UML for Inclusion of Privacy in Software Modeling

by

Sohail Ali

Submitted in partial fulfilment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
April 2013

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “UML for Inclusion of Privacy in Software Modeling” by Sohail Ali in partial fulfilment of the requirements for the degree of Master of Computer Science.

Dated: April 4, 2013

Co-Supervisors _____

_____

Readers _____

_____

DALHOUSIE UNIVERSITY

DATE: April 4, 2013

AUTHOR: Sohail Ali

TITLE: UML for Inclusion of Privacy in Software Modeling

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: MCSc CONVOCATION: October YEAR: 2013

_____

Signature of Author

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Online commerce and service obtain much private data from users. Collection, storage, management, and use of private data are subject to various privacy laws, regulations, and standards. To adhere to legal requirements, many privacy services, such as security, notice, and consent, are required. Inclusion of the required privacy services early in the life cycle of the software development is preferred and advocated. We extend UML use case diagrams with privacy components to represent example privacy services. These components are used to visually model privacy requirements in the analysis phase of the SDLC. We create a prototype by extending Microsoft Visio, a popular UML modeling tool, with our proposed privacy components. In summary, we show how privacy services may be specified in UML use case diagrams rather than adding privacy as an afterthought to software systems and services. The tool is demonstrated with real-world scenarios from the health sector.

LIST OF ABBREVIATIONS USED

ACL	Access Control List
CIM-SPL	CIM Simplified Policy Language
EPAL	Enterprise Privacy Authorization Language
ERM	Entity Relationship Modeling
FIP	Fair Information Practices
OWL	Web Ontology Language
P3P	Platform for Privacy Preference
PAS	Publish Privacy Act Statements
PI	Personal Information
PII	Personally Identifiable Information
PP	Privacy Policy
PPI	Privacy Program Information
PPT	Privacy Policy Template
PPVM	Privacy Policy Visualization Model
PPVM	Privacy Policy Visual Model
PPVM	Privacy Policy Visual Model
RBAC	Role Based Access Control
RDF	Resource Description Framework
SORNs	System of Records Notices
SPARCLE	Server Privacy Architecture and Capability Enablement
UML	Unified modeling Language
UR	Unit Receptionist
VSTO	Visual Studio Tools for Office
XACML	eXtensible Access Control Markup Language

ACKNOWLEDGEMENTS

With a deep sense of gratitude, I wish to express my sincere heartfelt thankful to my supervisor, Dr. Dawn Jutla, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the problem. Her perpetual energy and enthusiasm in research had motivated me a lot. In addition, she was always accessible and willing to help her students with their research. As a result, research life became smooth and rewarding for me.

I am also thankful to the entire faculty and staff members of Department of Computer Science of their direct and indirect unconditional help and cooperation that made my stay at Dalhousie University memorable.

Moreover, I would also like to thank my parents and my wife for their prayers and support and to all my friends for always encouraging me and believing in me.

CHAPTER 1 : INTRODUCTION

The internet has penetrated all aspects of life and has become a common model of interaction between businesses and customers through offering reliable and efficient services, whether in online shopping, making reservations, or visiting doctors (Md. Moniruzzaman, Ferdous, & Hossain, 2010), (Ghazinour, Majedi, & Barker, 2009). Buying and selling products or services through the web has been popular for the last two decades and has caused revenue for many companies to grow exponentially. It is thus very important that customers are provided with secure, quick and easy-to-use services at all times. Current trends are accelerating the demands of online or connected services in a secure and controllable way and such demands are not supported by existing hosting services or underlying platforms (Massacci & Naliuka, 2008). This has increased concerns of individuals and organizations regarding the protection of customers' personal information (PI) (Olivier & Oberholzer, 2005). Users unwittingly reveal a lot of their private and personal information when they are using online services. If such private and personal information is not protected through enforcement of privacy policies throughout an organization, it can be unintentionaly misused and/or leaked (Md. Moniruzzaman, Ferdous, & Hossain, 2010).

Maintaining and managing privacy of users' information is tedious, but since information is the corner stone of today's business, enforcement of privacy policies is essential to protect systems from unauthorized leaking of valuable customer and employee personal information. Privacy policies are drawing high attention from the business, information system, legal and other communities. Security and privacy policies are getting quite complex due to the integration

of large numbers of inter and cross-enterprise applications, e-commerce, social networking apps (Bodorik, Jutla, & Dhillon, 2009). Aditional privacy and security issues arise in applications that utilize cloud computing.

Identifying and making appropriate changes to rectify the errors in privacy models for software systems after the systems have been built can be expensive. Thus, it is crucial that a privacy model, used in building the system, be thoroughly examined at all the stages of software development to minimize the cost of mitigation of errors in later stages. Successful application and enforcement of privacy policies demands many years of experience, expertise, continuous improvement and adherence to laws (Goudalo & Seret, 2008). There are many tools available for writing and expressing security and privacy policies in business software such as XACML (Godik & Moses, 2005), CIM-SPL (Denker, Kagal, & Finin, 2005), and OWL/RDF (Lobo, 2009). These tools are very specific and do not provide support to software engineers and architects to visually model privacy controls when developing the system. There are no suitable tools available to incorporate a privacy control in use-case diagrams while modeling the overall system designs. Adhering to privacy laws, regulations of the organization, global and local policies and security laws, helps to increase the customer's trust and increase the likeliness of retaining the customer's business ((Goudalo & Seret, 2008), (Bodorik & Jutla, 2008)).

Currently available technologies, such as UML, Access Control List (ACL), Enterprise Privacy Authorization Language (EPAL), and Entity Relationship Modeling (ERM), lack in facilitating designers and engineers to visually integrate privacy controls in commonly used UML use-case or sequence diagrams during system modeling. In addition to the currently

existing technologies not being easy to master and demand experience, they do not have appropriate privacy controls to represent an organization's privacy policy during the software development phase.

In this thesis, we detail and implement parts of a proposed and promising approach (Jutla, 2012a, b, c, d) that will facilitate the software engineers, architects, project managers and policy auditors, to incorporate privacy controls, which may be represented by privacy services, during the software development phase. This work is important to all the key stake-holders in the early phase of software development. Our solution is based on the existing industry de-facto standard UML, a widely used tool by the industry to document the core components and overall behavior of the system. This work will substantially reduce the learning and application time for all those who will use it if they have a basic understanding of UML and use-cases. The proof-of concept for our work is implemented on Microsoft Visio 2010.

Furthermore, this work may also allow privacy policy officers and auditors to verify whether the system adheres to the defined privacy policy guidelines through easy-to-understand visual diagrams for privacy embedded in software documentation.

1.1 Research Problem

As the number of service providers is increasing, the collection of private and personal data from customers is also increasing – this results in increasing responsibility of organizations' handling of the collected data. The current way that the software industry is modeling privacy is not explicit and does not provide the insight to organization's compliance to its privacy policies.

Software developers and engineers should take into account the privacy and security of personal data from the beginning of the software analysis and design phase. Indeed, the way software has been written has not focused much on privacy policies or regulation compliance at initial stages of software development, but, rather, it adds privacy policies compliance on an ad hoc basis at later stages of the development. Therefore, the key problem with the state-of-the-art is that no appropriate way to integrate visual privacy requirements modeling at the beginning of the software development phase is available. This lack may result in the non-compliance with privacy policy and laws.

1.2 Research Objectives

The objectives of this thesis are:

1. Identify and examine privacy principles and underlying privacy safeguarding requirements for software requirement engineering.
2. Identify and examine privacy requirements and privacy services mapping, as expressed through standards, and eliminate redundant services.
3. Extend UML for engineering privacy into software systems.
4. Develop a prototype of a tool that will help software developers and designers to model and implement privacy in the early stages of the software development life cycle.
5. Illustrate how the tool can be used to embed important privacy services with use-cases from the health care sector.

1.3 Outline

Chapter 2 provides a literature review. Chapter 3 first reviews privacy requirements and privacy principles, and then presents a proposal on how to include privacy requirements and service controls in UML modeling. Chapter 4 then describes an implementation of our proposal in Microsoft's Visio modeling software while Chapter 5 demonstrates the use of our privacy modeling software. Summary and conclusions are presented in Chapter 6.

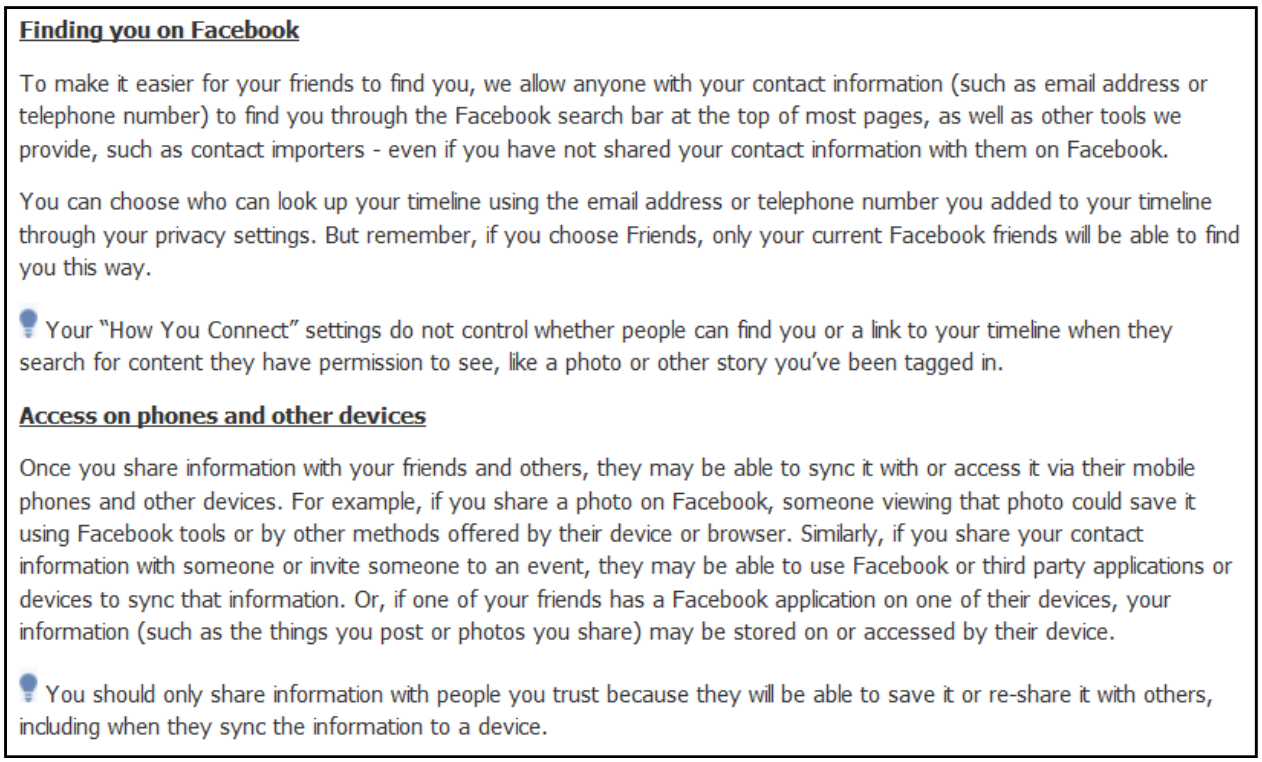
CHAPTER 2 : BACKGROUND AND RELATED WORK

With advancements in Internet technologies, organizations are expanding their business and are targeting more and more customers (Rodríguez & Mario Piattini, 2006). The customers are providing their valuable private and personal information to the organizations in order to benefit from their services. This personal and private data has become a leading concern for customers and users, and also for the organizations that need to comply with good consumers and privacy practices and regulations (Ghazinour & Barker, 2009). Organizations are promising sound ways of handling customers' personal and private data in their online privacy policies. However, having privacy policy does not guarantee a mechanism through which they are protecting personal information internally (Karjoth & Schunter, 2002). For the purpose of protecting customers' personal and private data, a number of technologies and tools are available to facilitate the software developers and designers to integrate privacy into the system. In this chapter we shed light on different research into various privacy-related issues.

2.1 Privacy Overview

A privacy policy is described as a statement, law, or a legal document that promises the fair use of information by an organization (Karjoth & Schunter, 2002). The document describes the ways an organization collects, stores, manages and uses the user's private information. It also describes what information will be kept secret, shared with business partners, or sold to third parties. Private information can be anything, such as name (first, last, middle), telephone number, mailing address, date of birth, credit card information, medical records, marital status, travel history, or any other piece of. Many countries have well defined privacy policy guidelines

and laws to protect the privacy of an individual, private sector, government operations, or enterprises. Individual organizations and enterprises have also defined regulations and policies to protect the privacy of their users and customers. Figure 2.1 shows the example of the Facebook's privacy policy (Facebook Data Use Policy, n.d.).



Finding you on Facebook

To make it easier for your friends to find you, we allow anyone with your contact information (such as email address or telephone number) to find you through the Facebook search bar at the top of most pages, as well as other tools we provide, such as contact importers - even if you have not shared your contact information with them on Facebook.

You can choose who can look up your timeline using the email address or telephone number you added to your timeline through your privacy settings. But remember, if you choose Friends, only your current Facebook friends will be able to find you this way.

Your "How You Connect" settings do not control whether people can find you or a link to your timeline when they search for content they have permission to see, like a photo or other story you've been tagged in.

Access on phones and other devices

Once you share information with your friends and others, they may be able to sync it with or access it via their mobile phones and other devices. For example, if you share a photo on Facebook, someone viewing that photo could save it using Facebook tools or by other methods offered by their device or browser. Similarly, if you share your contact information with someone or invite someone to an event, they may be able to use Facebook or third party applications or devices to sync that information. Or, if one of your friends has a Facebook application on one of their devices, your information (such as the things you post or photos you share) may be stored on or accessed by their device.

You should only share information with people you trust because they will be able to save it or re-share it with others, including when they sync the information to a device.

Figure 2.1 Facebook policy (Adopted from (Facebook Data Use Policy, n.d.))

The following subsections describe various aspects of privacy.

2.1.1 Information Privacy

Information privacy is a concept which deals with the collection and sharing of individual's information. The piece of information can be social security number, credit card number, bank account numbers and details, medical records, email address, home address, etc.

Information privacy can be broken down into many groups such as financial privacy, health record privacy, internet privacy etc. Failure or any violation in protecting an individual's information makes the user identity vulnerable or even put the user at risk.

2.1.2 Personal Privacy

Personal privacy deals with the right of an individual to be left alone or the right not to be disturbed unnecessarily at their office, home or anywhere.

2.1.3 Location Privacy

This type of privacy deals with how to prevent and safeguard the individual's current or past locations from third parties. It also deals with the issue of permitting access to the user's location by an automated agent in order for the agent to successfully provide a service, such as by a navigation system service (Beresford & Stajano, 2003). Location privacy must be carefully defined to simultaneously satisfy a user's desire to keep the location secret while providing a service that requires the user's location as an input parameter, such as finding out all restaurants that are close to the user's location (Beresford & Stajano, 2003).

