Extended Abstract: Towards Verified Privacy Policy Compliance of an Actor-based Electronic Medical Record Systems

Tom MacGahan Claiborne Johnson Armando L. Rodriguez Mark Appleby Jianwei Niu Jeffery von Ronne
Department of Computer Science
The University of Texas at San Antonio
vonronne@acm.org

1. Introduction

Many organizations use information systems to store, process, and communicate personal information about the individuals with which the organization interacts. These organizations needs to ensure that these information systems do not disclose information when not allowed by applicable privacy policies. One important instance of this are the electronic medical records systems utilized by health care providers. In the United States, these organizations are required to comply with the Privacy Rule of the Health Insurance Portability and Accountability Act and their own privacy policies. (In the rest of these papers, we will refer to these simply as "the privacy policy".) Verifying, whether a complex software system can cause an organization to violate the privacy policy, however, is not an easy task.

We have been developing a framework for building actor-based privacy policy compliant software systems. In this framework, the software system is decomposed into collections of actors instantiated from class and component definitions written in the domain-specific History Aware Programming Language (HAPL). Along the lines previously outlined in [2]. Each actor class definition and each component definition is also associated with formal specification written of relevant parts of its behavior. Actor code is verified against individual actor class specifications, and individual class specifications are shown to entail the specifications of larger component entities. The specification for the entire system is shown to entail the privacy policy. In this way, the system implementation is shown to implement the privacy policy.

We are currently in the process of creating the framework's build tools and runtime systems. We are also applying the framework to build a prototype electronic medical record system.

2. Framework Design and Implementation

Our framework can be viewed as facilitating two processes: creating a information system, and verifying its compliance with the privacy policy. The two are not unrelated, however, since the tools and techniques used to build the system are constrained so that the system can be more easily verified.

2.1 Language, Build Tools, and Runtime Support

In our framework, an information system is written in the History Aware Programming Language (HAPL), which provides facilities for defining *actor classes* and *component definitions*. In HAPL, an actor class is a static template from which actors can be instantiated at runtime. A component definition is a static description of a configuration of actors that can be instantiated at runtime. Component definitions can be defined in terms of actors instantiated from actor classes or subcomponents instantiated from other component definitions. A component definition is used to describe the system as whole.

A HAPL compiler translates actor classes and component definitions into Scala code using the Akka actor library. HAPL actor class code can make use a novel "dynamic history query" to make the execution of certain code dependent, at runtime, on the truth of first-order LTL formulas involving the messages in which the actor participates. This is supported through a runtime mechanism that dynamically tracks information necessary to answer potential queries. Special UI actor classes in the HAPL code generate Lift framework code to create actors that can communicate with a user through a web browser.

2.2 Verification Tools and Techniques

Our verification techniques aim to verify complete information systems against a formalization of the privacy policy in a first-order temporal logic.

Furthermore, our framework is built around a set of assume-guarantee specifications written in a first-order temporal logic. These specifications are associated with each actor class and each component definition. Specifications associated with component definitions describe not only the behavior of the actors in the initial configuration of components instantiated out of a definition but also describes the externally visible behavior of any new actors that are created as the component evolves from those in the initial configu-

