A Security and Privacy Validation Methodology for e-Health Systems

FLORA AMATO, VALENTINA CASOLA, GIOVANNI COZZOLINO, ALESSANDRA DE BENEDICTIS, and NICOLA MAZZOCCA, University of Naples Federico II
FRANCESCO MOSCATO, University of Campania Luigi Vanvitelli

e-Health applications enable one to acquire, process, and share patient medical data to improve diagnosis, treatment, and patient monitoring. Despite the undeniable benefits brought by the digitization of health systems, the transmission of and access to medical information raises critical issues, mainly related to security and privacy. While several security mechanisms exist that can be applied in an e-Health system, they may not be adequate due to the complexity of involved workflows, and to the possible inherent correlation among health-related concepts that may be exploited by unauthorized subjects. In this article, we propose a novel methodology for the validation of security and privacy policies in a complex e-Health system, that leverages a formal description of clinical workflows and a semantically enriched definition of the data model used by the workflows, in order to build a comprehensive model of the system that can be analyzed with automated model checking and ontology-based reasoning techniques. To validate the proposed methodology, we applied it to two case studies, subjected to the directives of the EU GDPR regulation for the protection of health data, and demonstrated its ability to correctly verify the fulfillment of desired policies in different scenarios.

CCS Concepts: • **Computing methodologies** → *Distributed computing methodologies*; • **Theory of computation** → *Automated reasoning*; *Finite Model Theory*; *Verification by model checking*; • **Security and privacy** → *Software and application security*; *Domain-specific security and privacy architectures*;

Additional Key Words and Phrases: e-Health management systems, security and privacy for e-Health data, security and privacy validation, formal methods for security validation

ACM Reference format:
Flora Amato, Valentina Casola, Giovanni Cozzolino, Alessandra De Benedictis, Nicola Mazzocca, and Francesco Moscato. 2021. A Security and Privacy Validation Methodology for e-Health Systems. *ACM Trans. Multimedia Comput. Commun. Appl.* 17, 2s, Article 67 (May 2021), 22 pages.
https://doi.org/10.1145/3412373

1 INTRODUCTION AND MOTIVATION

e-Health is a relatively recent practice in the healthcare domain that relies upon information and communication technologies for improving the management of medical information. Current

Authors' addresses: F. Amato, V. Casola, G. Cozzolino, A. De Benedictis, and N. Mazzocca, Department of Electrical Engineering and Information Technology, University of Naples Federico II, Via Claudio 21, 80125 Naples (NA), Italy; emails: {flora.amato, casolav, giovanni.cozzolino, alessandra.debenedictis, nicola.mazzocca}@unina.it; F. Moscato, Department of Information Technology and Electrical Engineering and Applied Mathematics, University of Salerno, Via Giovanni Paolo II, 132 - 84084 Fisciano (SA), Italy; email: francesco.moscato@unicampania.it.

1551-6857/2021/05-ART67 $15.00
https://doi.org/10.1145/3412373

e-Health applications and services enable one to acquire, process, and share patient medical data among healthcare professionals, in order to enhance diagnosis, treatment, and patient monitoring.

With the rapid development of Cloud and **Internet of Things** (**IoT**) technologies, e-Health services are taking advantage of more efficient solutions to collect biometric parameters of patients and to store and elaborate data [13, 22]. Despite the undeniable benefits brought by the digitization of health systems, the transmission of and access to medical information raises critical issues, mainly related to security and privacy. In fact, medical information (also called *health records*) includes patient personal data (e.g., social security number, personal telephone number), medical histories, diagnoses, medications, immunization status, laboratory test reports, and so forth, which are in great part sensitive and confidential. This data is protected by international laws and regulations, like the **Health Insurance Portability and Accountability Act** (**HIPAA**), which regulates the use and disclosure of **Protected Health Information** (**PHI**) by the organizations that manage patients' health data in the US, and the recent European **General Data Protection Regulation** (**GDPR**) [12], which has a wider scope and applies more in general to the protection of any sensitive personal data (including racial or ethnic origin and religion) belonging to EU citizens and residents.

As outlined by [16], the healthcare sector has been targeted by a growing number of data breach attacks over the last decades. Nevertheless, as outlined in [8], although several security mechanisms exist that can be applied to thwart most common cyber attacks even in the e-Health domain (e.g., [9, 20, 27]), they may be not adequate to ensure the needed level of security and privacy due to several factors.

Indeed, modern e-Health systems are highly decentralized and distributed, and typically integrate different subsystems (often legacy) by orchestrating their services and by suitably merging their data to build complex *clinical workflow processes*. A clinical workflow is a sequence of steps comprising a healthcare process, which may either refer to direct patient care actions (like the creation of nursing care plans or the review of laboratory results) or to other tasks aimed at supporting the care delivery (like the administrative and financial processes). A workflow typically involves different actors, such as doctors, nurses, administrative staff, technicians, patients, and different associated digital services (medical record management, hospitalization records, etc.), which are usually orchestrated via a workflow engine. Each step of the workflow involves a subset of these actors, and may require one to collect, access, process, and transfer a subset of the patients' sensitive personal data. They also contain complex *dataflows*, which identify what data is involved in each workflow step. In this regard, it is worth noting that the data processed by a workflow may be represented by both structured information (stored in suitable databases used for the workflow execution) and unstructured information (e.g., paper-based medical records, drug leaflets), available in separate documents. Being aware of the actual dataflow involving sensitive personal information is of paramount importance for ensuring compliance with regulations, since it helps organizations identify the data they hold and possibly detect unforeseen or unintended uses of data, or unnecessary data transfers.

Unfortunately, due to the high degree of heterogeneity of involved subsystems and to the issues related to their integration, ensuring that the needed security and privacy controls are correctly in place and that they are effective in securing the whole dataflow of sensitive information is quite a difficult task. The problem is exacerbated by the existence of inherent correlations among data, that would make it possible to leverage secondary, non-sensitive information in itself (even collected in an offline process), to infer private information, as widely demonstrated in the Social Media Analysis and Internet of Data literature (e.g., [11, 25]). From the observation made above, it is pretty clear that an effective policy validation strategy should take into consideration several factors, including the structure of the clinical workflow(s) established within the system, the definition of

the related dataflows, and a clear specification of all involved information and of their inherent correlations, in addition, of course, to an unambiguous specification of the applicable policies and some context information (e.g., the purpose of data processing). To the best of our knowledge, at the state of the art, such a comprehensive validation methodology has not yet been proposed, while there are several research studies that address separately the aspects of workflow modeling, information modeling, and policy/laws modeling and validation (in static contexts).

Hence, our contribution is the definition of a novel methodology for the validation of security and privacy policies in a complex e-Health system, that leverages

- a *formal definition of the involved clinical workflows and of the related dataflows*, specified by means of the **Business Process Model and Notation** (**BPMN**) language;
- an *ontology-based data model* of the system, able to represent both the concepts derived from the structured information used by the workflow processes, and the unstructured information extracted from clinical documents, and to describe the relationships existing among the concepts related to the health domain and the application of system policies; and
- an *automatic reasoning* over the system model in order to validate security and privacy policies.

As discussed in detail in the following sections, the workflow and data models mentioned above are suitably combined to generate a comprehensive model of the system under analysis via model transformation techniques. Afterward, the composite model is analyzed by means of a combination of model checking techniques and ontology-based reasoning techniques, which enable one to verify, in an automated way, whether the input policies are respected or not.

To validate the proposed methodology, we applied it to a case study represented by an example medical care process of a hospital, involving a simple workflow. With regard to policies, we considered the directives of the EU GDPR regulation related to the protection of health data, which state that any data regarding the past, present, or future health status of a subject is to be considered as personal data and must be protected from unauthorized or unlawful processing. We analyzed the same scenario (i.e., an actor attempting to access data for which he/she is not authorized) in the presence of different global conditions (i.e., different processing purposes), suitably modeled by the comprehensive data model of the system, and demonstrated the ability of our methodology to correctly verify, in both cases, the fulfillment of desired privacy policies.

The article is organized as follows. In Section 2, we illustrate some relevant related work on security policies validation and on different approaches for validation of complex distributed systems with semantic and composition approaches. In Section 3, we illustrate the open challenges related to data protection in modern e-Health systems by discussing an example clinical workflow, which will be used as a reference to validate the proposed methodology. Section 4 illustrates the main steps of the proposed methodology, which is based on model transformation and semantic-based reasoning. Section 5 discusses the application of the methodology to a case study and provides some details on the modeling and validation steps for some privacy rules. Finally, Section 6 presents some concluding remarks and a discussion about future works.