2.1.4 Organizational Privacy

Companies, groups, organizations and agencies, whether public, private or government owned, want to keep their information, data and operation private and to safeguard them from other organizations or individuals.

2.1.5 Spiritual and Intellectual Privacy

Government protects the individual's spiritual and intellectual privacy with laws that clearly state and protect the individual's spiritual practices, feelings, and intellectual thoughts.

2.2 Privacy Technologies

Extensive research has been carried out in this area and this section describes the related work done in the field of privacy technologies.

2.2.1 eXtensible Access Control Markup Language (XACML)

XACML, an OASIS standard (A Brief Introduction to XACML, n.d.) defines a policy language and access control decision request and response language, which are expressed in an XML format. The XML based policy language is used to define the general access control requirements to access the protected resource ((A Brief Introduction to XACML, n.d.), (Yagüe, 2006)). It also provides extension points for implementing new functions, data types, and combining logic. In order to specify access control requirements, the policy language uses attributes that encode the different properties of subject, resource, action and environment (Md. Moniruzzaman, Ferdous, & Hossain, 2010). The access control request/response language facilitates the user in creation of a query to request whether an action performed by the user will be allowed and what will be the response (A Brief Introduction to XACML, n.d.). The response of the access control request can be one of the following four values: Permit, Deny, Indeterminate (indicates an error generated during the validation) or Not Applicable.

The privacy profile of XACML defines two attributes. The *resource purpose* defines the purpose for which the data resource was collected. The resource owner should be informed and obtain assent from owner for the use of the resource for a given purpose (Rissanen, 2010). The *action purpose* indicates the reason for using the data that is submitted by the user; it may be organized hierarchically (Md. Moniruzzaman, Ferdous, & Hossain, 2010). An access request is allowed only when the action purpose matches the resource purpose. Figure 2.2 is an example of a policy defined using XACML features.

```
<Policy PolicyId=”SamplePolicy”
          RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
                                                  algorithm:permit-overrides">
    <!-- This Policy only applies to requests on the SampleServer -->
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
                                                                  equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                           SampleServer
           </AttributeValue>
          <ResourceAttributeDesignator
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
      </Resources>
      <Actions>
        <AnyAction/>
      </Actions>
    </Target>
    <!-- Rule to see if we should allow the Subject to login -->
    <Rule RuleId="LoginRule" Effect="Permit">
      <!-- Only use this Rule if the action is login -->
      <Target>
        <Subjects> <AnySubject/>
        </Subjects>
        <Resources> <AnyResource/>
        </Resources>
        <Actions>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
                         DataType="http://www.w3.org/2001/XMLSchema#string">
                         login
            </AttributeValue>
            <ActionAttributeDesignator
```

```
                        DataType="http://www.w3.org/2001/XMLSchema#string"
                        AttributeId="ServerAction"/>
        </ActionMatch>
      </Actions>
    </Target>
    <!-- Only allow logins from 9am to 5pm -->
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-
                                                    greater-than-or-equal"
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-
                                                              and-only">
          <EnvironmentAttributeSelector
                          DataType="http://www.w3.org/2001/XMLSchema#time"
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-
                                                                  time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">
                        09:00:00
        </AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-
                                                          than-or-equal"
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-
                                                              and-only">
          <EnvironmentAttributeSelector
                          DataType="http://www.w3.org/2001/XMLSchema#time"

     AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">
                        17:00:00
          </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <!-- We could include other Rules for different actions here -->
  <!-- A final, "fall-through" Rule that always Denies -->
  <Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

Figure 2.2 XACML Privacy Policy Example
(Adopted from (Rissanen, 2010))

The Target attribute is used to define the policy that applies only to the requests for the server called "SampleServer". The rule of the policy has an action of "login" and a condition that will be applied only if the subject wants to log in between 9am and 5pm (Rissanen, 2010).

XACML is a standard tool for access control used by the software industry. It has been reviewed by the community, experts and users. It is widely deployed and easy to incorporate with other applications (A Brief Introduction to XACML, n.d.). It is a generic approach in that organizations are not trying to provide access control for a particular environment or some particular kind of resource; rather, XACML can be used in any environment (A Brief Introduction to XACML, n.d.). This standard is also distributed and powerful because many policies, which are kept at different locations, can be combined into one decision. Figure 2.3 shows the XACML editor screen developed in Java by the University of Murcia (Morcillo & Lázaro, 2012).

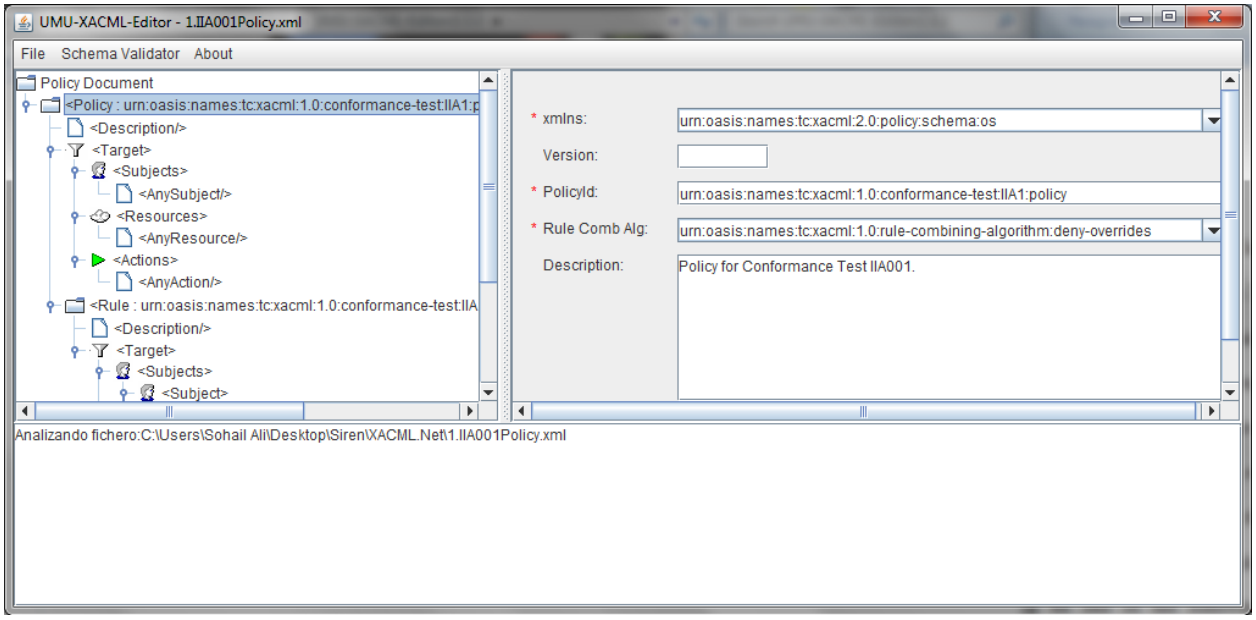


Figure 2.3 XACML Editor provided by UMU (Adopted from (Morcillo & Lázaro, 2012))

XACML has several limitations. A policy language is complex and wordy. In order to encode a simple policy, many lines of code and expertise are required (Md. Moniruzzaman, Ferdous, & Hossain, 2010). It does not accommodate domain specific permission types

(Naedele, 96-98). XACML also does not facilitate, for the software engineers, designers, or architects, graphical modeling and embedding of privacy into their systems.

2.2.2 Enterprise Privacy Authorization Language (EPAL)

Enterprise Privacy Authorization Language (EPAL), developed by IBM, is a formal language to write enterprise privacy policies (Ashley P., Hada, Karjoth, Powers, & Schunter, 2003). Privacy policies expressed in EPAL are used to enforce the organization privacy policies through the enforcement engine (Md. Moniruzzaman, Ferdous, & Hossain, 2010). EPAL's main focus is on the core privacy authorization while keeping data models and user authentication on the abstract level away from deployment details (Ashley P., Hada, Karjoth, Powers, & Schunter, 2003). EPAL policy is similar to access control rules or permissions. It is based on XML, similar in concept to XACML, and it contains a set of attributes that define a privacy vocabulary (Md. Moniruzzaman, Ferdous, & Hossain, 2010). A schema file is used to define the semantics of the vocabularies. The main components of EPAL policy are target, condition, and obligation (Md. Moniruzzaman, Ferdous, & Hossain, 2010). The target consists of data categories, data users, a set of privacy actions, obligations, and conditions (Ashley P., Hada, Karjoth, Powers, & Schunter, 2003). Data user includes the categories/entities that will consume the collected data. Data categories are applied to the collection of data that helps to organize the data differently from a privacy point of view, e.g., medical record vs. contact data (Ashley P., Hada, Karjoth, Powers, & Schunter, 2003). The purpose is to map the intention of data utilization. Action in a policy is defined as a task that will be performed on the data, e.g., read, write and update. Obligations are used to define the actions that must be implemented by the EPAL environment,

such as delete after 20 days. Conditions are defined as boolean expressions that must be satisfied for the rules to be executed (Yu & Murthy, 2007).

Figure 2.4 shows an EPAL vocabulary information. It defines the information of the sub category into three sub elements. The attribute "id" defines the vocabulary name, <issuer> identifies the issuer, and <version-info> defines the version and the management information (Ashley P., Hada, Karjoth, Powers, & Schunter, 2003). EPAL has no proper framework for the enforcement of obligation policies. The whole model does not provide the role hierarchy concept for encoding enterprise policy (Md. Moniruzzaman, Ferdous, & Hossain, 2010). EPAL also fails to define the role hierarchy concept, which limits the organization policy encoding. This model also fails to provide the visual aspect early in the software development phase. EPAL was abandoned as XACML provided a superior specification for access control (Ashley P., Hada, Karjoth, Powers, & Schunter, 2003).