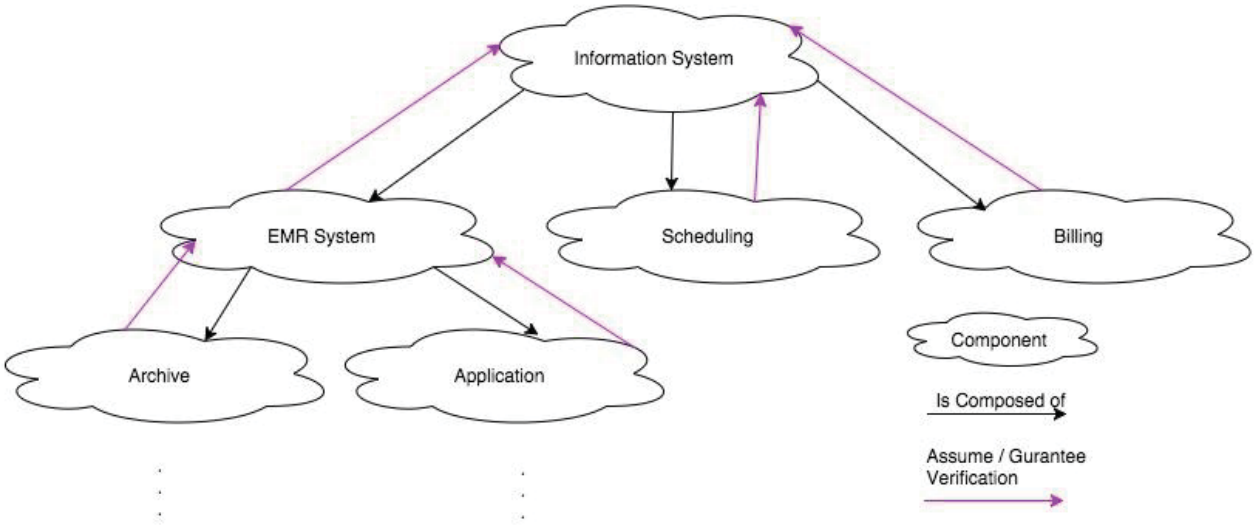


Figure 1. Component Structure of Prototype Medical Record System

ration. The specification associated with the "outer-most" whole-system component is taken as system specification.

Individual actor classes are verified against their specifications using an abstract interpretation that abstracts program traces according to the specification subformulas that are true when execution reaches a particular program point.

A variety of formal methods can be used to show that the refinement of a system specification into the specifications of the component definitions that realize it, as well as the specification of the component definitions into the subcomponent definition and actor class definitions. We are currently looking into utilizing model checking and Coq theorem proving for this purpose.

3. Prototype Medical System

In order to evaluate our framework, we have been using it to develop a prototype electronic medical record system. It consists of a patient and doctor user interface, an medical record archive, a scheduling subsystem and a billing subsystem. Patients are allowed to view their own record. Doctors can view and edit records of any patient for whom they claim to be giving care. The medical record archive is presumed to be persistent and is acts as a database.

The hierarchical component structure of the prototype medical system is shown if Figure 1. The "Information System" component is comprises of the "EMR System", "Scheduling", and "Billing" subcomponents. Our framework associates each of these components has an associated assume-guarantee specification. It is then necessary to show that the supercomponents specification is entailed by the specifications of the subcomponents and actors that comprise the supercomponent.

4. Future Work and Outlook

Although our policy language is well developed [1], many aspects of our framework are still incomplete. We are in the process of completing our runtime system implementation, our verification tools, and verifying our medical system prototype.

We are also still investigating the best method for verifying that the subcomponent assume/guarantee specifications compose to entail the supercomponent assume/guarantee specification; we are currently looking at using model checking tools or the Coq assisted theorem prover.

Our experience so far, however, leads us to believe in the appropriateness of our framework for building privacy-protecting actor-based information systems.

Acknowledgments

This work is dedicated to the memory of William Winsborough. The authors also thank Omar Chowdhuri, Shamim Ashik, and Sam Miller who have contributed to this project. This work was supported by the National Science Foundation under grant CNS-0964710.

References

[1] O. Chowdhury, A. Gampe, J. Niu, J. von Ronne, J. Bennatt, A. Datta, L. Jia, and W. H. Winsborough. Privacy promises that can be kept: A policy analysis method with application to the hipaa privacy rule. In *Proceedings of the 18th ACM Symposium on Access Control Models and Technologies*, SACMAT '13, pages 3–14, 2013.

[2] J. von Ronne. Leveraging actors for privacy compliance. In *Proceedings of the 2Nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions*, AGERE! 2012, pages 133–136, 2012.