2 RELATED WORK

A large number of security controls and mechanisms are already being used in smart health systems to prevent and detect unauthorized access to health data, nevertheless, existing privacy-preserving mechanisms are often not adequate and many security challenges are still open [8]. In this context, as already recognized in [10], a more formal approach to privacy and security analysis would help identify the deficiencies within an organization's current policies management. In particular, the author proposed a general question-based privacy model aimed at assessing the

soundness and completeness of enforced policies with the goal of minimizing the risks, but, while the general approach is useful and interesting, an automation of the assessment process is needed, especially to manage security in complex and composite systems.

In the last decade, several research works have focused on security policy analysis and verification by means of formal methods and model checking techniques. Most of the existing work is related to the verification of policy consistency and correctness [19, 21], while fewer proposals have addressed the formal verification of the compliance of a system to established policies, which is the main focus of this article. Among the papers that address this aspect, most deal with access control policies, with particular reference to **role-based access control policies** (**RBAC**) and their extensions. As an example, in [24], the authors model a **Generalized Temporal RBAC** (**GTRBAC**) set of policies as a timed automaton, and use a model checker to verify that the set of policies satisfies safety and liveness properties; in [14], the authors define a model-checking technique based on the NIST generic model-checking technique [17] to be used for the verification of multi-domain cloud policies, enriched with RBAC reasoning; the authors of [18] provide a general approach to formally specify access control models and safety requirements with the adoption of model checkers to verify the integrity, coverage, and confinement of access policies. Finally, more recently, the authors of [5] have proposed an approach that uses the **Hierarchical Timed Colored Petri Nets** (**HTCPN**) formalism to model and analyze Temporal RBAC policies.

Among the works specifically focused on the verification of privacy policies, it is possible to mention [15], whose authors propose a model and a language for specifying privacy policies in data-intensive applications based on **Metric First Order Temporal Logic** (**MFOTL**), leveraged in [6] for the runtime monitoring of system properties. The approach described in this work is very interesting, but it is based on the monitoring of log data collected by a system related to relevant events, and is not suited to distributed environments where several subsystems process information extracted from multiple heterogeneous sources.

The scientific literature proposes many solutions to the problem of analyzing and verifying privacy rules and policies in composite, workflow processes. The work described in [4] shows the effectiveness of the use of event-driven models, of domain-specific languages (coping with BPMN orchestrated services), and of an analysis engine for events that is able to monitor the enforcement of local and global security policies. We differ from this approach because we provide a formal methodology for the analysis and verification of privacy policies in BPMN processes, which is based on a model transformation technique and semantic-based enforcement of model checking. We generalize policies and requirements by using ontology-based languages, thus we are able to also model rules from other domains, as government laws. Another approach related to our proposal is in [26]. There, the authors use event-based automata for validation. We use a graph-based approach too, but we are able to cope with complexity by merging features from domains of model checking and ontology-based reasoning.

A first attempt toward the definition of a comprehensive methodology for the verification of security policies in complex workflow-based applications was done in [3], where we proposed a methodology relying upon workflow languages and semantics to validate the security policies enforced in an IoT system and verify whether they match with global end-user policies and national and international laws and rules. In this article, we further extended that methodology and applied it to the very complex e-Health scenario, where several factors must be taken into account (the structure of the workflow(s) established within the system, the definition of the related dataflows, and a clear specification of all involved information and of their inherent correlations, in addition, of course, to an unambiguous specification of the applicable policies and other context information).

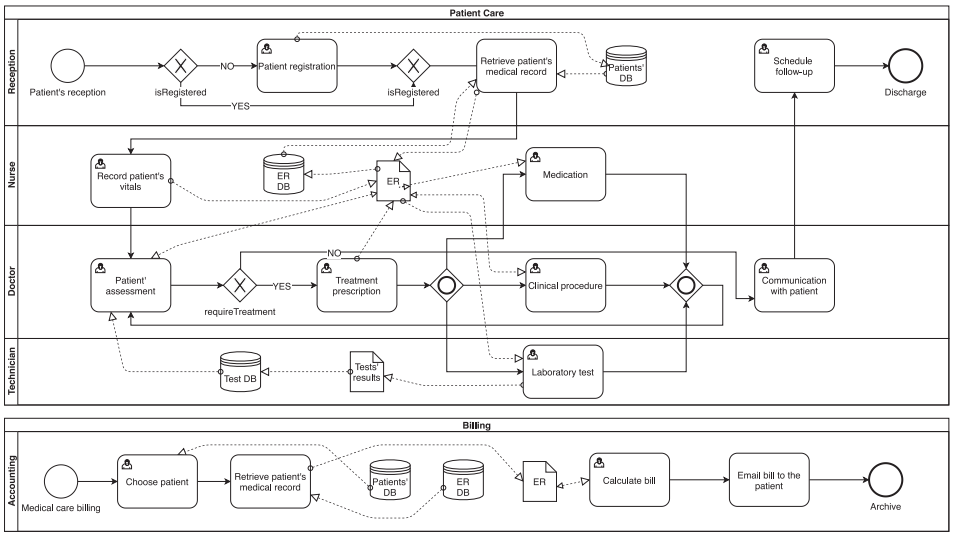

Fig. 1. Workflow diagram of the clinical running example.

3 THE DATAFLOW IN A REFERENCE E-HEALTH SYSTEM

Nowadays, a healthcare information system is composed of a set of services aimed primarily at automating interactions with patients. The system records and processes patients' personal data (e.g., name, home address, date of birth) both for administrative and for medical purposes. In order to execute a clinical workflow, the single workflow activities are assigned to specific resources, which can be both human actors and software applications, performing automated actions. In many cases, it is useful to classify resources according to the roles they perform in the process, so as to define the assignment of permissions and tasks based on roles. Thanks to the specification of the activities constituting the process and of their temporal sequence, and thanks to the classification of the resources responsible for carrying out these activities, it is possible to control the processes and adapt them promptly to changes in the surrounding context, such as law innovations or the introduction of new clinical protocols.

To provide a concrete example of a clinical workflow, let us refer to the workflow instance depicted in Figure 1, which models a simplified medical care process of a hospital or a surgery. The model, which will be used as a running example throughout the article, is designed by using the BPMN formalism, a standard for business process modeling that provides a graphical notation for specifying business processes. Tasks reported in the process follow patients management in the hospital from *initial registration* to *health assessment*, *treatment identification*, *therapy definition*, and *follow-up*. Tasks are executed by different actors with different roles (receptionists at administration offices, doctors, laboratory technicians, etc.), who may only have access to specific parts of shared patient medical records. In particular, we can identify the following five main actors:

— *Administration*: performs all the administrative tasks, i.e., the entry of the patient's personal data and the scheduling of follow-up appointments.
— *Nurse*: mainly records the patients' vital parameters and dispenses drugs according to the therapy.
— *Doctor*: prescribes clinical treatments, according to patients' clinical status. A doctor can also be involved in the execution of a clinical procedure and, finally, communicates the results of the therapy and the clinical overview to the patient.

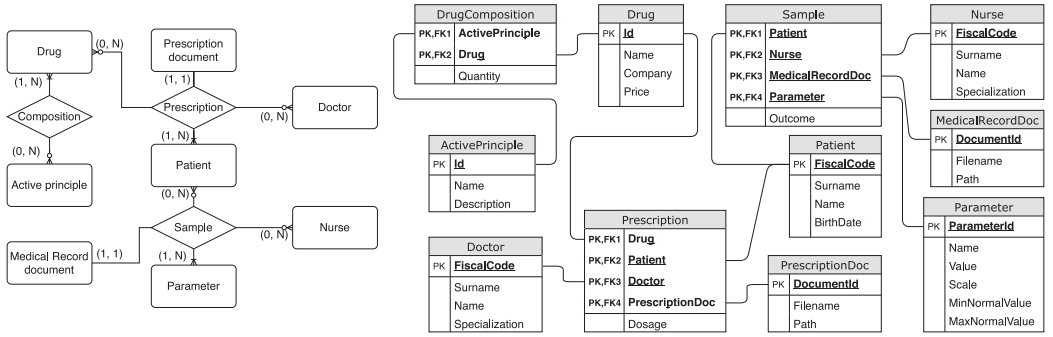

Fig. 2. Data model used by the running example.

— *Technician*: performs diagnostic tests or a clinical examination.
— *Accounting*: dispatches billing data.