```
<vocabulary-information id="sample-policy">
<short-description language="en">Example policy information</short>
<long-description language="en">long-description</long-description>
<!-- Entity issuing the policy -->
<issuer>
<name>EPAL Working Group</name>
<organization>IBM Research</organization>
<e-mail>testmail@zurich.ibm.com</e-mail>
<address>Saumerstr. 4, 8838 Ruschlikon</address>
<country>Switzerland</country>
</issuer>
<version-info start-date="2002-03-19T00:00:00" test="true"
last-modified="2002-03-19T00:00:00" revision-number="1.02A"
end-date="2004-03-19T00:00:00"/>
</vocabulary-information>
```

Figure 2.4 EPAL Vocabulary Information

2.2.3 Platform for Privacy Preference (P3P)

Platform for Privacy Preference (P3P) allows different websites to state their privacy practices in a well-defined format that can be easily interpreted by user agents (Wenning, 2007). P3P enabled browsers can fetch the policy automatically and can compare it with the user's set of privacy preferences (Karjoth & Schunter, 2002). This model has developed standards for the user agents in order to automatically examine the privacy policy of e-business (Bodorik & Jutla, 2008). As a result, users need not to read the privacy policies of each website they visit (Wenning, 2007). P3P policy consists of an XML document that defines the practices of data collection for a site (Karjoth & Schunter, 2002). Furthermore, P3P contains a base schema for collected data and vocabulary for expressing purposes, recipients and retention policies (Karjoth & Schunter, 2002).

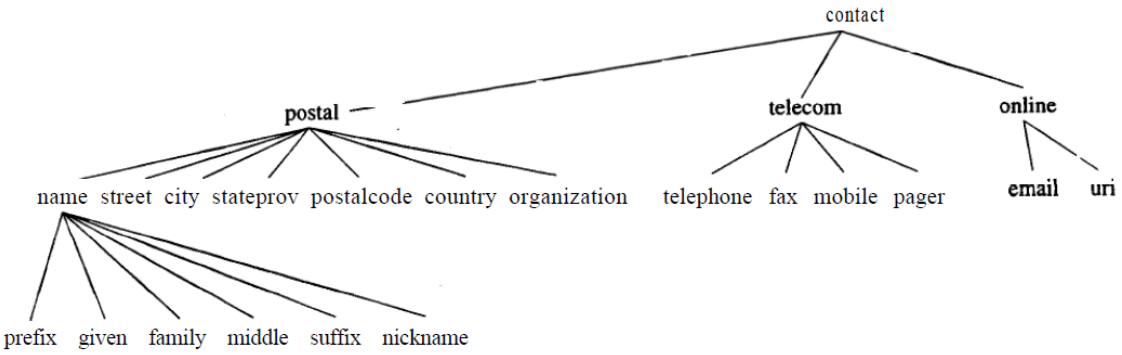


Figure 2.5 Basic Data Structure of P3P (Adopted from (Wenning, 2007))

Figure 2.5 shows the basic data structure of P3P whose elements are organized into a hierarchy. A chosen level's data element groups all the similar data elements in its hierarchy (Karjoth & Schunter, 2002). This provides a more convenient way to organize group of related

data elements. There are several tools available to define P3P Policies. JARC Policy Workbench is an API for building and editing policy and the testing environments (see Figure 2.6).

P3P does not deal with privacy policies' compliance, i.e., it does not deal with checking whether an organization's business process complies with the organization stated privacy policies (Bodorik & Jutla, 2008). Furthermore, it does not provide the procedure to check an access request against the stated privacy policy (Karjoth & Schunter, 2002).

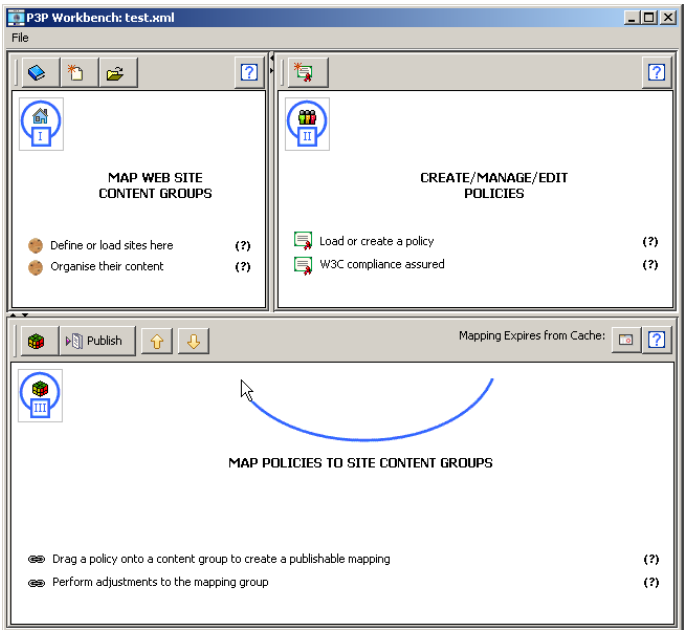


Figure 2.6 JRC Policy Workbench

2.2.4 Server Privacy Architecture and Capability Enablement (SPARCLE)

SPARCLE (*Server Privacy Architecture and Capability Enablement*) was developed by Brodie et al. to facilitate privacy policy creation, implementation, and compliance monitoring (Brodie, Karat, Karat, & Feng, 2005). This tool helps users to create privacy policies,

relationships, and access rules using natural language or well defined (structured) format that are then translated into machine-readable format. SPARCLE allows the user to edit either of the formats and keeps the formats synchronized. The authored policies can be translated into machine-readable formats such as XACML and EPAL (Yu & Murthy, 2007). SPARCLE parses the user-authored policy (in natural language) using parser and predefined dictionaries. There are three major sections in SPARCLE: 1. *Author* – allows user to write policies in natural language, 2. *Transform* – transforms the input policies into machine -readable format; and 3. *View* – allows the user to verify the generated rules. Tools based on SPARCLE do not allow users to automatically verify the generated rules; once the rules are generated users have to manually check them. None of the tools allow a user to visually model the privacy policy rules. Figure 2.7 shows the screenshot of SPARCLE policy workbench – a prototype of SPARCLE.

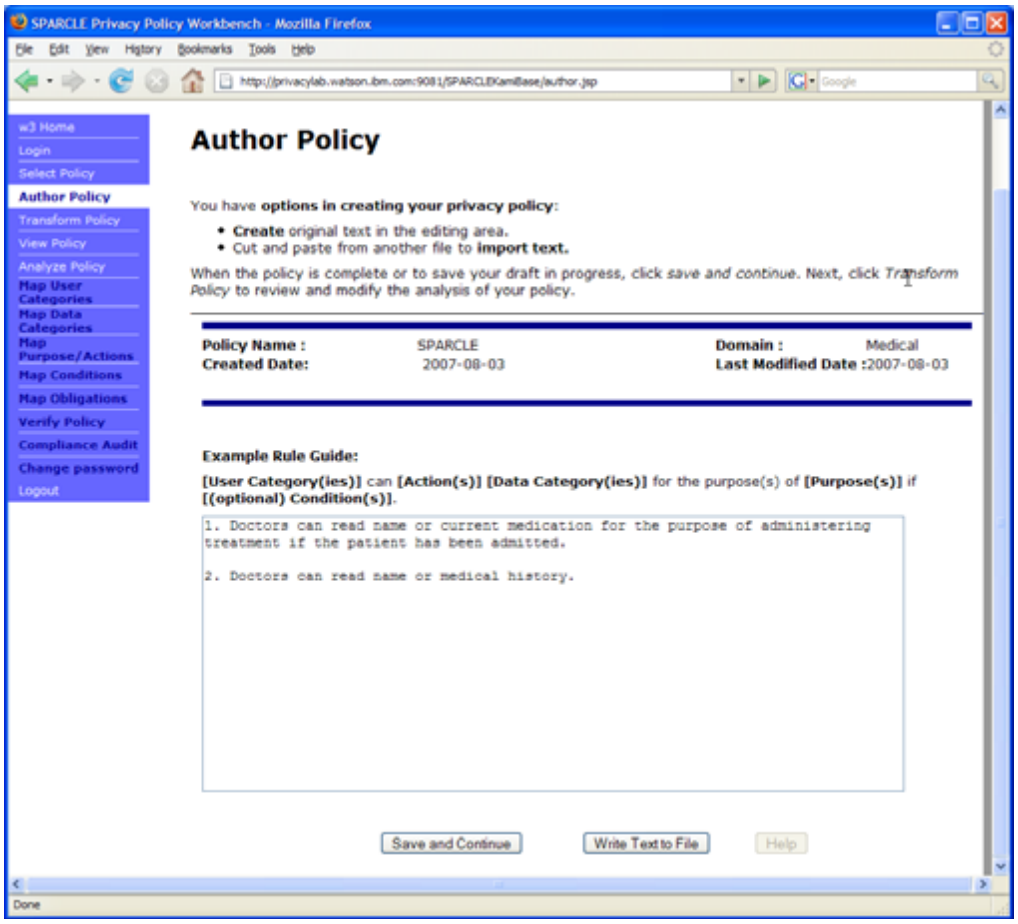


Figure 2.7 SPARCLE Policy Workbench - Author Policy Screen

2.2.5 Role Based Access Control (RBAC)

Role Based Access Control (RBAC) was first proposed by Ferraiolo et al (Ferraiolo & Kuhn, 1992). This model assigns the roles to users or a group of users based on their designation and responsibility. Permissions, which give access to the data items, are granted to the users based on their role. The major drawback of this model is that it was not designed keeping in mind privacy of data subjects and thus failed to express privacy preferences and privacy policies under which the private data was collected.

Ni, Bertino, and Lobo (2008) propose a model called P-RBAC, a privacy preserving access control model. This model defines the basic components of the privacy policy as data *d,* action *a,* purpose *p,* condition *c,* and obligation *ob*. P-RBAC expresses a privacy policy based on the access model for a given action *a* on data *d* for a given purpose *p* meeting the obligation *ob* and condition *c*. For a user to get access, privacy requirements should be satisfied (Md. Moniruzzaman, Ferdous, & Hossain, 2010). Many models have been proposed based on RBAC, such as PBAC – a purposed based access control model for relational databases, proposed in (Byun & Li, 2008), and TPBAC, a transaction based PBAC, proposed in (Yang, Barringer, & Zhang, 2007).

2.2.6 Hippocratic Database

A Hippocratic Database model, based on a relational database system, was proposed in (Agrawal et al., 2005). Privacy preferences are generated using the P3P model at the time of data collection and these privacy preferences are translated into a schema file called T1, using a policy translator module. T1 is a collection of policies that describes which recipient can use what data for what purpose. This model allows a user to create a constraint, called *restriction,* on data items based on policies in the T1 schema file. When a constraint is satisfied, access to a data item is granted. Figure 2.8 shows a sample restriction constraint. Privileges to data users, e.g. *grant,* and are specified through the SQL syntax. This model determines which cell or column user can access and blocks any illegitimate request because a constraint allows access for the right purpose only.

```
Create restriction name_of_restriction
on table_T
for authors_1_name [ except authtors_2_name]
(((to columns name_of_columns) | (to rows [where searching_condition])
                                | (to cells (list_of_columns[where searching_condition])+) )
[ for purpose list_of_purposes]
[ for recipient list_of_recipients])+
restricting access to (all | (select | update | insert | delete)+ )
```

Figure 2.8 Hippocratic Database language constructs (Adopted from (Agrawal et al., 2005))

2.2.7 Privacy Policy Visual Model (PPVM)

The main purpose of a Privacy Policy Visualization Model (PPVM) (Ghazinour, Majedi, & Barker, 2009) is to assist the data owner to understand the policies, as well as provide means to the policy officers to better understand the designed policies. This model is based on a use-case, which is drawn from the existing privacy policies (Ghazinour, Majedi, & Barker, 2009). Kambiz, Maryam and Ken (2009) presented this model to provide communication assistance with the privacy concerns of data owners and collectors. Although, different Privacy Policy (PP) frameworks have been proposed by researchers, and various policy languages, such as EPAL (Ghazinour, Majedi, & Barker, 2009), (Sandhu, Ferraiolo, & Kuhn, 2000) and P3P (Wenning, 2007), (Cranor, 2002) have been used, they do not provide visual modeling. By using these frameworks, most of the organizations publish their privacy policies online in a machine-readable format. When the user browses to their websites, the browser pulls these policies from

the website and matches it with the user's privacy preferences. If privacy violation occurs, a browser alerts the user. P3P provides a simple mechanism to understand the privacy policy instead of exhaustively searching through the entire website. Despite of its usefulness, there are some problems in this approach. The user can take in many pages of the privacy policy statements that are not easy to understand from the outside world. To overcome this problem, Policy Privacy Visualization Model (PPVM) was proposed (Ghazinour, Majedi, & Barker, 2009).

Figure 2.9 shows a PPVM diagram. PPVM model represents each privacy policy component with different symbols: data collector is represented by a house; entities and attributes are shown by rectangles and ovals; a relation is expressed by a line segment; privacy policy is represented by a note symbol with predicates {*P: purpose, G: granularity, V: visibility, R: retention*, *C: constraint}*; group attributes are expressed by drawing circle around attributes; and default values are shown by text following a predicate. PPVM is inspired by the Entity Relationship Model (ER) to demonstrate the privacy policies that are defined by the organization. PPVM has its own collection of symbols and that does not correspond to software design modeling languages, such as UML. Furthermore, these diagrams are drawn separately from existing use-case diagrams.

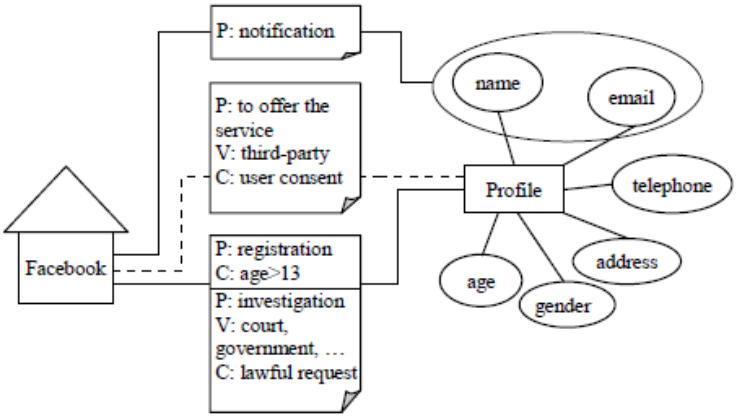


Figure 2.9 PPVM example diagram (Adopted from (Ghazinour, Majedi, & Barker, 2009))

2.2.8 Privacy Policy Modeling Language Processor (PPMLP)

Privacy Policy Modeling Language Processor (PPMLP) (Yu & Murthy, 2007) is based on the Service Oriented Architecture (SOA). PPMLP was developed by Weider and Savitha (2007) to help organizations to define comprehensive privacy policy documents. PPMLP models the structure and contents of the organization's privacy policies and converts it into a Privacy Policy Template (PPT) that is understandable by the machine (Yu & Murthy, 2007). The foundation of a template is based on the meta version of Privacy Policy Specification expressed in XML – which defines the content and schema of the Privacy Policy. After creating the template, it is then parsed and verified by a Privacy Policy grammar. In order to ensure the completeness of the template, its contents are cross-checked against the privacy principles. Natural language privacy policy can be generated by substituting the template words in PPT. The PPT processor translates only the EPAL syntax tree data into machine-readable format that will become the input of the policy enforcement engines (Yu & Murthy, 2007).

The PPT generation component takes the Privacy Policy Specification file as input as shown in Figure 2.10. The user will input the data for each of the headings and then the Privacy Policy Parser will use the Privacy Policy Grammar file rules for parsing the user input as shown in Figure 2.11 which is in BNF format. The Natural Language Policy Generation component will generate the Natural Language Privacy Policy file by using the input Privacy Policy Template file. The EPAL Policy Generation components will generate the EPAL rules by transversing the tree, which was built during Privacy Policy Template Generation phase. (Yu & Murthy, 2007).

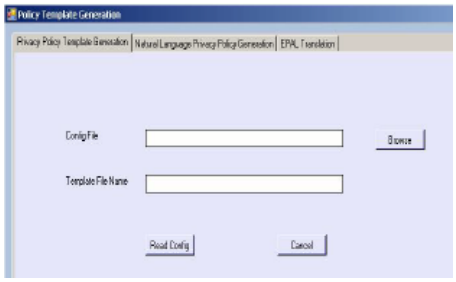


Figure 2.10 GUI prototype of PPMLP Architecture.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<xml>

<element>
  <label>Organization Title</label>
  <line> 1 </line>
  <position> center </position>
</element>

<table>
  <row>1</row>
  <column>1</column>
  <element>
    <label>Policy Title</label>
    <line> inc </line>
    <position> left </position>
  </element>

  <row>2</row>
  <column>1</column>
  <element>
   <label>Approved By</label>
   <line> inc </line>
   <position> left </position>
  </element>
```

Figure 2.11 Sample Meta Privacy Policy specification

```
<privacy_policy>      ::=   <organization_name> <sections>  ;

<organization_name>   ::=   <words> <newlines>    ;

<newlines>            ::=   <newlines> EOL    |   EOL   ;

<title_value>         ::=   <heading> <words> <newlines>    ;

<name_value>          ::=   <heading> NAME_TITLE <words>
<newlines>
                      |     <heading> <words> <newlines>    ;

<revision>            ::=   <heading> DIGIT '.' DIGIT <newlines>    ;

<sections>            ::=   <sections> <section>    | <section>    ;

<section>             ::=   <heading> <body> '.' <newlines>   ;

<body>                ::=  <paragraphs>
                      |     DIGIT '/' DIGIT '/' DIGIT '.'   |   DIGIT '.'
DIGIT '.'
                      |     NAME_TITLE <words> '.'   ;

<heading>             ::=   <words> '.'    ;

<paragraphs>          ::=   <paragraphs> PARA_END <statements>
                      |     statements    ;

<statements>          ::=   <sequence_list>     |   <enum_list>
                      |     <sequence_list> EOL <enum_list>
                      |     <enum_list> EOL <sequence_list>
                      |     PARA_END   ;
```

Figure 2.12 Portion of Privacy Policy Grammar

2.3 Privacy Implementation Approaches

Different models are practices by organizations to implement privacy components in software systems, which are described below:

2.3.1 Privacy by Architecture

FIPs principles are not strictly taken into account when a system is developed based on a privacy friendly architecture. In privacy friendly architecture, organizations minimally collect

personal information from their customers. Customers become powerful, or in control, because data processing occurs at customers' machines instead of organization's infrastructure. Thus, eliminating the need to transfer and storing of data in an organization's database. This significantly reduces the secondary usage of customers' data.

In the extreme case, if an organization decides to refrain from collecting personal information or bases its business on unidentifiable users, it is not mandatory to provide a customer with notifications and choices. There are systems based on this approach, such as the collaborative filtering system proposed in (Canny, 2002), in which individuals' private data is stored on their own systems and an '*aggregate*' is computed on their data, which can later be shared with others.

2.3.2 Privacy by Policy

This approach is completely opposite of privacy-by-architecture. FIPs comes into play when an organization decides to implement FIPs principles by making customers feel comfortable about their private information, providing adequate security mechanism and giving customers enough degrees of control over their information. This approach is called as *privacy-by-policy*. This approach is widely practiced in software industry by those who collect individual's personal information. To use any of these approaches, or hybrid of the two, depends on the customers' concerns, business requirements, or government regulations etc.

2.4 Discussion

The common problem with all of the models discussed above is that they do not facilitate software developers and engineers to visually model privacy processes early in the software development cycle, i.e., at the time of requirements specification or at the time when requirements are transformed into the use-cases. If privacy is not taken into account at the beginning of software development phase, it becomes very hard for the developers to integrate privacy in later phases, resulting in increased development, testing, and bug fixing costs.

