *Agents and Artificial Intelligence*, volume 67 of *Communications in Computer and Information Science*, pages 219–232. Springer Berlin Heidelberg, 2010.

[34] ParoshAziz Abdulla, Johann Deneux, Gunnar Stalmarck, Herman Agren, and Ove Akerlund. Designing safe, reliable systems using scade. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods*, volume 4313 of *Lecture Notes in Computer Science*, pages 115–129. Springer Berlin Heidelberg, 2006.

[35] J. Farines, M.H. De Queiroz, V.G. da Rocha, A.M.M. Carpes, F. Vernadat, and X. Cregut. A model-driven engineering approach to formal verification of plc programs. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–8, Sept 2011.

[36] Francesco Moscato, Rocco Aversa, Beniamino Di Martino, Teodor-Florin Fortis, and Victor Ion Munteanu. An analysis of mosaic ontology for cloud resources annotation. In *IEEE Proc. of FedCSIS 2011 Conference*, pages 973–980, 2011.

[37] Annie I Antón, Julia Brande Earp, and Angela Reese. Analyzing website privacy requirements using a privacy goal taxonomy. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 23–31. IEEE, 2002.

[38] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[39] Ariel Fuxman, Marco Pistore, John Mylopoulos, and Paolo Traverso. Model checking early requirements specifications in tropos. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 174–181. IEEE, 2001.

[40] Rune Fredriksen, Monica Kristiansen, Bjørn Axel Gran, Ketil Stølen, Tom Arthur Opperud, and Theo Dimitrakos. The coras framework for a model-based risk management process. In *Computer Safety, Reliability and Security*, pages 94–105. Springer, 2002.

[41] Torsten Lodderstedt, David Basin, and Jürgen Doser. Secureuml: A uml-based modeling language for model-driven security. In *UML 2002The Unified Modeling Language*, pages 426–441. Springer, 2002.

[42] Fabio Massacci, Marco Prest, and Nicola Zannone. Using a security requirements engineering methodology in practice: the compliance with the italian data protection legislation. *Computer Standards & Interfaces*, 27(5):445–455, 2005.

[43] Axel Van Lamsweerde, Simon Brohez, Renaud De Landtsheer, and David Janssens. From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering. *Proc. of RHAS*, 3:49–56, 2003.

[44] Ambrosio Toval, Alfonso Olmos, and Mario Piattini. Legal requirements reuse: a critical success factor for requirements quality and personal data protection. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 95–103. IEEE, 2002.

[45] Yun Ding and K. Klein. Model-driven application-level encryption for the privacy of e-health data. In *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*, pages 341–346, Feb 2010.

[46] Jinfei Liu, Li Xiong, and Jun Luo. Semantic security: Privacy definitions revisited. *Transactions on Data Privacy*, 6(3):185–198, 2013.