In the figure, tasks are represented as rectangles and are connected with one another by solid connections arrows. All the tasks belonging to the same layer in the model are associated to one of the actors, so the workflow engine automatically assigns the current task to the corresponding role. Moreover, the identity of the user that performs a task is verified before granting the access to data. It is worth noting that Figure 1 also shows the dataflow within the execution of the process instance. In particular, we can observe the presence of three main data sources, which are kept distinct in order to ensure the separation of sensitive data according to the basic privacy recommendations, also prescribed by the GDPR:

(1) Patient Personal Data (*Patient DB*): stores information for administration tasks (i.e., personal data of the patients, information related to clinical services, appointment scheduling or bills).
(2) Electronics Records (*ER DB*): stores the medical data related to the patients, including diagnosis and therapies.
(3) Examinations (*Test DB*): stores the result of diagnostics and laboratory tests (blood analysis, radiography images, etc.).

The main data object of the process instance is the **Electronic Record** (**ER**), that is created during patients' reception and is initialized with the patients' identification numbers, related to their personal data. The ER is updated with information related to the vital parameters by the Nurse actor and, then, it is analyzed by Doctors in order to evaluate further actions. Doctors can also update the ER by adding prescriptions, and can access the Examination repository results. Technicians can read the ER to know which examination has to be performed. Finally, the Accounting actor can read the ER to execute billing and scheduling activities.

In Figure 2, we show a simplified conceptual data model of an ER along with the relational scheme that enables its implementation in a relational database, to be used by the workflow activities.

3.1 Privacy Concerns

The schema in Figure 2 has two entities, namely, *Patient* and *Doctor*, who both include sensitive information referring to identifiable, living subjects (e.g., the fiscal code). For this reason, the storage and retrieval of these data is subject to privacy protection rules (like those established by GDPR). In particular, the GDPR prescribes the *pseudonymization* of data, which consists in making personal

and sensitive data not easily associable, by means of encryption techniques and by substituting identifying fields with artificial identifiers, or pseudonyms.

However, as shown in the model in Figure 2, the relationship *Prescription* joins all the information related to an electronic health record, combining data coming from patients, doctors, and drugs. We can observe also the entity *PrescriptionDocument*, that is responsible for storing information related to the prescription, such as the dosage, the route of administration, and so forth. This document can store relevant sensitive data that can reveal information about the clinical status of the patient. Moreover, we can observe that there exist some correlations among domain entities that enable users to reconstruct sensitive information without breaking privacy policies enforced into the system. For example, in our model there is a correlation between patients, their prescriptions, the drugs chosen for the therapy, and the diseases that each drug can cure, making it possible to infer the diseases/pathologies suffered by patients by analyzing drug prescriptions. It is worth outlining that such kind of correlations may also be made by exploiting external resources (e.g., the drug leaflets in the considered situation), thus making it impossible to detect violations. Hence, the adoption of pseudonymization techniques within the single entities of the data model is not enough to ensure the compliance to privacy policies, while the anonymization should be enforced across the whole patient management workflow. For example, in our reference model, pathologies should not be associated with the name and surname of patients, while the data should be split into different tables having correspondence keys that cannot be associated with any subject without knowing the right access key: only the patient or his/her attending doctor have the privilege to access this data, while for system administrators, administration users, or technicians, it should be forbidden.

4 A VALIDATION METHODOLOGY

This section introduces the methodology that enables the identification of possible violations to privacy-related policies or data-breaches in a complex, distributed environment. As already said, in order to illustrate the approach, we apply the methodology to an e-Health environment, but it is general enough to be applied in different domains.

As discussed in the previous sections, a workflow is a sequence of activities, typically involving different subsystems and services, which are carried out by different actors and that entail accessing, processing, and exchanging different information. Usually, in workflow-based systems, policies are defined in a static way, by simply considering the information organized according to the data model used for the workflow execution, while, as discussed in the previous sections, a more dynamic point of view should be considered, able to take into account several factors, including the inherent correlation among underlying information and the convergence among different workflow processes, plus the impact of dynamic context conditions.

In order to overcome the limitations of current approaches, we propose a novel methodology that relies upon a formal definition of involved workflow processes and an enriched ontology-based workflow data model, in order to build a comprehensive model of the system, specified as an automaton, on which it is possible to perform automated reasoning. Figure 3 shows the main elements exploited by our methodology, summarized in the following.

A **BPMN Process** block takes care of the formal definition of the workflow, which is modeled in BPMN language. As illustrated in Section 3, a workflow model includes the workflow activities, the profile of (human and software) actors that execute the activities, and the dataflow, namely, the specification of what information is used by each activity and how it is routed throughout the workflow.

The modeling of the data used by the workflow is up to the **Data Model** block, which includes both structured and unstructured information: the former is modeled by means of a relational

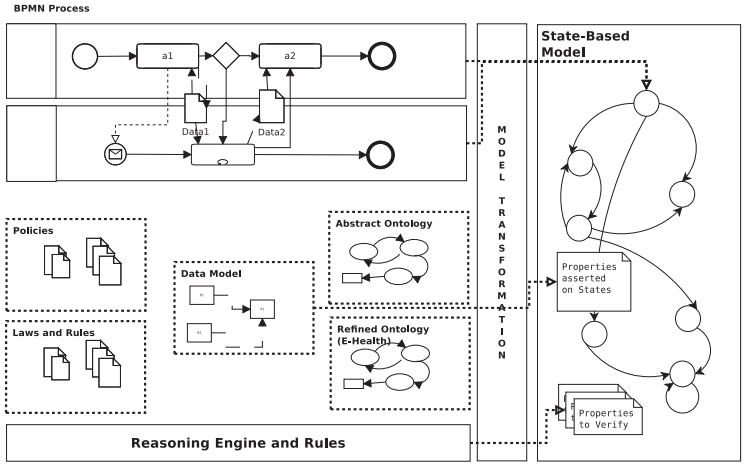

Fig. 3. Methodology overview.

schema used to store workflow data in a relational database management system, while the latter is extracted from the structure of available documents used in some steps of the process (e.g., medical programs, expense notes for drugs or medical provisions).

The relational model may not be enough to outline existing relationships among concepts. For this reason, our methodology supports an explicit semantic-based description of data by means of OWL ontologies. In particular, we consider ontologies at two levels of abstraction. An **Abstract Ontology** describes high-level concepts and properties related to the main elements of the relational model used by a workflow, including those related to the application of policies. A **Refined Ontology** contains the definition of additional concepts and properties related to the considered domain, plus a semantic-based description of the elements belonging to the relational schema of the proposed data model.

Security and privacy policies related to users, systems, and other elements in the process are also formally modeled by the **Policies** block, while the **Laws and Rules** block models applicable laws and regulations by means of ontologies.

The core idea behind our methodology is to leverage the knowledge base composed by the models discussed above to build a comprehensive model of the system that can be analyzed for validation purposes. In particular, we rely upon **Model Transformation** techniques to translate the description of a system in a source formal language (i.e., the *source model*), into a description of the same system in another formal language (i.e., the *target model*). Model transformation taxonomy defines two classes of translation algorithms: vertical and horizontal transformations [23]. The former class includes abstraction and refinement of models (e.g., automatic generation of code from UML diagrams), while the latter works on models having the same abstraction degree (they are also known as *model to model* transformations). The proposed methodology adopts a model-to-model transformation and, in particular, the target model of the system obtained upon transformation is represented by a **State-Based Model**. More specifically, we adopt ***Timed Automata*** (**TA**) [2], in order to exploit model checking procedures available for models defined in this language. A timed automaton is a finite state automaton that manages time and other variables, whose state transitions depend on the time variable evolution and on the occurrence of events (guards). TA are actually a composition of many automata that act concurrently in the analysis of the composite automaton (which is called product automaton).

As said, our methodology is aimed at exploiting model checking procedures available in the literature for TA. A model checking procedure enables one to assess whether a model satisfies a logical formula. Temporal logic is the *language* used to define these formulas. Since TA explicitly support time, model checking on TA usually leverages a logic called ***Timed Computation Tree Logic*** (**TCTL**) [1], or one of its variants. With this logic, it is possible to express conditions (formulas) like "property x holds in state y" or "property x holds from state y until the reaching of state z." Moreover, it is possible to express formulas related to paths, where a path is identified by a list of events (timed and guarded) and states that enable state transitions toward a target state, like "it exists at least one path connecting state y to state z."