CHAPTER 3 : PRIVACY SERVICES IN UML

We discussed a number of approaches to modeling software and requirements in Chapter 2, but none of these are capable of providing visual models for privacy to be used at the early phases of software development. Many of the technologies as discussed in Chapter 2 require developers to have extra expertise to master them.

In this thesis, we flesh out a section of a novel approach (Jutla, 2012a, b, c, d) to solve the modeling problem of embedding privacy requirements during the early phases of software development, i.e., at the time of requirements modeling that utilizes use-case and other commonly-used UML diagrams. Use-case diagrams model the requirements rather than writing down the requirements in a natural language, which may be confusing and prone to ambiguity, depending on the writers. The proposal is to extend UML to model privacy requirements since UML is attractive and expressive and has a wide user-base in the software industry (Sun, France, & Ray, 2011). This work will help software developers to easily incorporate the privacy requirements from the beginning of the software development life cycle.

Generally, privacy-friendly systems are grouped into two categories: Privacy by Policy and Privacy by Architecture (Spiekermann & Cranor, Engineering Privacy, 2009). These two approaches are based on Fair Information Practice Principles (FIPs). Here we extend UML use-case diagrams to support privacy controls or services that can be used by software engineers to represent privacy requirements. The model is based on FIPs, OASIS' Privacy Management Reference Model and Methodology (PMRM) privacy services, and follows the recommendations

of the Privacy By Design principles (Cavoukian, 2013) for embedding privacy by design. The developed prototype will help engineers to integrate privacy requirements early into software systems based on FIPs and PMRM requirements and services.

In the following subsections, we first review the FIPs privacy principles and PMRM model and methodology in sections 3.1 and 3.2 respectively. We examine, in section 3.3, the FIPs principles and PMRM model for commonalities and determine a set of privacy services that cover both, that is privacy requirements of FIPs principles and PMRM model, but eliminating redundancies when requirements appear in both. We propose an extension to the UML use-case by a privacy container in section 3.4, and describe its significance using an example in section 3.5.

3.1 Privacy Requirements Identification for Software Engineers

The US Federal Trade Commission proposed Fair Information Practices (FIPs) – guidelines that focus on minimizing personal data collection, communicate to the user about collection of data, and effectively maintaining the collected data (Spiekermann & Cranor, Engineering Privacy, 2009), (Pitofsky, Anthony, Thompson, Swindle, & Leary, 2000). These principles focus on fair information practices in all information systems: electronic, manual and hybrid. Privacy policies revolve around FIPs principles which are discussed below.

3.1.1 Notice

Users should be given clear notice about entity's privacy policies and practices before any information is collected which includes: what piece of information is collected, how it is

collected, how it is used (purpose), whether it is disclosed to other entities either internal or external, how it is protected, when it is deleted, how long it is retained, and how its confidentiality and integrity is ensured (Spiekermann & Cranor, 2009).

3.1.2 Choice

Choice means providing consumer options as to how their personal information is used beyond the use for which it was collected. There are typically variations of two choices Opt-In and Opt-Out. In Opt-In, consumers explicitly grant permission to use their information for secondary purposes, i.e., the user has to provide consent and information cannot be used for purposes that are not mentioned in the notice. In Opt-Out option information can be used for purposes other than primary, (Spiekermann & Cranor, 2009), and users have to take action to opt out of personal information collection and/or sharing by the organization with which he/she is interacting.

3.1.3 Consent

Consumers express their assent or approval after thoughtful consideration in different forms, such as digital signature or written note. Some systems have sophisticated consent management processes. Different types of consent include “explicit” and “implied”. They are the opposite of each other. Implied consent mechanisms often mean that the user is unaware of what he or she is consenting to.

3.1.4 Access

Consumers should be allowed reasonable access to the information collected about them and the ability to review, correct errors, or inaccuracies and delete the information, (Spiekermann & Cranor, 2009).

3.1.5 Security

Collected information should be safeguarded against external and internal security threats. It should be ensured that information collected is secured and confidential. Users should also be notified that personal information will be destroyed after a particular time, (Spiekermann & Cranor, 2009).

Fair Information Practices focuses on minimizing the collection and use of personal information, informing individuals about collection of data, and properly protecting and maintaining the collected data (Spiekermann & Cranor, 2009). Organizations draft their privacy regulations based on FIPs.

3.2 Privacy Management Reference Model and Methodology (PMRM)

The Privacy Management Reference Model and Methodology (PMRM) (Sabo, Willett, Brown, & Jutla, 2012, 2013), helps organizations to minimize the complexity of managing the customers' information in today's networked environment. It also helps the organizations to improve privacy management and compliance in day-to-day business where customers' information is protected by laws and regulations, and where the technologies discussed in chapter 2 are not sufficient. The foundation of this model is derived from FIPS.

The PMRM is used to analyze, understand complex, and composite use-cases and to derive and implement appropriate privacy functionality, and to attain system wide privacy compliance. This model also helps in the selection of incorporated methods capable of running privacy controls in line with organizations privacy policies (Sabo J. , Willett, Brown, & Jutla, 2012). This model's service functionality is unaffected by the barriers such as multiple jurisdictions, regulations, business laws and practices, and often conflicting laws etc. (Sabo J. , Willett, Brown, & Jutla, 2013).

The PMRM provides a standards-based model and is neither a static model nor a fixed set of defined rules. Software engineers, architects and developers have flexibility in implementing privacy and security policies (Sabo J. , Willett, Brown, & Jutla, 2013). This model helps to overcome a few of the shortcomings of technologies discussed in Chapter 2. The PMRM model serves as an analytical tool and helps to assess the completeness of proposed privacy in the actual business process and it also helps in the analysis and design of the system functionality that are required to implement a set of privacy requirements. This model also provides stakeholders, such as software architects, developers and policy makers, a tool to embed good privacy management and compliance in software systems (Sabo J. , Willett, Brown, & Jutla, 2012).

3.2.1 PMRM Services

The PMRM Services bridge the gap between the organization privacy and business requirements by providing constraints on system processes which deal with users' private information (Sabo J. , Willett, Brown, & Jutla, 2013). These services aim to support an arbitrary

set of privacy policies at the functional level of software development phase. These eight services can be logically classified into three groups as shown in Table 3.1 (Sabo J. , Willett, Brown, & Jutla, 2013).

The logical grouping of services elucidates the relationship in an operational design of the software system. Functionality provided by all services can be used by software developers, architects or policy makers without any restrictions and is also used for mutual interaction. The functionality of one service can be used by other services to accomplish a privacy management operation. The PMRM eight services and their functionalities are defined in Table 3.2, which is replicated from the Privacy Management Reference Model and Methodology (PMRM) Version 1.0 working draft 4 (Sabo J. , Willett, Brown, & Jutla, 2013).

Table 3.1 Logical grouping of eight PMRM Services

Core Privacy Services	**Privacy Assurance Services**		**Presentation and Lifecycle Services**
Agreement	Validation	Certification	Interaction
Usage	Security	Enforcement	Access

Table 3.2 Eight PMRM Services with functionality

Service	**Functionality**	**Purpose**
Agreement	• Define and document permissions and rules for the handling of PI based on applicable policies, individual preferences, and other relevant factors; • Provide relevant actors with a mechanism to	Manage and negotiate permissions and rules.

	negotiate or establish new permissions and rules; • Express the agreements for use by other Services and different organization roles.	
Usage	• Ensure that the use of PI complies with the terms of any applicable permission, policy, law or regulation, • Including PI subjected to information minimization, linking, integration, inference, transfer, derivation, aggregation, and anonymization, • Over the lifecycle of the use-case.	Control PI use.
Validation	• Evaluate and ensure the information quality of PI in terms of Accuracy, Completeness, Relevance, Timeliness and other relevant qualitative factors	Check PI.
Certification	• Ensure that the credentials of any Actor, Domain, System, or system component are compatible with their assigned roles in processing PI; • Verify compliance and trustworthiness of that Actor, Domain, System or system component against defined policies and assigned roles.	Check credentials.
Enforcement	• Initiate response actions, policy execution, and recourse when audit controls and monitoring indicate that an Actor or System does not conform to defined policies or the terms of permission (agreement).	Monitor and respond to audited exception conditions.
Security	• Provide the procedural and technical mechanisms necessary to ensure the confidentiality, integrity, and availability of personal information; • Make possible the trustworthy processing, communication, storage and disposition of privacy operations.	Safeguard privacy information and operations.
Interaction	• Provide generalized interfaces necessary for presentation, communication, and interaction of PI and relevant information associated with PI; • Encompasses functionality such as user interfaces, system-to-system information exchanges, and agents.	Information presentation and communication.

Access	• Enable data-subject Actors, as required and/or allowed by permission, policy, or regulation, to review their PI that is held within a Domain and proposes changes and/or corrections to their PI.	View and propose changes to stored PI.

3.3 Privacy Services to support FIPs and PMRM

We examined the FIPs requirements and PMRM services and discovered some overlap. For instance, the Security requirement of FIPs matches closely the PMRM Security Service. Elimination of such redundancy leads to the following set of services that will support the FIPs requirements and also cover the privacy services proposed by the PMRM model: Notice, Agreement, Consent, Access, Certification, Security, Interaction, Usage, Validation, and Enforcement. We now present our proposal of how these services are to be represented in UML.

3.4 A Proposal for A Ribbon with Privacy Icons

Privacy by Design principles (Cavoukian, 2013) state that the software development team should incorporate privacy in the early software development phase as an integral part of the software solution. Our proposal supports these principles by helping the software engineers/developers to incorporate privacy right in the requirements phase of the software development life-cycle. Furthermore, it may help auditors to verify an organization's compliance to best privacy practices.

We created a set of privacy controls for the UML use-case in order to visually model privacy requirements. The controls are defined in Table 3.3 and, as was described in the previous subsections. They are based on Fair Information Practices Principles (FIPPs) and PMRM.

Table 3.3 Privacy Services

Privacy Service Name	Proposed Icon
Notice	
Agreement	
Consent	
Access	
Certification	
Security	
Interaction	
Usage	
Validation	
Enforcement	

In Table 3.3, the icon column shows a prototype of an icon for each privacy control, which is to be used by software developers to specify related privacy functionality. Similarity of icon colors shows their logical grouping. For example, all the yellow icons refer to the family of Access control mechanisms that may be deployed to protect privacy. Blue icons are used for notice, agreement, and consent services. They belong to one family that has closely related

members. In some implementations, agreement and consent directives are the same. Notice mechanisms closely preceding consent mechanisms are representative of a good practice in implementations. An illustration of the Privacy ribbon for Microsoft Visio 2010 is given in Figure 3.1.

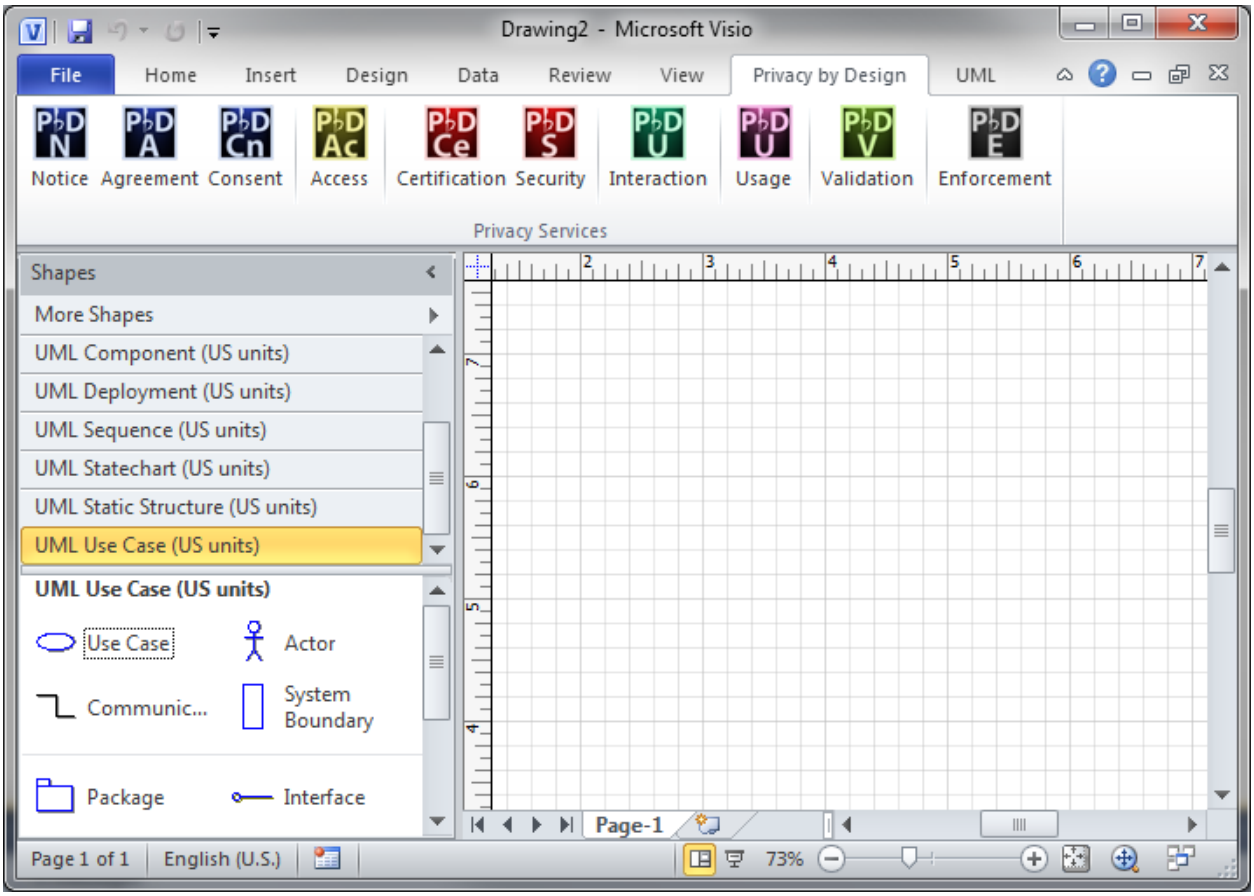


Figure 3.1 Privacy Icons ribbon in Microsoft Visio 2010

3.4.1 Notice, Consent, and Security in Use-Case Diagrams

For the purpose of a proof-of-concept prototype, we cover the Notice, Consent and Security control components in more detail. When a user clicks on a control shown in Table 3.3, he/she is provides with the following screen as shown in Figure 3.2, in which the user clicked on the Consent control. The screen is used by the software architect to input further details relevant

to the control/component. In Figure 3.2 for instance, the user can select a service related to the Consent control component.

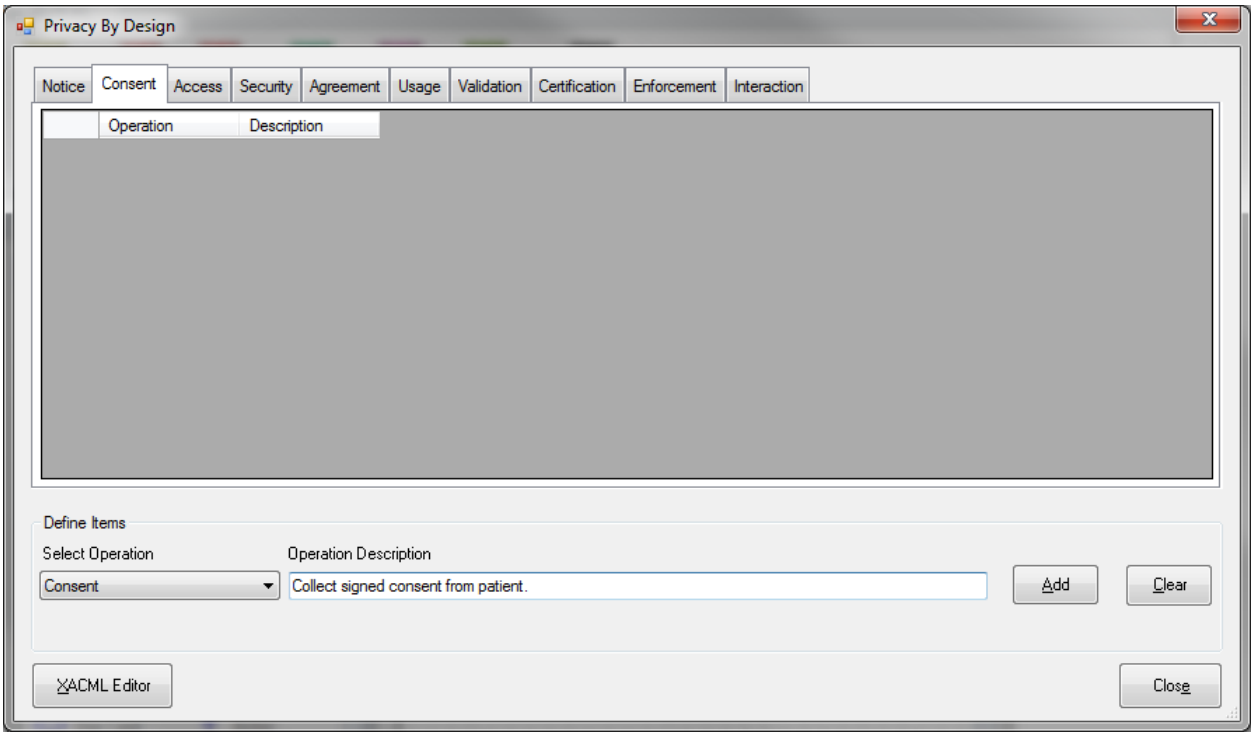


Figure 3.2 User Input screen for Consent privacy service.

Example operations supported by each of the Notice, Consent, and Security components are shown in Table 3.4. The table in this example refers to the NIST controls (Ross et al, 2012) for US Federal Information Systems.

Table 3.4 Privacy Components

Component	Operation and its purpose
Notice	• **Privacy Notice** The purpose of this operation is to present a detailed notice to system users and to individuals to:

1. Specify the organization's purpose for collecting personally identifiable information (PII) and how the organization will use it internally.
2. Communicate to the user, whether the PII collecting organization will share the PII with other third party entities and the purpose for sharing.
3. Describe whether an individual has the ability to consent to the specific use or sharing of PII.
4. Highlight activities that will impact user or individual privacy. This includes private information collection, usage, sharing, maintenance, safeguarding and appropriate ways of disposal of PI information.
5. Provide authority to the organization for collecting PII.
6. Provide choices to users and individuals, if any, individuals may have regarding how the service providing organization uses PII and outcomes of choosing those choices.
7. Allow users and individuals to access and correct their PII if required.
8. A medium to notify users about the changes in practices or policy that effect PII.

The purpose is to provide effective notice to the user or consumer, by virtue of its readability, clarity and comprehensive. This will enable individuals to understand how an organization consumes PII and help them to make an informed decision prior to providing PII. Effective notice demonstrates the organizations practices and consideration in implementing its policies and practices. Notices can be provided in different ways to users, which includes: publishing on website, paper based or electronic forms etc. (Ross et al., 2012)

- **System of Records Notices and Privacy Act Statements**

  The purpose of System of Records Notices (SORNs) is to provide notice and to educate users regarding collection process of PII in a system of records. The Privacy Act define this as "*a group of any records under the control of any agency from which information is retrieved by the name of an individual or by some identifying number, symbol, or other identifier.*" (Ross et al., 2012). The purpose of this control is that:

  1. The organization has to be consistent with the Privacy Act and information system containing PII, published in the SORNs and Federal Register (FR).
  2. Maintain SORNs up to date and current.
  3. Publish Privacy Act Statements (PAS) on all forms that collect PII from users. It also allows user to retain a copy of PAS before disclosing PII to organization.
  4. SORNs outline the process of how the collected information is retained, used, if required how it will be corrected and which portions of the systems are excluded from the Privacy Act for national security reasons.
  5. PAS is used to provide notice of:
     a. Organizations authority to collect PII.
     b. Provided PII is mandatory or optional

	c. The sole purpose of PII collection d. For what purpose it will be used e. The consequences of not providing all or part of the information requested by the organization. (Ross et al., 2012) • **Dissemination of Privacy Program Information (PPI)** Organization will adopt different ways and mediums for informing and education their users about their privacy practices. It is not limited to PAS, SORNs, privacy reports, publicly available webpages, mass email distribution, maintaining public blogs, posting on social media. Organization also adopts ways such as phone lines or direct question answer sessions etc. (Ross et al., 2012).
Consent	• **Authority to Collect** "This operation will notify the user about the legal authority that permits the collection, maintenance, use and sharing of PII with third parties. The collection authority is well documented in the SORNs. This authority to collect PII is consulted with Chief Privacy Officer and legal counsels." (Ross et al., 2012). • **Purpose Specification** "The purpose is to help the organization to describe the purpose(s) for which PII is collected, maintained, shared and used in its privacy notices." (Ross et al., 2012). • **Consent** The consent operation plays a fundamental role in the participation of an individual in the decision making process regarding the collection and use of their PII. This control also provides the details about the technology used during collection process. To collect consent, organizations provide notices of the purpose to all the individuals. Consent can be obtained through opt-in, opt-out, digital signature, or implied consent. The preferred method is opt-in, but it is not always feasible (Ross et al., 2012).
Security	• **Inventory of Personally Identifiable Information** This control helps the organization to follow out effective security procedures and policies to safeguard their customers PII and methods to mitigate the PII risks (Ross et al., 2012). The major purpose is to: 1. Update and maintain a database of all the programs and system identified as collecting, maintaining, sharing and using PII. 2. CIO or information system offices should be provided with each update of the PII inventory or database to support the establishment of information security requirements for all new or modified system used in collecting and storing PII. 3. Helps to extract the information elements from Privacy Impact Assessments of systems containing customers PII.

	a. Name, acronyms or alias of identified systems. b. Type of customers or users PII contained in the system. c. Classification of all types of PII. d. Classification of level of sensitivity and potential risks associated with each type of PII • **Privacy Incident Response** The purpose is to (Ross et al., 2012): 1. Help to develop and implement a privacy incident response plan. 2. Formulate an effective response in accordance with the organization Privacy Incident Response Plan in case of privacy incidents. Privacy Incident Response Plan includes: a. Building a Privacy Incident Response Team that formulates reviews, approves and actively participates in the execution of the response plan. b. If required, provide notice to the effected individuals. c. Execution of the privacy risk assessment process to determine the extent of harm, inconvenience, or unfairness to the effected individuals. Take appropriate steps to mitigate such risks and harms. d. Prompt reporting of privacy incident to the security officers/Chief Privacy Officer. • **SSL Implementation** The objective is to secure all communication between server and client by trusted third party issued certificate.

We gave more details of the Consent, Notice and Security privacy components in Table 3.4, but for the other privacy services, namely Agreement, Access, Certification, Interaction, Usage, Validation and Enforcement, we have only provided the basic controls without more detailed examination of the specific services that are required. Note that we also provide a way for the software developers and designers to easily input the operation name for a privacy service in a control and also its description by choosing "Other" option from Selection Operation dropdown as shown in Figure 3.2. From the design and implementation point of view we created the foundation layers so that anyone who wants to extend this work could simply plug-in the remaining privacy services' operations and their descriptions.

3.5 Privacy Controls in UML Use-Case Diagrams

In the previous section we described our proposal to visually represent privacy controls in UML. In this section we describe their representation and use in modeling privacy requirements in use-cases.

Figure 3.4 shows the Security control represented in UML using a container for security mechanisms, e.g. the SSL operation along with a brief description. Consider a software engineer creating a use-case to visually represent the software requirements. The engineer chooses controls, also referred to as components, from a stencil and drops them onto the use-case canvas and then draws connections/communication-lines between the various components. We now describe how a privacy control is represented on the canvas. The basic visual layout to represent a privacy control is the same for all 10 privacy services. The header displays the name of the control while the content of the box is used to display the name and description of the particular privacy service operation selected by the software engineer. Figure 3.4 shows the Security privacy control. The header, in the top-right corner, displays the control's name, while the content of the box displays the privacy service operation, SSL, and its description.

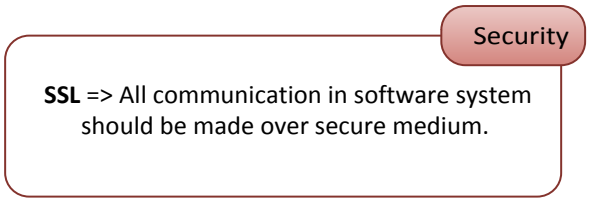


Figure 3.4 Security (Privacy Service) Example

Similarly, the Notice privacy service control is shown in Figure 3.5, which describes what steps a software developer has to take to make the software compatible with Notice service.

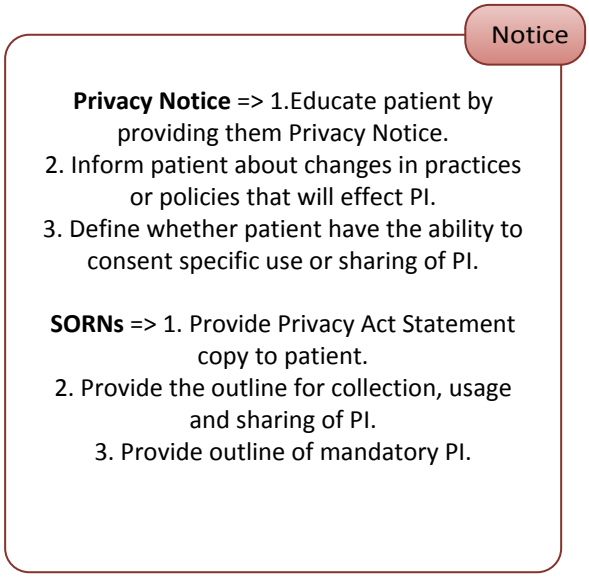


Figure 3.5 Security (Privacy Service) Example

3.6 Privacy Services Container Component in UML

Most use-cases are too complex to be represented in one page so software engineers utilize sub use-cases to hide the system complexity and to scale the diagram. There is a similar scaling problem when representing privacy requirements. For instance, communication lines between a few actors and sub use-cases will lead to many instances of the same control with many communication lines/connections between the various actors and components, rendering the diagram unmanageable and difficult to understand. For instance, if a few actors communicate with a few sub use-cases and such communication requires SSL, the SSL privacy control will appear on each communication line, which will clutter the diagram. Clearly, some representation is required to reduce the clutter and support scaling of the diagram as the number of privacy service operations that are required increases. We propose the use of a supercontainer as is shown in in Figure 3.6. The supercontainer will host all the privacy services controls required for a use-case diagram and reduce the diagram's complexity by avoiding the creation of

multiple instances of a privacy service. We now describe how the communication lines are represented to connect the privacy service operations with the actors and other controls, such as sub use-cases.

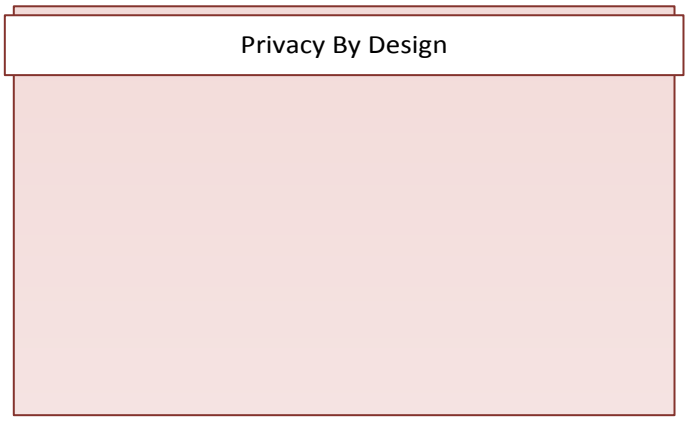


Figure 3.6 Privacy Service SuperContainer Control

Figure 3.7 shows a simple use-case wherein an actor (receptionist) is viewing the patient's appointment history without privacy requirements. For instance, it does not show that the retrieval of private data from the database must be over a secure channel. So at the time of code writing, the programmer will only implement what is shown in Figure 3.7, while completely ignoring the security aspects. By using privacy controls, however, the software engineer will represent the privacy requirement as shown in Figure 3.8.