Going back to our formalization of the workflow-based processing system, states in the target model obtained upon transformation identify data and information owned by actors. Interactions among actors and other events in the workflow processes (like choices, messages, timing events, etc.) define transition conditions among states, and help model the dataflow of information during the process. Since workflow processes can implement different control and dataflow patterns (e.g., parallels, synchronizations, or choices) [28], the target model has to be able to represent these different execution paths. For this reason, the target model is built upon synchronization (i.e., by building a product automaton) of different automata, as shown in detail in the next subsection. In this way, it is possible to correlate even activities from *different* workflow processes. This is important because one of the main problems that may arise is that the *same* actor can access data during the enactment of *different* workflows. Even if single workflows verify security and privacy policies, it may be possible to correlate information from different workflows and data sources and violate privacy and other rules.

In order to analyze if considered workflow processes satisfy existing policies and laws, the **Reasoning Engine and Rules** block identifies the properties to check on the target model depending not only on the available descriptions of policies, laws, and regulations, but also on the abstracted and refined concepts modeling system data.

In particular, *reasoning* on the target model follows three mains steps:

Step 1: We write a reasoning rule (**Semantic Web Rule—SWRL**). We need to verify the rule on each state of the target model. The rule cannot be verified directly on the Ontology because we do not have information about data owned by actors during the execution of the workflow process: this depends on the given workflow process and on the translated model.

Step 2: Since rules are usually a composition of some logical properties, we check these properties on the target model. If a property holds somewhere on the target model, we add a proper individual to the ontology, in order to enable the evaluation of the original reasoning rule.

Step 3: We reason again on the main rule, this time with proper individuals on the ontology, and we evaluate if any security problem exists.

Figure 4 resumes the main phases of the methodology we discussed before. In phases (a) and (b), we respectively translate BPMN processes and the relational schema into a TA, and we build a full ontology about extra information. Then, in phase (c) we exploit reasoning rules and abstract properties to check on the full ontology, in order to retrieve refined properties on instances of the ontology. In phase (d), we use these last elements as properties to check on the TA obtained in phase (a), in order to understand if a security property holds on the modeled system.

In the next subsection, we illustrate some details on how to build the target model and check properties on it.

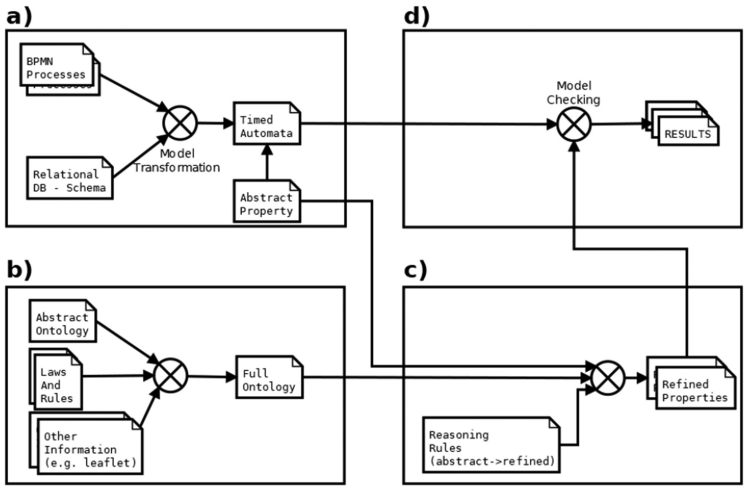

Fig. 4. Methodology phases.

4.1 Model Transformation and Reasoning

As introduced before, Timed Automata are the target formalism we use, in the variant proposed by the UPPAAL Model Checker [7]. In particular, *clocks* variables address time in states evolution. Some properties can be asserted on states, like state *invariant*, or on transitions, like conditions, guards, and variables assignments. Construction of product automata is enabled by a synchronization mechanism that allows for events assertions or waiting in different sub-automata. As described before, we face complexity of business processes by using a bottom-up approach: we first translate subprocesses in TA, and then we compose them into product automata.

Here we make the assumption of having already prepared the full ontology, so we can focus on steps (a), (c), and (d) of Figure 4. In phase (a), we first translate BPMN and relational schema into a TA model by creating, as anticipated, several sub-automata (one for each actor in BPMN processes) that are later composed in a product automaton. Since BPMN processes describe activities as well as data use and management, we divide the transformation algorithm into two parts. **BPMN control flow translation**: this part creates different automata analyzing pools and lanes in BPMN processes. Then we use interactions (from sequences and messages flows) in workflow processes to define guards and events on automata (i.e., synchronization elements among sub-automata), that allow for the correct creation of the product automaton. **BPMN data flow translation**: here we analyze access to data and documents in workflow processes, in order to define state information and transition actions on the automata created in the previous part. In addition, accesses to same or related data (this information is available from the data model) create further interactions (i.e., transitions, events, variables, constraints, and actions) among sub-automata.

Algorithms 1, 2, and 3 describe in detail the first part of model transformation.

Algorithm 1 shows how we analyze BPMN processes in order to instantiate target automata. Every *lane* in the BPMN workflow (i.e., every actor) corresponds to a component sub-automata. Usually, we translate BPMN activities into states of timed automata. In addition, we identify if parallel or choice patterns are in workflow definitions. In this case, we need to replicate the same behavior in the target model. This is done by creating a proper sub-automaton that manages synchronization. This part of the transformation is described in Algorithm 2. We use the mechanism of *urgent* semantics of locations (i.e., states). This allows all its outgoing transitions to be managed *before* any other evolution of automata states, implementing a synchronization mechanism

ALGORITHM 1: Control Flow Transformation

```
for each lane l in all pools do
    create a new automaton a_l
    Set the start state of a_l with all data available for actor l
    Identify Sequences, Parallel and Choice Patterns in l
    for each Parallel/choice pattern do
        Create a Sub-automaton for each parallel/choice path
        Create a sub-automaton for parallel/choice synchronization
    end for
    Create a Map M [Lane_Activity, TA_State]
    for all Activities act_i in l do
        create a new State s_i in a_l;
        add (act_i, s_i) to M
    end for
    ConnectState(M);
end for
```

ALGORITHM 2: Create Sub-Automaton for Parallel/Choice Synchronization

```
create a new sub-automaton for synchronization STA_a.
Create in STA_a a Start State ss and an End State es with urgent semantic.
if parallel synchronization then
    Create a new state s_i for each parallel branch with urgent semantic.
    for all s_i do
        connect with a new transition t_i, s_i with s_{i+1}, ss with s_1, and last s_n with es
        Create a new asserted event e_i for each parallel branch b_i ∈ BPMN pattern
        associate e_i to the transition incoming into t_{i-1}
    end for
else
    choice synchronization:
    for all choices c_j do
        create a new transition t_j connecting ss and se
        Create a new asserted event e_j labeled with the choice in BPMN pattern
    end for
end if
```

for component sub-automata. For parallel execution, we create as many events as is necessary to activate related sub-automata: events are asserted in sequence, but since we are using urgent states, the result is a parallel activation of automata from a simulation perspective. For choices, we only need to assert possibility on many transitions connecting start and end states of synchronization automata. Since model checking considers any possible evolution, we will have as many possible paths as the number of choices. Algorithm 3 describes the last step. It aims at connecting states created in previous steps. Since we connected created states from activities in BPMN lanes, basically we create a proper transition among states if a connection among activities in the lane exists. Notice that here we have already considered parallels and choices in the lanes. In addition, if connections among different lanes exist, they are translated by proper couples of waited and asserted events, depending on the lanes asserting and waiting the event.

ALGORITHM 3: ConnectState Procedure

Let $M.first$ be the set of Lane activities in M
$LetgetState\,(M, act_k)$ be the procedure that returns the state related to the key act_k in M
for all incoming transition t_k in $act_i \in M.first$ in the BPMN process **do**
  Create new transitions tr_m, in the automaton with:
  let s_o be the state in a_l generated from $act_k = from\,(t_k)$
  if act_k is in l **then**
    $from\,(tr_m) = s_o$
  else
    create a synchronization (waiting) event in a_l on tr_m from automata of the lane of act_k
  end if
  $to\,(tr_m) = s_i$
end for
for all outgoing transition t_s in act_i in the BPMN process **do**
  Create new transitions tr_n, in the automaton with:
  let s_p be the state in a_l generated from $act_j = to\,(t_s)$
  if act_j is in l **then**
    $to\,(tr_n) = s_p$
  else
    create an asserted event in a_l on tr_n
  end if
  $to\,(tr_n) = s_p$
end for

ALGORITHM 4: Data Flow Translation

Let L be the set of all states in target TA.
for all states $s_i \in L$ **do**
  add to s_i an empty List $data_i$
  get from $M\,[Lane\_Activity, TA\_State]$
  $la_i \in Lane\_Activity : s_i = M\,[la_i]$
  add to $data_i$ all data accessed in la_i
end for
for all data in all sets $\forall Lane\_Activity$ in all lanes **do**
  Let DP_{data} be the set of activities in the BPMN process that
  are in the data path for d.
  for all activities $act_k \in DP_d$ **do**
    get from all maps for all lanes $M\,[Lane\_Activity, TA\_State]$
    the all states s_w s.t. $s_w = M\,[act_k]$
    add to d to the list $data_w$ of s_w;
  end for
end for

Once we have the backbone of the target model, we must insert information about dataflow. This is the goal of the BPMN **dataflow translation** described in Algorithm 4.