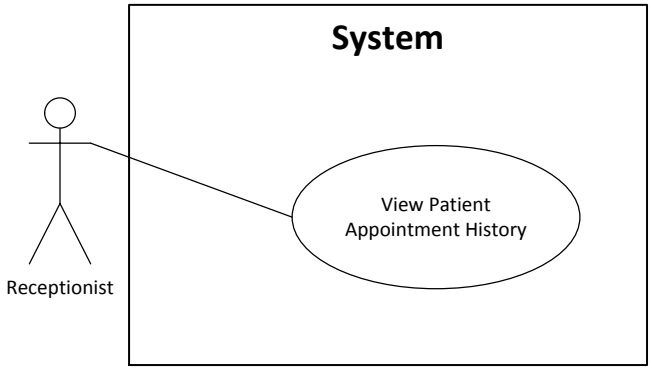


Figure 3.7 View Patient's Appointment History Use-Case

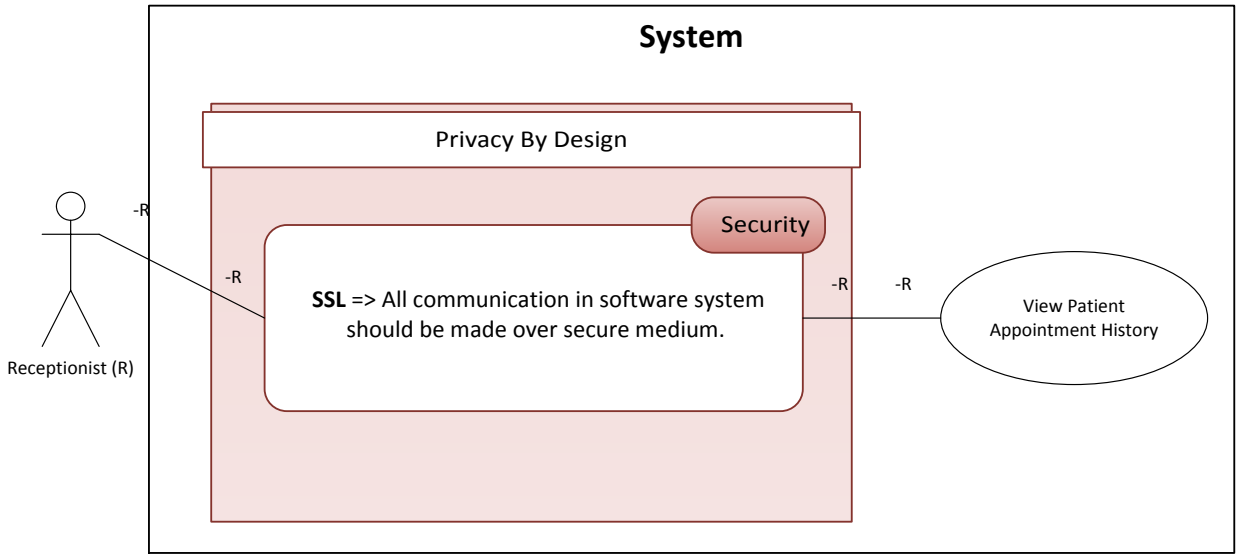


Figure 3.8 View Patient's Appointment History Use-Case - using prototype

In Figure 3.8, the communication line from the actor, Receptionist, to the privacy security service specifies the use of the SSL operation. The letter R on the communication lines specify that it belongs to the actor, Receptionist.

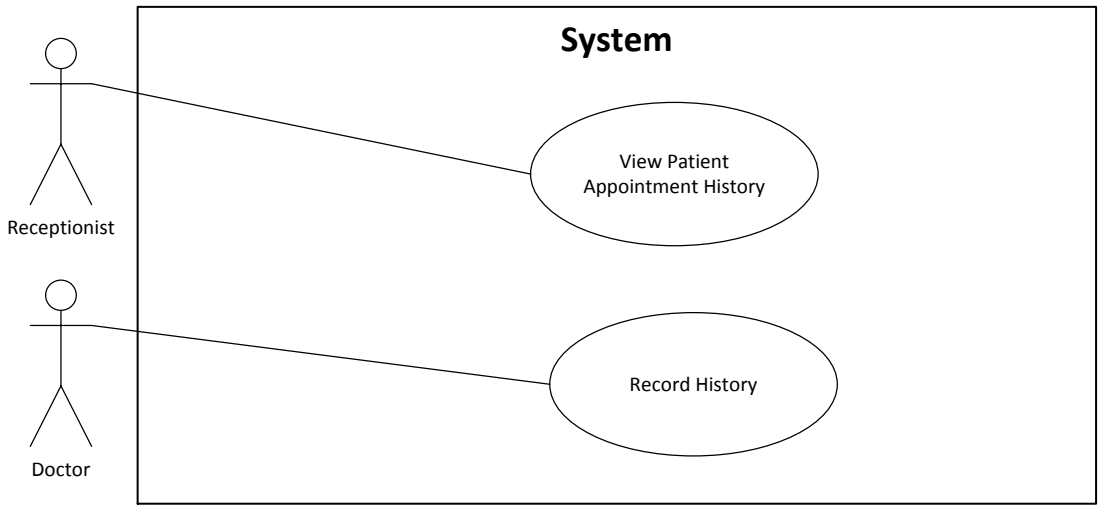


Figure 3.9 Clinical Activity Use-Case I

In the Clinical Activity I use-case, shown in Figure 3.9, the Receptionist actor is using View Patient's Appointment History, while the doctor records the patient's medical history through the Record History sub use-case. Considering privacy, the use-case (Figure 3.9) violates the Security, Notice, and Consent privacy services requirements. Using our proposed privacy controls, the same use-case is re-drawn (Figure 3.10) in order to eliminate the privacy violations. In Figure 3.10, the communication lines from the receptionist and the doctor actors are labeled/marked with letters R and D, respectively, in order to identify to which actor the communication line belongs. The communication line from the receptionist, labeled with R, is directly connected to the Security services control (and not to the supercontainer) and then further extended, via a communication line, labeled with R, between the Security services component and the View Patient Appointment History sub use-case. This means that the receptionist actor is using Security service for getting the patient's appointment history. The communication line between the doctor and the Record History sub use-case of Figure 3.9, is represented by a line, labeled with D, from the doctor to the supercontainer, as opposed to the individual control(s) within the container, which means that all privacy services in the

supercontainer apply. The line is then extended, labeled with D, from the supercontainer to the Record History sub use-case. Thus, for the doctor, to record patient's medical history, all the supercontainer's privacy services, Security, Notice, and Consent, must apply.

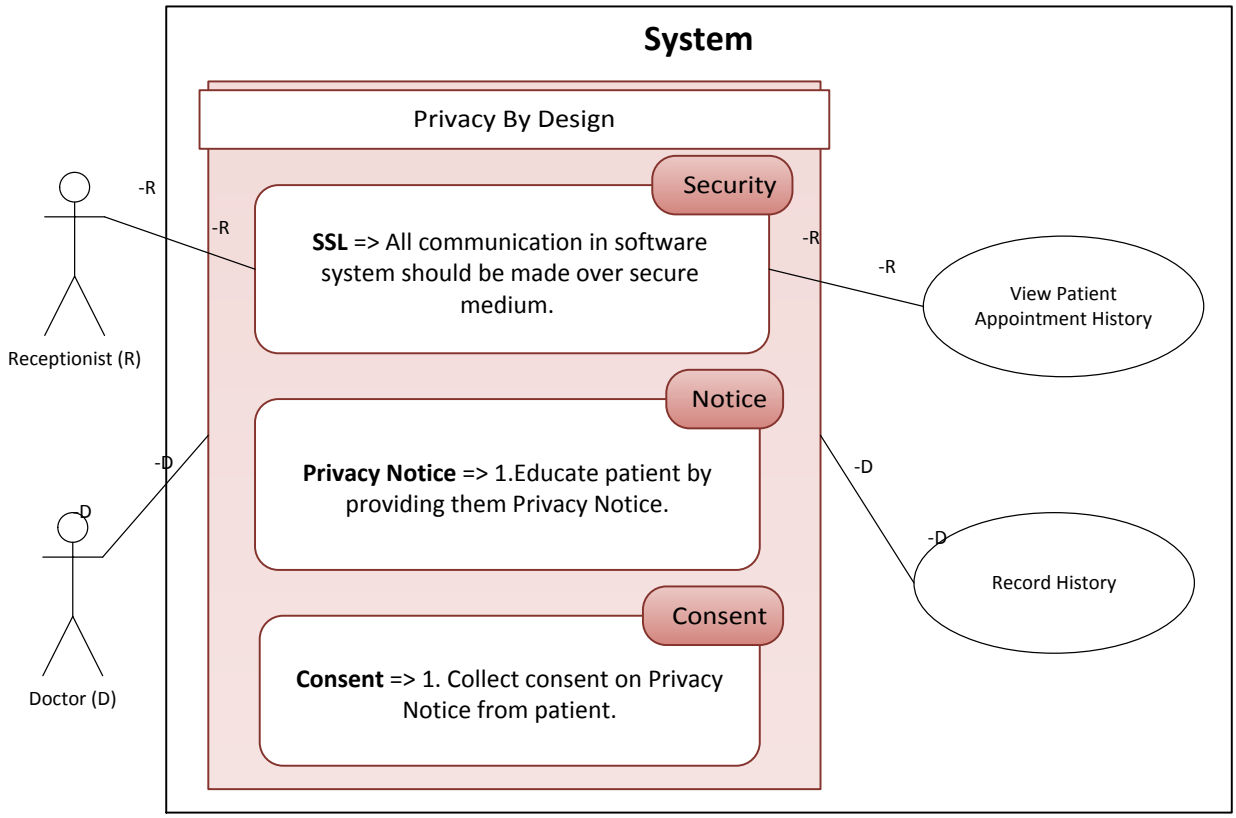


Figure 3.10 Clinical Activity Use-Case I with Privacy Container

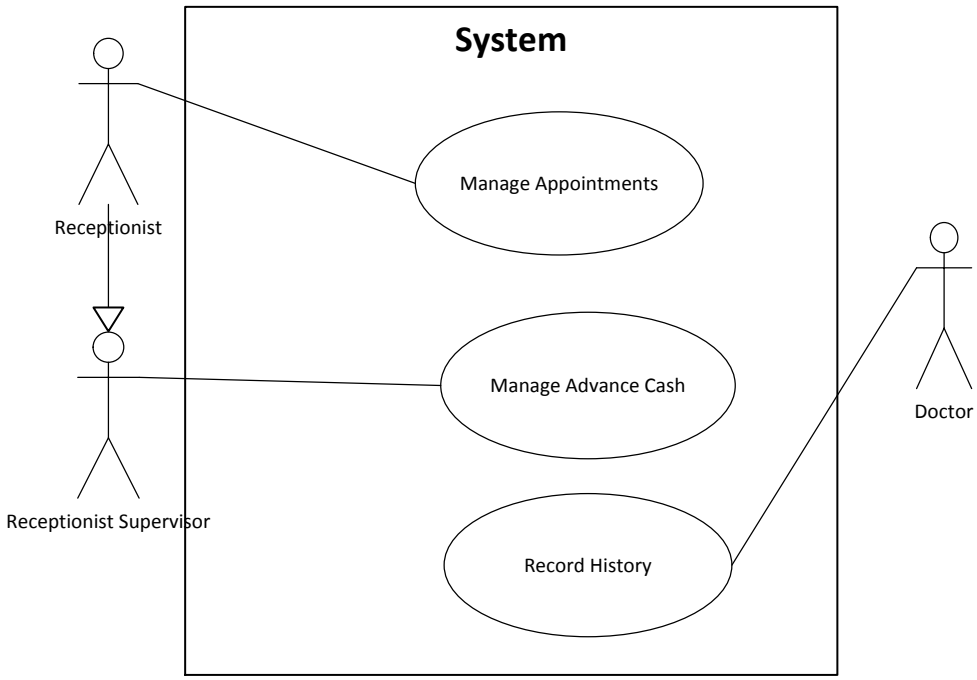


Figure 3.11 Clinical Activity Use-Case II

Now consider the Clinical Activity Use-Case II in Figure 3.11, in which three actors are dealing with three sub use-cases. We have identified the privacy violations and re-drawn the same use-case using our privacy controls as shown in Figure 3.12. Specifically, the receptionist needs to use SSL when communicating with the system, and must also present a patient with Notice and Consent when obtaining private information from the patient. The Receptionist Supervisor is receiving a payment and thus needs to use only SSL. The doctor is recording new patient medical records: the patient needs to be provided with a Notice (which must be retrieved from the system), a consent must be obtained from the patient and recorded, and SSL must be used in communication with the system.

To represent these privacy requirements, for a receptionist supervisor, there is a line, labeled RS, form the RS actor to the Security control containing the SSL privacy service

operation, and then another line, labeled RS, from the Security control to the Manage Advance Cash use-case.

As the doctor requires all privacy controls (Security, Notice, and Content), the communication line, labeled with D, is connected from the D actor to the supercontainer and not to the individual controls. Similarly, there is a communication line, labeled with D, from the Privacy by Design container to the Record History sub case. There are also communication lines, labeled with D: from the Notice to the Retrieve Notice sub-case, in order to retrieve the notice; from the Consent to the Store and Retrieve and there are also communication lines from the Consent to the Retrieve and Store Consent sub-case. (We shall address the labeling of the lines with D and R shortly.)

As the receptionist requires all privacy controls (Security, Notice, and Content), the communication line, labeled with R, is connected from the R actor to the supercontainer and not to the individual controls. Similarly, there is a communication line, labeled with R, from the Privacy by Design container to the Manage appointments sub case. There are also communication lines, labeled with R: from the Notice to the Retrieve Notice sub-case, in order to retrieve the notice; and from the Consent to the Retrieve and Store use-case.

As for both the doctor and receptionist, there needs to be a line from Notice to Retrieve Notice sub-case, instead of drawing two lines, one labeled with D while the other with R, there is only one line with two labels, R and D. For the same reason, there is only one line from Consent to the Retrieve and Store sub-case, but labeled with D and R.

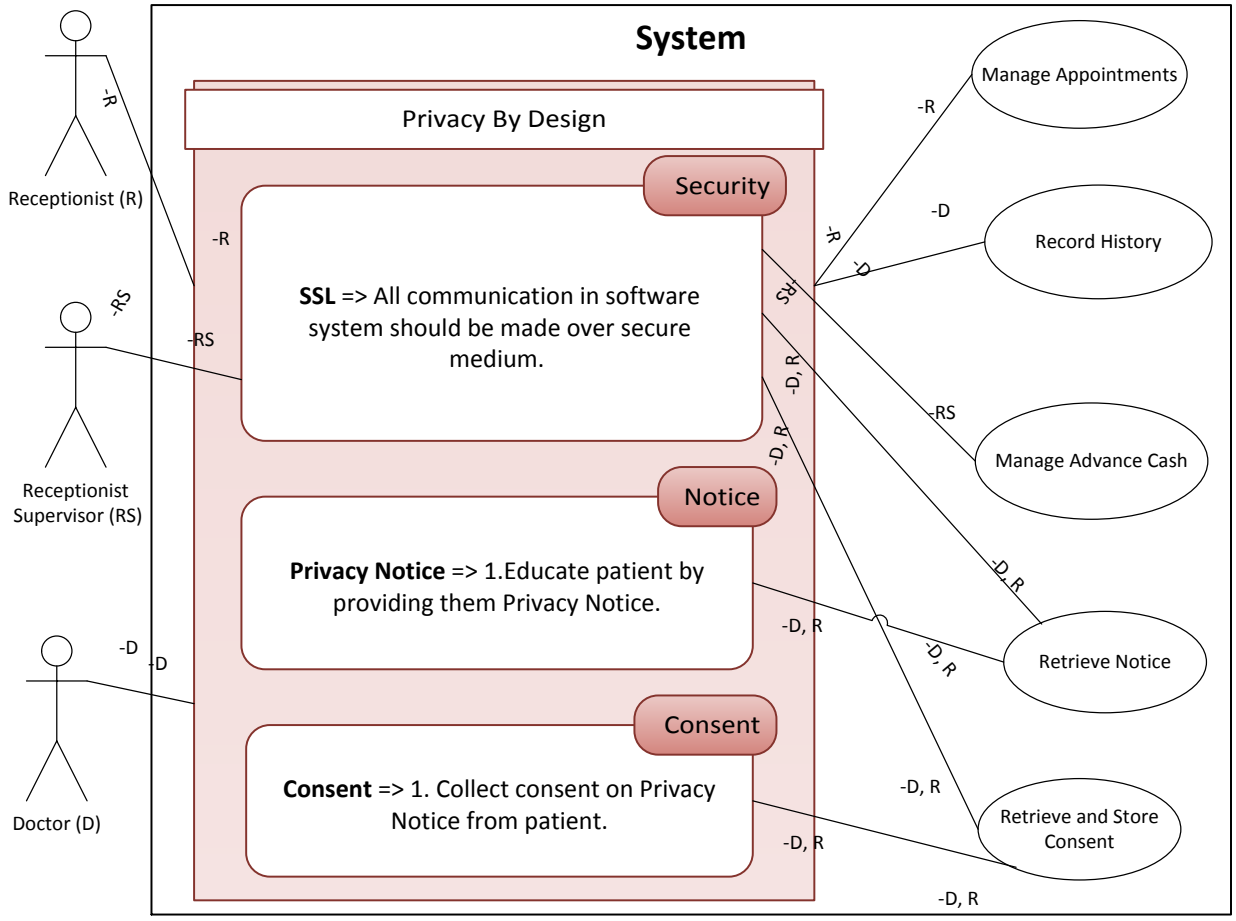


Figure 3.12 Clinical Activity Use-Case II illustrating more complex interaction with the supercontainer of Privacy Controls

Further examples will be used in subsequent sections and chapters.