In few words, we follow the data path for all data in the BPMN processes, and we add to each state in the target model the information that a state (i.e., an activity in BPMN workflows) accessed that data. For simplicity's sake, we addressed data access in general, without distinguishing

between read and write operations. Anyway, this can be generalized by a proper management of dataflow.

Once the TA model has been prepared, we can proceed to the phases (c) and (d) of the methodology depicted in Figure 4. Here, we refine abstract properties to check on the target model, by applying reasoning on the available ontology. We enforce reasoning by writing SWRL,[1] where a rule has the form

$$antecedent \Rightarrow consequent$$

where *consequent* is usually a simple predicate, while *antecedent* is a logical composition of simple predicates. The idea is the following: in phase (c), we write an abstract predicate to reason on the target model; then, we write some SWRL rules having the abstract predicate as consequent in the rules. If the rule is correct, we can isolate simple predicates of *antecedent*. Checking the abstract properties becomes the problem of checking all simple predicates, as well as the composite predicate in the antecedent, quantified in the same way of the consequent predicate.

Hence, if we have the SWRL rule in this form:

$$\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n \Rightarrow \phi.$$

In phase (d), we can check ϕ by checking all ψ_i, and by replying the same quantifiers of checked formula in the consequent part while checking the antecedent part. This means that if the check of ϕ requires a universal quantifier (i.e., we check that ϕ holds in all possible states in the evolution of target TA), we have to check ψ_i with the same quantifier, and then we can compose the antecedent, and check it again with proper quantifier depending on the composition of all ψ_i. In the example proposed above, since the antecedent is a conjunction of other formulas, we will use the same universal quantifier of the whole antecedent formula.

The next section illustrates an example for the application of the methodology and the execution of these steps.

5 CASE STUDIES

In this section, we propose two case studies in order to clarify and validate the application of our methodology. The first case is related to the scenario presented in Section 3. The aim of this first example is to show that, even if we have a system with *sound* policies, some violations of privacy may occur if we take into account public information and correlation of data. In the second case study, we have again a system with all policies to model rules according to current laws; in this scenario, we force the access to some information, and we ask our framework if there is a (legal) way to bypass policies enforcement and rules.

5.1 Example 1

Regarding the first case study, the workflow and the data model diagrams are reported in Figure 1 and Figure 2. In addition, we need to specify the abstract and the refined ontologies.

In the following figure, abstract and refined ontologies are represented in OWL. Figure 5 shows part of the abstract Ontology. OWL Classes are in named boxes; continue lines with arrows report *is-a* relationships; dotted lines with arrows report object properties (i.e., relationships among classes) with the name of the element reported in bold characters. In particular, let us notice the presence of concepts related to *Tables* and *References* for the data model, that are implemented by using foreign keys. A Table *contains* references to other information in other tables, and also References *connected to* other tables. This part of the ontology is used to infer connections among

[1] https://www.w3.org/Submission/SWRL/, retrieved May 30th 2020.

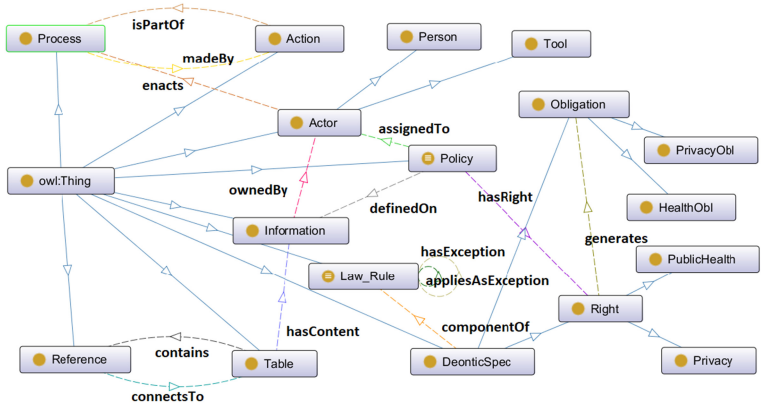

Fig. 5. Abstract ontology.

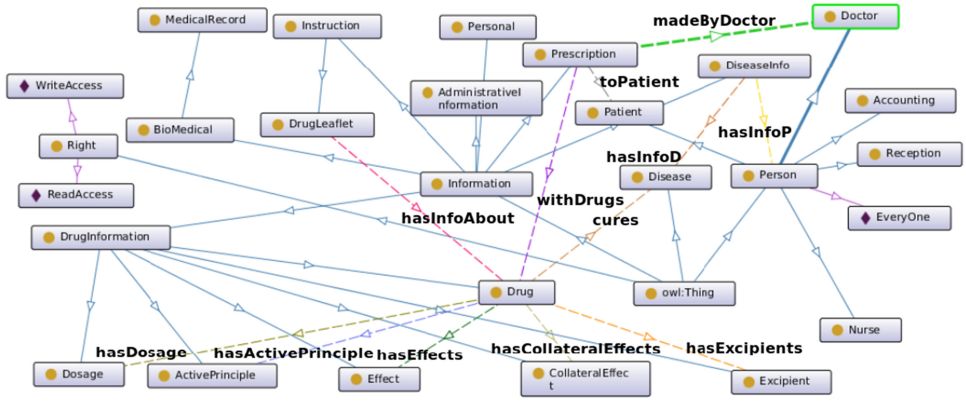

Fig. 6. Refined ontology.

information contained in the databases. *Information* is *owned by Actors*, which in turn can be classified as humans (*Person*) or computer applications (*Tools*).

The abstract ontology also models the case when a *Policy* defines access *Rights* for an *Actor* to information. Actors, in turn, are responsible for enactment of workflow *Processes*, which are composed of (*madeBy*) *Actions*. When defining a Law or a rule, we usually deal with *rights* and *obligations* expressed by deontic logic, so that these two elements can be managed as specification in a proper logic (*DeonticSpec*). In addition, laws and rules can have other laws and rules that define exceptions to the first ones (*hasException* and *appliesAsException* object properties). Notice that these last concepts will be used in the second example we propose here.

Figure 6 shows part of the refined ontology. In particular, it addresses the part related to Actors in the E-health example (*Doctor, Nurse, Accounting, Reception*, and *Patient*); the type of *Information* we deal with: *Medical records, Bio Medical* information, *Drug*-related-*Information*, Doctors *prescriptions*, drugs *Instructions* and related *Leaflets*. Moreover, the ontology contains classes related to *Drugs* with their *ActivePrinciples*, *Excipients*, *Effects*, and *Collateral Effects*. The class *DiseaseInfo* describes the information that a patient has a given disease. This is sensitive information that needs proper management according to GDPR. This last class has two connections, by proper object properties, to the patient (*hasInfoP*) and his disease (*hasInfoD*).

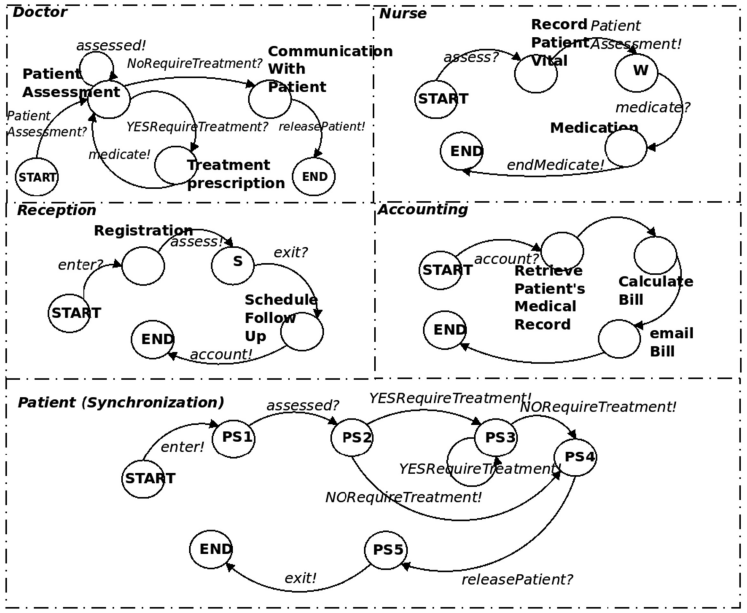

Fig. 7. Target model.

Notice how the object property *cures* describes the relationship between a drug and the disease it cures. As defined in the ontology, the same information is present in medical prescription (*withDrugs* object property), and in leaflets for the same drug. This relationship is not modeled in our Data Model, since usually leaflets are not considered as records to save. On the other hand, leaflets can be considered as public information, accessible to everyone.

According to the algorithm of model transformation, we applied transformation rules to the process described in Figure 1 and the result is a set of target automata. For simplicity's sake, Figure 7 only reports sub-automata of Doctors, Nurse, Reception, and Accounting actors, without showing variables hold in states or assignments on transitions for readability's sake. The figure reports states with their name (in bold characters). If they are generated by workflow activities, we defined in the states the same name of activities. On Transitions in target automata, names of synchronization (channels) events are in italic characters. According to TA notation, events awaited in a state end with a question mark; events generated by transitions end with an exclamation point.

Since the source workflow has some choice patterns (e.g., when a doctor chooses if a patient requires a treatment or not), we need a synchronizing automaton, which is the one called *Patient*, that we use as the starting point of the product automaton. In particular, a patient generates the event *enter!* that enables the transition of the Reception automaton from the START state to Registration. Then, an *Assess*sment is enacted by a Nurse that possibly requests a *PatientAssessment* to a Doctor, whose automaton enters in the state with the same name. There, depending on Patient events, the Doctor chooses if a patient requires a treatment with a medication (*YESRequireTreatment* event) or not (*NORequireTreatment*). Obviously, the Patient automaton generates both events. If the doctor decides for a medication, it makes a *Treatment Prescription* in the homonym state and then generates a request for medication (*medicate*) to a Nurse. The procedure iterates until the patient needs some treatments, prescriptions, and medications. When the patient needs no more treatments, the doctor can release him/her (*ReleasePatient*). The patient *exit*s from the hospital, and the accounting process can start (*Account*). The Accounting personnel then enacts all the activities to produce a

Table 1. Example of Data Held in Automata States

Automaton	State	Data	Access
Doctor	*	Patient.*	R
		Drug.*	R
		DrugComposition.*	R
		ActivePrinciple.*	R
		Prescription.*	RW
Reception	Registration	Patient.*	RW
Accounting	*	Patient.*	R
		Drug.Price	R
		Drug.ID	R
		Prescription.Patient	R
		Prescription.Doctor	R
		Prescription.Drug	R

Table 2. Example of Data Written During Automata Transitions

Automaton	Transition	Data	Operation
Doctor	medicate!	Prescription	Add record
Reception	assess!	Patient	Add record

bill for prescriptions. This requires the access to anonymized medical records to produce the bill and to send it by email. In this last activity, the Accounting personnel needs personal data of the patients (e-mail, name, etc.), but is not allowed to access patient' disease information, since this data requires proper authorization based on GDPR, i.e., the Accounting personnel cannot know patients' diseases, but they have to elaborate a bill for doctor prescriptions.

Table 1 contains an example of the variables held in some states and transitions of discussed automata with reference to the data model presented in Figure 2. As usual, the $*$ character is a wildcard and stays for *all elements*. We use dotted notation to indicate attributes of a database table. It is clear from Table 1 that a doctor is allowed to access all information of the patient, while the accounting personnel is able to access only to anonymized data: in fact, the attributes Patient, Doctor, and Drug of the Prescription table contain only IDs for external tables, to use as foreign keys.

Table 2 reports an example of actions on data enacted during transitions execution. By abusing notation, for simplicity's sake, we use the name of the event to address the related transition.

At this point, we have developed a system that seems to protect sensible data of patients. We can now reason over policies, workflow, and information from the ontology in order to detect unexpected privacy flaws. Hence, after the model translation phase, we need to follow the steps outlined in the last part of Section 4 to understand whether someone in the workflow can access to reserved information. To this aim, let us consider the example related to the actors in the *Accounting* role, who cannot access data that associate personal information of patients with their diseases, but have to prepare bills and need at least patients' names and address to complete this task. To enable the execution of the billing task while preserving patients' privacy, the database that stores prescriptions and other data anonymizes information by storing, in the Prescription table, only external references (i.e., foreign keys) to other tables. It is worth noting, moreover, that the considered database does not even store any data about disease associated to drugs. Anyway,

as anticipated, we assume the existence of other sources of information (see Figure 6), like leaflets, that actually contain a description of the diseases that a drug is able to treat. If we use a standard common identifier for drugs (such as EMEA European universal identifier, or the AIC code in Italy) to identify a drug in the database, then it is possible to relate the content of table *Prescription* to the right drug leaflet.

In order to illustrate how our methodology is able to cope with this condition, let us discuss all the steps introduced in Section 4. In the first step, a SWRL rule to be evaluated is defined. The following equation describes an example:

$$\begin{aligned} & Drug\,(?dr) \wedge cures\,(?dr, ?dis) \wedge Disease\,(?dis) \\ \wedge & Prescription\,(?pr) \wedge toPatient\,(?pr, ?pat) \wedge withDrugs\,(?pr, ?dr) \\ \wedge & hasInfoAbout\,(?info, ?dr) \wedge ownedBy\,(?info, : Everyone) \end{aligned} \tag{1}$$

$$\begin{aligned} \wedge & Actor\,(?act) \wedge Information\,(?dr) \wedge Table\,(?t) \\ \wedge & hasContent\,(?T, ?dr) \wedge ownedBy\,(?dr, ?act) \\ \wedge & diseaseInfo\,(?dinfo) \wedge hasDiseaseP\,(?dinfo, ?pr) \\ \wedge & hasDiseaseD\,(?dinfo, ?dis) \end{aligned} \tag{2}$$

$$\longrightarrow ownedBy\,(?dinfo, ?act) \tag{3}$$

The first part of rule (1) describes the case when a Prescription (*pr*) exists for a Patient (*pat*) related to a Drug (*dr*). In addition, the rule states that the Drug *dr* cures the Disease *dis*, and the information that *dr* cures *dis* is owned by Everyone. In this case, in order to associate the information of a disease with a patient, the only thing to know is the description of a well-known information about *dr* (in order to retrieve needed information from a leaflet).

This condition is reported in the second part of rule (2), where we model the condition that an Actor (*act*) owns an Information $dinfo$ from a Table d able to correlate a person ($hasDiseaseP\,(?dinfo, ?pr)$) to a disease ($hasDiseaseD\,(?dinfo, ?dis)$).

The last part of Equation (3) is the knowledge the rule infers: if preconditions in part 1 and part 2 hold in a state, we can infer that the actor to which the state belongs, owns a confidential information. At this point, we can match this with available policy and rules defined for the actor, in order to state if some of them are violated.

The second step of the methodology entails the evaluation of all predicates in the SWRL rule in each state of the timed automata modeling the system. The elements in part 1 of the rule are always true in the automata, since we are reasoning in the case of a drug with a leaflet, that has been prescribed to a patient by a doctor. Part 2 of the rule, instead, holds in the state *Accounting. "Retrieve patient's medical record,"* and in all subsequent states of the same actor. Hence, unifying the formula with the given generic Drug *D*, the table *Prescription*, the actor *Accounting*, and the generic Patient *Pr*, we can create proper individuals in the ontology.