3.7 Example of UML Privacy Controls Significance

The Unified Modeling Language (UML) is the industry de-facto standard for software and systems development. To develop a system on a large scale is difficult and can be made up of tens or hundreds of software and hardware components. It is difficult for the software development teams to keep the track of all the components and their functionalities. It is also difficult to share the design with the team members. Furthermore, there are many details associated with the system components that could easy be misinterpreted or overlooked while

developing system. Modeling is used to address these issues (Miles & Hamilton, 2006). During system designing, the purpose is to manage complexity and make it easy to understand the requirements and the design across the team(s). UML helps to reduce the efforts to author long documentation describing the system. UML also helps to abstract away details that are confusing and thus makes it easier for the software developers to understand and examine the system.

The purpose of this work is to extend the UML with new components to address the privacy modeling problem. Figure 3.13 shows an online patient registration system use-case developed using the existing UML controls. As UML does not have any privacy controls, the model does not address privacy issues. When the use-case, shown in Figure 3.13, is used in the implementation phase, services ensuring privacy will be absent and, unless the implementers are cognizant of the privacy requirements, implementation will result in violating the user's privacy and leaving room for unauthorized actors to get access to the user's information as shown in Figure 3.13.

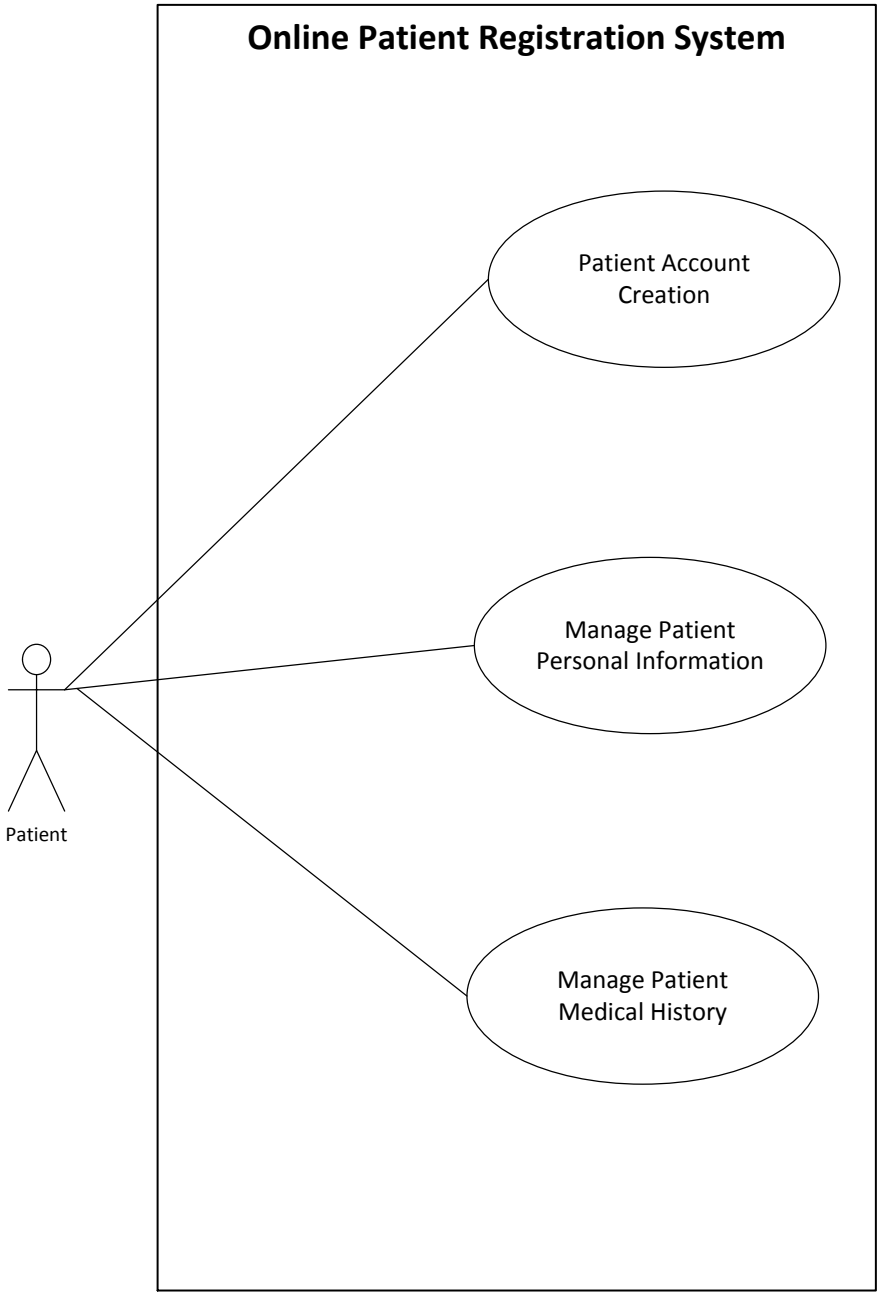


Figure 3.13 Use-Case - Online Patient Registration System

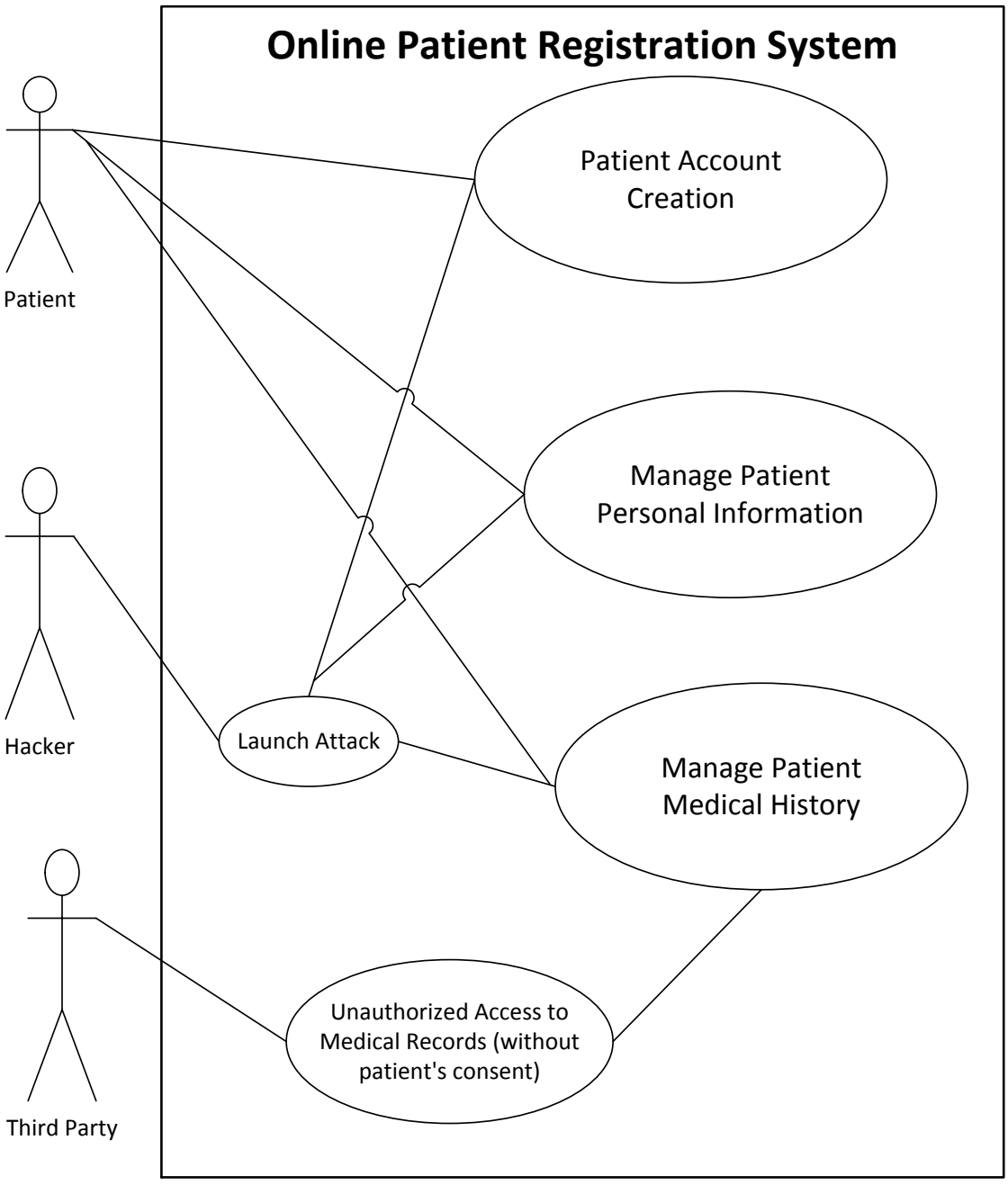


Figure 3.14 Use-Case - Online Patient Registration System

At the implementation phase, the developer will only focus on the parts which are drawn in the use-case (shown in Figure 3.13), as it is used to document the requirements. Figure 3.14 shows privacy problems by introduction of two additional actors, a hacker and a third party. It

shows that the patient's information may be shared with third parties without getting the formal approval or consent from the patient. Also, as the patient data is stored in plain-text, if a hacker is successful in getting into the system, he/she can gain access to the patient data.

If the proposed UML privacy extension is used, however, privacy issues can be addressed in the UML use-case right from the start. Using the proposed controls, appropriate privacy services are specified in the early, requirements specification, phase. Figure 3.15 shows the result of the software engineer specifying, using the Security, Notice, and Consent controls, the corresponding privacy services and thus eliminating the privacy breaches made, apparent when the use-case was developed without the proposed privacy controls.

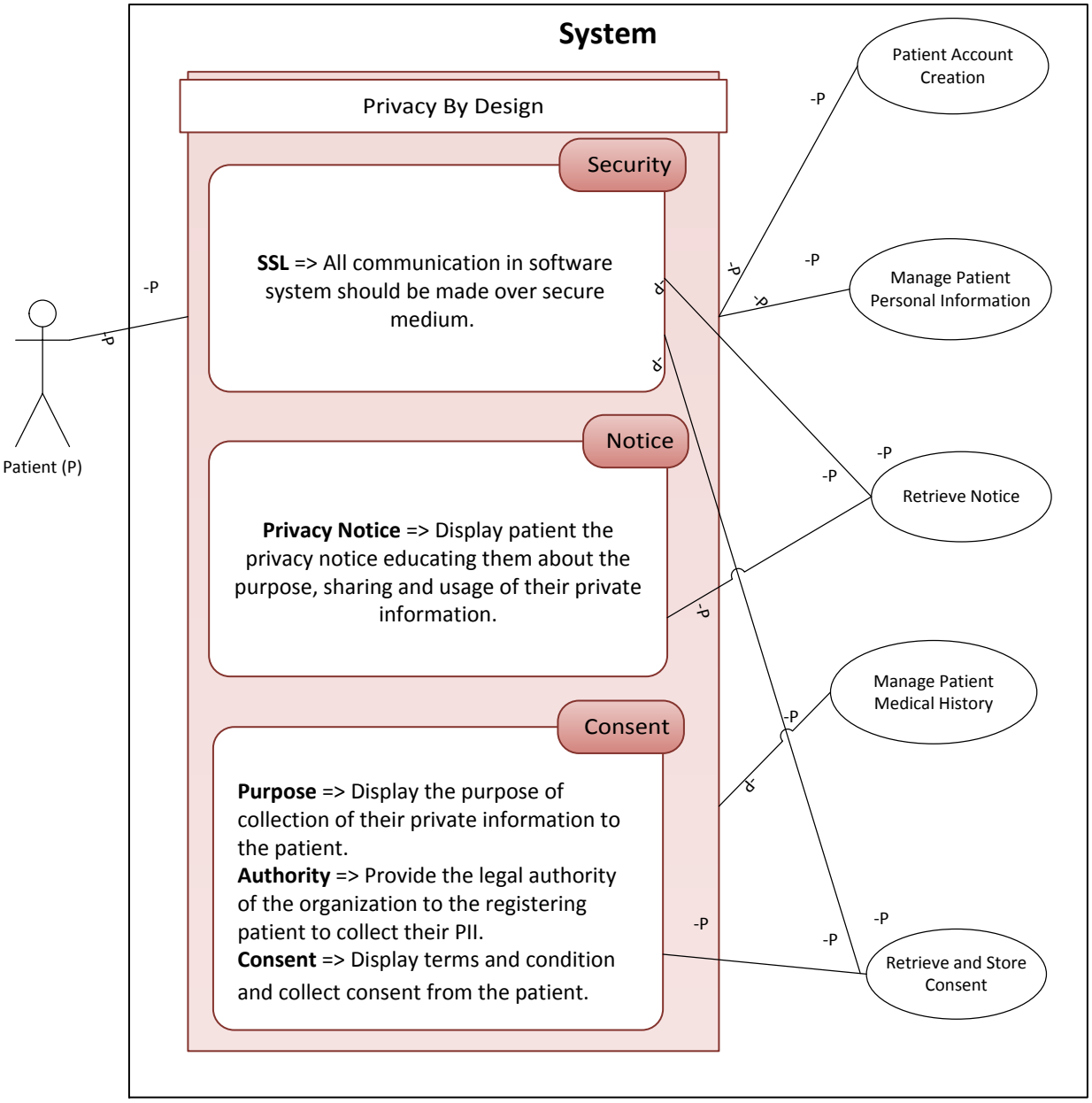


Figure 3.15 Use-Case - Online Patient Registration System

The use-case of online patient registration systems shows, in Figure 3.15, that all the communication should be done over secure channel that is using HTTPS protocol. Before registering the patient with the hospital system, a patient should be provided with a privacy notice and should be informed about the hospital privacy guidelines. Patient should also be informed about the purpose of collecting information about her/him. Notice should also contain

information about the usage, retention, sharing, and maintenance of the patient's private information. The purpose of the controls shown in this use-case diagram is to make sure that while developing the registration module, the software developer should cover all these aspects. In this way, organizations can make privacy an integral part of their software systems.