We apply Model Checking on the target model for the verification of the property in the aforementioned state. The formula we use to check the state is not in the form exposed by part 2. In fact, let ϕ be the formula related to part 2. If we want to study if the formula holds in one of the possible states generated during the evolution of the target model, we have to "ask" the model checker if, starting from a given state, we *eventually* have one evolution of the target model that leads (in the *future*) to a state where ϕ holds. This corresponds to the evaluation of the formula in TCTL, quantified by temporal operators *Exists* (E) and *Finally* (F). If we try to check the CTL formula,

$$EF\,(\phi)\,,$$

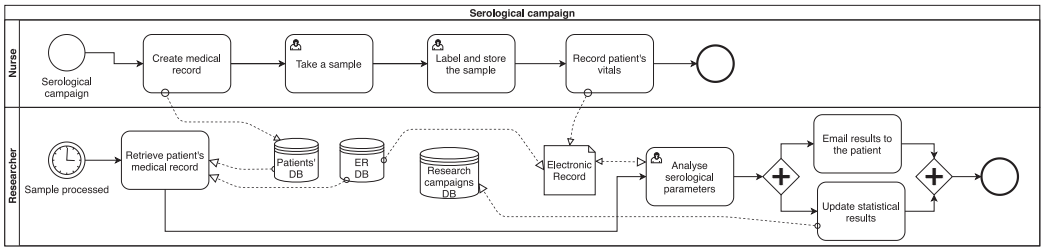

Fig. 8. Workflow process of the second example.

we have a positive check, but we are not able (except by using interactive simulation) to understand the state where the formula holds. Hence, we try to negate the formula, checking this second one:

$$AG\,(\neg\phi)\,.$$

Here we use universal quantifiers *All* (A) and *Globally* (G): we are checking if the formula ($\neg\phi$) holds in any possible evolution of the target model, and for each evolution, the formula ever holds. If we have a negative check, a counterexample is given by the model checker, that identifies one of the states where the formula holds.

Once we have populated the ontology with all individuals above identified, we are able to reason on the full SWRL rule (Step 3 of the methodology). The reasoning process obviously returns that the actor Accounting is able to create a connection (a *DiseaseInfo* individual) between a patient and his/her disease. This is done (as identified in the second step of the methodology) in the state *Accounting. "Retrieve patient's medical record,"* thus violating a GDPR rule.

Notice that the execution of the second step alone (i.e., model checking the property) does not identify any specific privacy problem, because the workflow process and the timed automata, as well as the data model, do not include any evidence of the existence of a leaflet, that has been codified in the ontology. On the other hand, reasoning directly on the ontology is not useful, as we do not have any information on the workflow we are analyzing, nor on the states that can lead to potential privacy problems.

5.2 Example 2

Let us consider the second example, where we consider a simpler workflow process shown in Figure 8. Here we have two actors in two lanes: the **Nurse** is responsible for taking blood samples from patients and storing results from the analyses in the medical records. A nurse is enabled to read the Patients' DB and the documents in the ER DB. In the other lane, there is a new actor: the **Researcher**. He needs to collect data from patients' medical records in order to update statistics on retrieved data for research purposes. In particular, in the example, the researcher uses medical records to analyze serological parameters in an epidemic scenario, updating statistical results and notifying patients of gathered information.

It is worth outlining that, in normal conditions, if the owner of a medical record does not provide any explicit informed consent, no one (including researchers) is enabled to access that data (unless information is anonymized). Thus, common access policies would forbid this kind of analysis. However, according to the GDPR regulation, the general prescriptions related to prohibiting personal data processing can be waived if at least one of the conditions specified in Article 9 (paragraph 2) is met. In particular, for the health sector, the conditions set out in points (h) and (i) are relevant. Point (h) refers to the case in which "processing is necessary for the purposes of preventive or occupational medicine, for [...] the provision of health or social care or treatment or the management of health or social care systems and services [...] and subject to the conditions and

Table 3. Researcher Accesses to Data

Automaton	State	Data	Access
Researcher	Analyze	Patient.BirthDate	R
	Serological	Sample.*	R
	parameters	Parameter.*	R

Table 4. Instances in Refined Ontology

Instance	Class	Domain in property
Art9GDPR	Law_Rule	hasException(Par1Art9GDPR)
PatientSample	Information	
AuthorizedAccess	Policy	assignedTo(Patient)
	Policy	definedOn(PatientSample)
	Policy	hasRight(BioDataPrivacy)
BioDataPrivacy	Right	
Art53GDPR	Law_Rule	appliesAsException(Art9GDPR)
Researcher	Actor	

safeguards referred to in paragraph 3." Paragraph 3, in particular, specifies that personal data must be processed by or under the responsibility of a professional (or other person) who is subject to professional secrecy. Point (i) refers to the case in which "processing is necessary for reasons of public interest in the area of public health, such as protecting against serious cross-border threats to health." Once the existence of one (or more) of the above-mentioned conditions has been ascertained, the general prohibition applied to information processing is no longer applicable, with the consequence that health-related data may be processed, similarly to the "ordinary" categories of personal data.

Our methodology devises the modeling of these conditions as well, which are represented in our ontology as legal rule exceptions, and used to establish if a data processing action can be considered legal or not, depending on the processing purposes.

We omit for this example the TA generated by model transformation due to the lack of space. Anyway, the workflow process of Figure 8 is simple enough to imagine the target model. In addition, here we want to point out the focus on the reasoning step. With reference to Figure 2, we simply assume that a *Nurse* has read rights on the *Patient* table, and write rights to *MedicalRecordDoc*, *Sample*, and *Parameter*. In addition, we assume, in this example, that a Researcher actor is normally not allowed access to any data because of GDPR rules defining privacy restrictions to bio-medical data. Finally, let us assume that action *Analyze Serological parameters* requires access to data reported in Table 3.

Even if some access policies exist that negate the ability of a Researcher to read medical data, we can use our methodology in order to verify whether, depending on specific context information (i.e., the processing purpose), the access can be actually granted. As described before, we need here a refined ontology containing domain-related laws and policies. In particular, we consider both privacy rules of GDPR (that we addressed in the first example), and the rule exception we describe in Table 4 as ontology instances.

The table contains only elements useful to implementing reasoning rules. In particular, we have the rule (Article 9 of GDPR) stating that bio-medical information can be accessed only upon explicit authorization of patient. Hence, data concerning Patient samples (like the table Sample in Table 3)

cannot be usually read and analyzed without consensus because this violates the Right *BioDataPrivacy*. Anyway, a rule exists that defines an exception to Article 9 of GDPR. This is Article 53 of GDPR that states that in case of global conditions impacting public health (like epidemic), data can be accessed for the purpose of providing effective solutions at the National and International levels.

We can model preconditions of laws, rules, and exceptions in the refined ontology, in order to understand if a law or an exception is applicable. Anyway, for simplicity's sake, we do not illustrate this part of the ontology and we limit this example to the identification of an *opportunity* to violate a rule. All we need here is to formalize a reasoning rule that is able to understand if we can *legally violate* an existing privacy policy.

The SWRL rule we use at step (3) of the methodology is the following one:

$$\begin{aligned} &Actor\,(?pat) \wedge Information\,(?info) \wedge Table\,(?t) \wedge hasContent\,(?t, ?info) \wedge ownedBy\,(?dr, ?pat) \\ &\wedge Actor(?researcher) \wedge Policy\,(:AuthorizedAccess) \wedge Right\,(?r) \wedge hasRight(:AuthorizedAccess, ?r) \wedge \\ &\neg assignedTo\,(:AuthorizedAccess, ?researcher) \wedge Law\_Rule\,(?lr) \wedge componentOf\,(?r, ?lr) \wedge \\ &Law\_Rule\,(?excp) \wedge hasException\,(?lr, ?excp) \\ &\longrightarrow assignedTo\,(:AuthorizedAccess, ?researcher) \end{aligned} \tag{4}$$

Here, we declare that, if we have a Patient (the actor *?pat* in the rule) and a Researcher (the actor *?researcher*), when the Patient can access to an information (with the right *:AuthorizedAccess*, which is *assignedTo* him/her), and if the same right is not assigned to the Researcher, despite there exists a law (*?lr*) that negates the access to Researcher, if there exists a valid law that acts as an exception of the former one (*?excp*), we hypothesize that the researcher can still access to information *?info*.

It is easy to prove that the aforementioned instances (Table 4) and ontology structure makes true the *consequent* part. In this way, once we have prepared rule (4), we can use it in different scenarios where we need to understand if we can violate a privacy rule. Unification of *variables* in the *antecedent* part helps identify laws and conditions (i.e., ontology instances) that enable violations.

6 CONCLUSIONS AND FUTURE WORKS

Modern e-Health systems are very complex, due to the high degree of decentralization and distribution of data and control, and to the typical integration of several different subsystems and services, often legacy, which implies a security-hardening of the system that is usually done a posteriori (often with a reduced efficacy). Another issue is represented by the presence of complex workflows to perform the needed clinical processes, which entail complex flows of information, including personal sensitive data subject to privacy regulations. While specific policies can be set up and enforced within a system to regulate the access to specific data by a subset of authorized subjects, a privacy violation may still happen due to the inherent correlations existing among data, which may be exploited in some steps of the workflows to infer protected information.

In this work, we have shown how it is possible to exploit workflow-based definitions of processes involving information in e-Health systems, in combination with a semantic-based definition of domain entities, rules, and data models, in order to perform automated security and privacy policy validation in complex systems, by leveraging reasoning and model checking techniques. We have illustrated, with reference to a case study, all the steps of the proposed validation methodology, demonstrating that it is able to verify the fulfillment of a policy by taking into account all the relevant aspects (workflow, dataflow, semantics of data, global conditions, laws and regulations, end-user defined policies).

As a final remark, it is worth mentioning that, although in this article we referred to an e-Health domain, the proposed methodology can be applied to any system characterized by complex

workflow processes involving the access to and the transmission of sensitive data, in the presence of heterogeneous policies and of complex inter-correlation of data. Therefore, we plan to improve it to include the evaluation of quantitative metrics and to apply it to other domains.

REFERENCES

[1] Rajeev Alur, Costas Courcoubetis, and David Dill. 1993. Model-checking in dense real-time. *Information and Computation* 104, 1 (1993), 2–34.

[2] Rajeev Alur and David L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2 (1994), 183–235.

[3] F. Amato, V. Casola, G. Cozzolino, A. De Benedictis, and F. Moscato. 2019. Exploiting workflow languages and semantics for validation of security policies in IoT composite services. *IEEE Internet of Things Journal* (2019), 1–1. DOI: https://doi.org/10.1109/JIOT.2019.2960316

[4] Muhammad Asim, Artsiom Yautsiukhin, Achim D. Brucker, Thar Baker, Qi Shi, and Brett Lempereur. 2018. Security policy monitoring of BPMN-based service compositions. *Journal of Software: Evolution and Process* 30, 9 (2018), e1944.

[5] Hasiba Attia, Laid Kahloul, Saber Benharzallah, and Samir Bourekkache. 2019. Using hierarchical timed coloured Petri nets in the formal study of TRBAC security policies. *International Journal of Information Security* 19 (2020), 163–187. DOI: https://doi.org/10.1007/s10207-019-00448-9

[6] David Basin, Felix Klaedtke, Samuel Müller, and Eugen Zălinescu. 2015. Monitoring metric first-order temporal properties. *Journal of the ACM* 62, 2 (May 2015), Article 15, 45 pages. DOI: https://doi.org/10.1145/2699444

[7] Gerd Behrmann, Alexandre David, and Kim G. Larsen. 2004. A tutorial on UPPAAL. *Formal Methods for the Design of Real-time Systems*. Springer, 200–236.

[8] S. Chenthara, K. Ahmed, H. Wang, and F. Whittaker. 2019. Security and privacy-preserving challenges of e-health solutions in cloud computing. *IEEE Access* 7 (2019), 74361–74382. DOI: https://doi.org/10.1109/ACCESS.2019.2919982

[9] Junho Choi, Chang Choi, SungHwan Kim, and Hoon Ko. 2019. Medical information protection frameworks for smart healthcare based on IoT. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics (WIMS'19)*. Association for Computing Machinery, New York, NY, Article 29, 5 pages. DOI: https://doi.org/10.1145/3326467.3326496

[10] Peter R. Croll. 2011. Determining the privacy policy deficiencies of health ICT applications through semi-formal modelling. *International Journal of Medical Informatics* 80, 2 (2011), e32–e38. DOI: https://doi.org/10.1016/j.ijmedinf.2010.10.006. Special Issue: Security in Health Information Systems.

[11] Salvatore Cuomo, Francesco Maiorano, and Francesco Piccialli. 2018. Remarks of social data mining applications in the Internet of data. In *International Conference on Network-Based Information Systems*. Springer, 944–951.

[12] European Commission. [n.d.]. General Data Protection Regulation. Retrieved January 23, 2020 from https://gdpr-info.eu/.

[13] Bahar Farahani, Mojtaba Barzegari, Fereidoon Shams Aliee, and Khaja Ahmad Shaik. 2020. Towards collaborative intelligent IoT eHealth: From device to fog, and cloud. *Microprocessors and Microsystems* 72 (2020), 102938. DOI: https://doi.org/10.1016/j.micpro.2019.102938

[14] Antonios Gouglidis, Ioannis Mavridis, and Vincent C. Hu. 2014. Security policy verification for multi-domains in cloud systems. *International Journal of Information Security* 13, 2 (April 2014), 97–111. DOI: https://doi.org/10.1007/s10207-013-0205-x

[15] Michele Guerriero, Damian Andrew Tamburri, and Elisabetta Di Nitto. 2018. Defining, enforcing and checking privacy policies in data-intensive applications. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'18)*. Association for Computing Machinery, New York, NY, 172–182. DOI: https://doi.org/10.1145/3194133.3194140

[16] Jigna J. Hathaliya and Sudeep Tanwar. 2020. An exhaustive survey on security and privacy issues in Healthcare 4.0. *Computer Communications* 153 (2020), 311–335. DOI: https://doi.org/10.1016/j.comcom.2020.02.018

[17] Vincent Hu, D. Kuhn, Tao Xie, and Jeehyun Hwang. 2011. Model checking for verification of mandatory access control models and properties. *International Journal of Software Engineering and Knowledge Engineering* 21 (Feb. 2011), 103–127. DOI: https://doi.org/10.1142/S021819401100513X

[18] V. C. Hu and D. R. Kuhn. 2016. General methods for access control policy verification (application paper). In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI'16)*. 315–323. DOI: https://doi.org/10.1109/IRI.2016.49

[19] Amani Abu Jabal, Maryam Davari, Elisa Bertino, Christian Makaya, Seraphin Calo, Dinesh Verma, Alessandra Russo, and Christopher Williams. 2019. Methods and tools for policy analysis. *ACM Computing Surveys* 51, 6 (Feb. 2019), Article 121, 35 pages. DOI: https://doi.org/10.1145/3295749

[20] Fakhri Alam Khan, Sadaf Shaheen, Muhammad Asif, Atta Ur Rahman, Muhammad Imran, and Saeed Ur Rehman. 2019. Towards reliable and trustful personal health record systems: A case of cloud-dew architecture based provenance

framework. *Journal of Ambient Intelligence and Humanized Computing* 10, 10 (2019), 3795–3808. DOI: https://doi.org/10.1007/s12652-019-01292-4

[21] J. Ma, D. Zhang, G. Xu, and Y. Yang. 2010. Model checking based security policy verification and validation. In *Proceedings of the 2010 2nd International Workshop on Intelligent Systems and Applications*. 1–4. DOI: https://doi.org/10.1109/IWISA.2010.5473291

[22] Irfan Mehmood, Zhihan Lv, Yudong Zhang, Kaoru Ota, Muhammad Sajjad, and Amit Kumar Singh. 2019. Mobile cloud-assisted paradigms for management of multimedia big data in healthcare systems: Research challenges and opportunities. *International Jouornal of Information Management* 45 (2019), 246–249. DOI: https://doi.org/10.1016/j.ijinfomgt.2018.10.020

[23] Tom Mens and Pieter Van Gorp. 2006. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science* 152 (2006), 125–142.

[24] Samrat Mondal, Shamik Sural, and Vijayalakshmi Atluri. 2011. Security analysis of GTRBAC and its variants using model checking. *Computer Security* 30, 2–3 (March 2011), 128–147. DOI: https://doi.org/10.1016/j.cose.2010.09.002

[25] Francesco Piccialli and Jason J. Jung. 2018. Data fusion in the internet of data. *Concurrency and Computation: Practice and Experience* 30, 15 (2018), e4700.

[26] Rohit Ranchal, Bharat Bhargava, Pelin Angin, and Lotfi Ben Othmane. 2018. Epics: A framework for enforcing security policies in composite web services. *IEEE Transactions on Services Computing* 12, 3 (2019), 415–428.

[27] Sriti Thakur, Amit Kumar Singh, Satya Prakash Ghrera, and Mohamed Elhoseny. 2019. Multi-layer security of medical data through watermarking and chaotic encryption for tele-health applications. *Multimedia Tools and Applications* 78, 3 (Feb. 2019), 3457–3470. DOI: https://doi.org/10.1007/s11042-018-6263-3

[28] Wil M. P. Van Der Aalst and Arthur H. M. ter Hofstede. 2012. Workflow patterns put into context. *Software & Systems Modeling* 11, 3 (2012), 319–323.

Received February 2020; revised June 2020; accepted July 2020