CHAPTER 4 : PROTOTYPE FOR PROOF OF CONCEPT

4.1 UML Modeling Tools

Unified Modeling Language is used to model the application structure, behavior, architecture, business process and data structures (Object Management Group – UML, n.d.). It is most widely used standard for the software systems development (Visual Paradigm for UML 10.1 Community Edition, n.d.). In this section we describe four leading UML modeling tools currently available. We examined each one with the objective to determine whether the tool's architecture is sufficiently open for us to integrate our proposed privacy controls. Our examination revealed that the first three tools are not amenable for integration of our privacy controls; however, we also determined that the last one, Microsoft Visio 2010 is and hence we selected it for developing of a proof-of-concept prototype for our work. The following sections describe how the proposed privacy controls are embedded into Microsoft Visio 2010.

4.1.1 Papyrus Eclipse Plug-in

It is an open source tool which is a part of Eclipse project. The main aim of Papyrus is to provide an integrated environment for system modeling and specially to provide support for the UML based languages such as SysML and MARTE. It provides editors, for the UML's different types of diagrams), for Eclipse Modeling Framework (EMF) based modeling language – among them are UML 2 and SysML (Papyrus, n.d.). Figure 4.1 shows the use-case diagram in Papyrus plug-in installed in Eclipse IDE.

Papyrus also provides advanced support for UML profiles that helps users to define the editors for Domain Specific Languages (DSLs) based on UML 2 standard (Papyrus, n.d.). The most powerful feature is the IDE customization mechanism to create a user defined Papyrus perspective which provides the same look and feel as a pure DSL editor (Papyrus, n.d.). During prototype development, the idea is to extend the Papyrus plug-in but documentation and lack of technical support from the Papyrus team led us to a decision not to use it for our prototype.

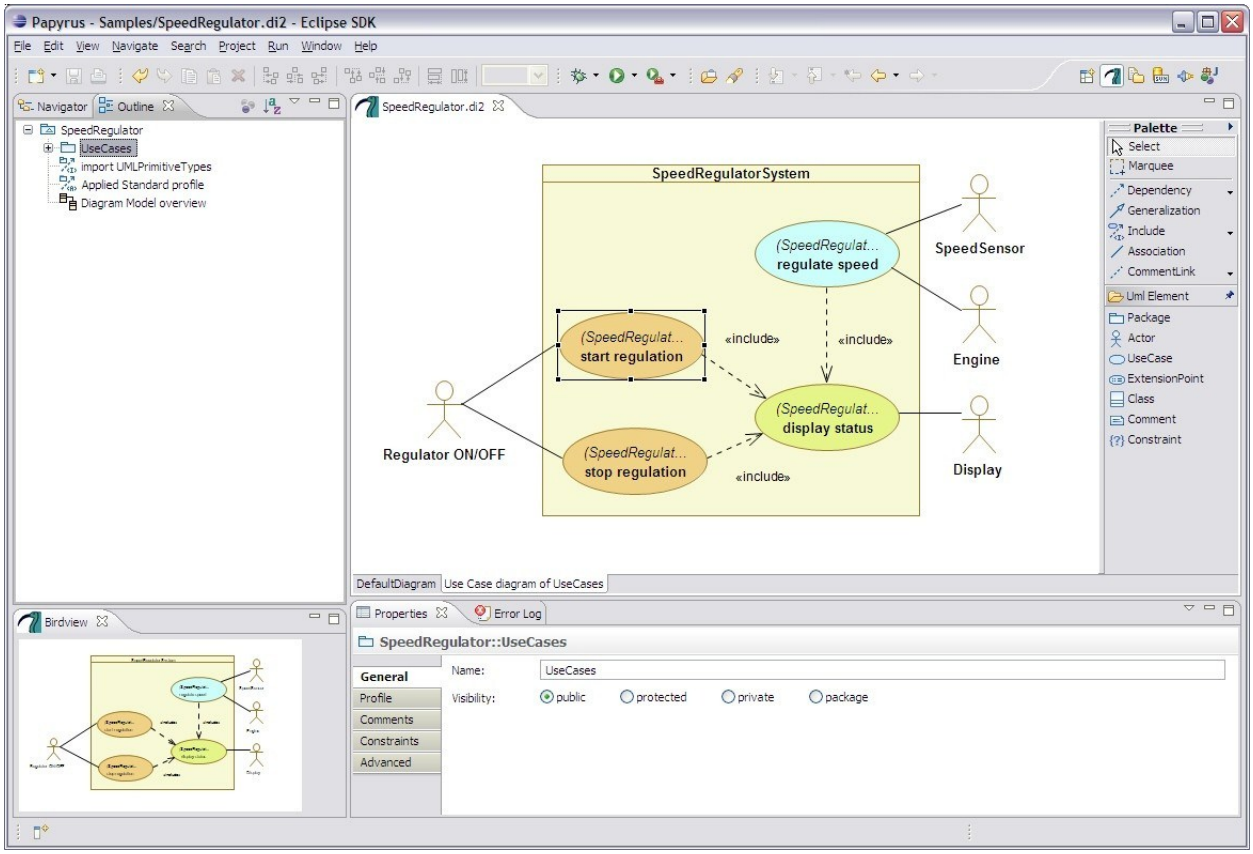


Figure 4.1 Use-Case Diagram in the Papyrus Plugin and Eclipse

4.1.2 Net Beans UML

NetBeans is an open source integrated development environment for software development, which is used to create desktop, enterprise web, and mobile applications (NetBeans IDE - Overview, n.d.). The UML plugin for NetBeans IDE is available for the version 6.7 and the earlier releases. NetBeans UML plugin supports activity, class, sequence, state, and use-case diagrams (UML, n.d.). Figure 4.2 shows the class diagram in NetBeans 6.5.

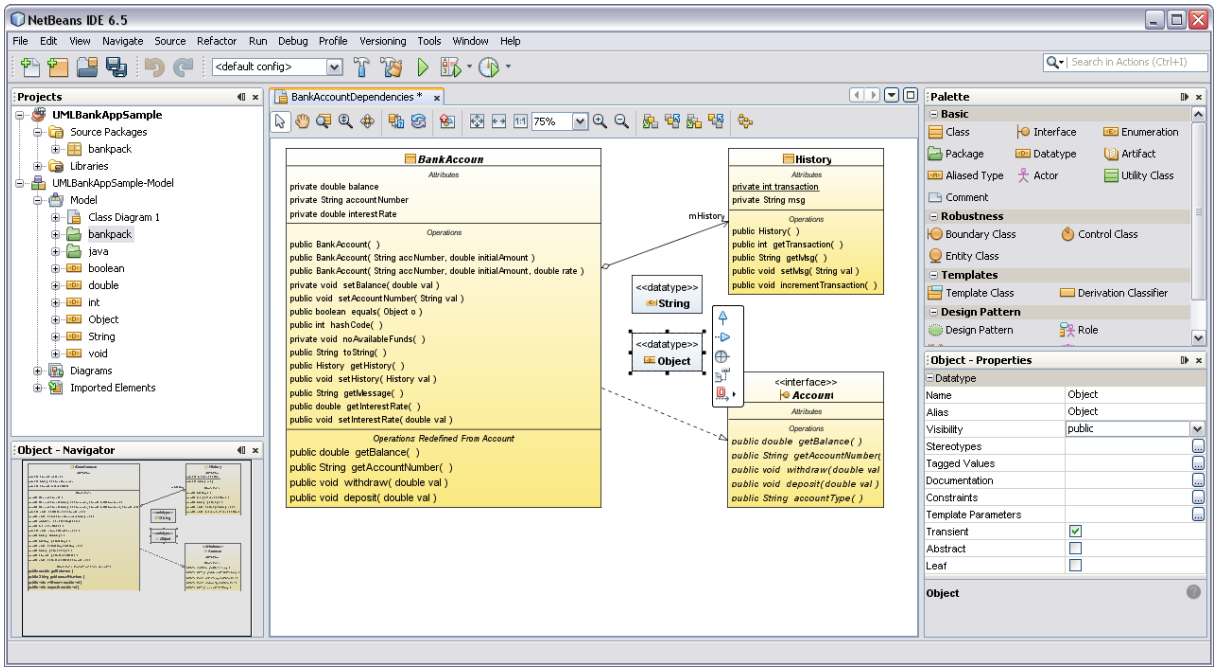


Figure 4.2 Class Diagram in NetBeans UML Editor

NetBeans UML editor provides support to programmers to use diagramming environment easily. It also maintains use-case diagrams, class diagrams, sequence diagrams and other UML diagram while effectively handling visual display of the diagram's elements (UML, business process, requirement and database design plug-in for NetBeans, n.d.). The source code for the NetBeans UML plug-in is available on the NetBeans project web page. However, an outdated

project documentation and poor support, discourage software developers to make further contributions to the project. These are also the reasons that we did not choose it for our work.

4.1.3 IBM Rational Software Architect

IBM Rational helps to transform the software development phase to the delivery phase. It provides the guidance in the entire project life cycle from design to deployment (IBM developerWorks - Are you new to Rational?, n.d.). It provides assistance to the software developer to automate and integrate main aspects of the software delivery across different roles from the requirement analyst to the developer, tester and project manager. This approach addresses each phase of the software development that helps teams to manage values and, control risk and change (IBM developerWorks - Are you new to Rational?, n.d.).

IBM Rational Software Architecture is built on the Eclipse open source software framework. It supports the model-to-code and code-to-model transformation. In model-to-code transformation, it transforms UML into Java, C#, C++, EJB, WSDL, XSD, COBRA and SQL. In reverse transformation it transforms Java, C++ and .Net into UML. Figure 4.2 shows an example of the UML modeling in IBM Rational Software Architecture.

This tool is good for the development point of view in that it transforms UML into code or code into UML. There is one drawback, however, in that the developers are not allowed to customize this software – only IBM has the code base and manages the releases. Thus, the tool cannot be used to integrate our privacy constructs into UML models.

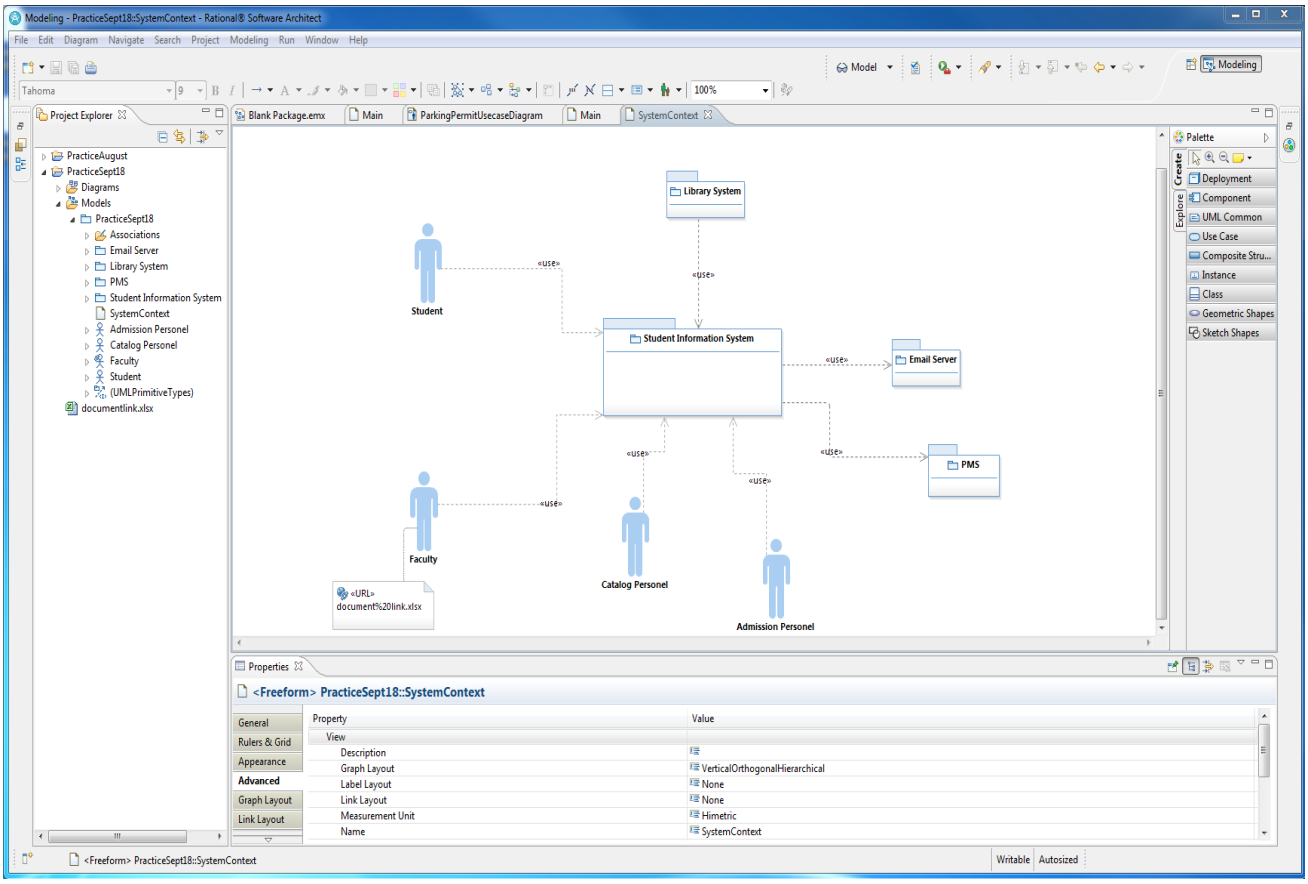


Figure 4.3 IBM Rational Software Architect

4.1.4 Microsoft Visual Studio 2010 UML Modeling

Visual Studio IDE was developed by Microsoft Corp. and is a powerful tool that can simplify the entire application development process from the beginning to the end. Visual Studio provides templates for the five frequently used UML diagrams (activity, class, sequence, component, and use-case). It also provides the facility to create a layer diagram, which can help to define the structure of the system (How to: Create UML Modeling Projects and Diagrams, n.d.).

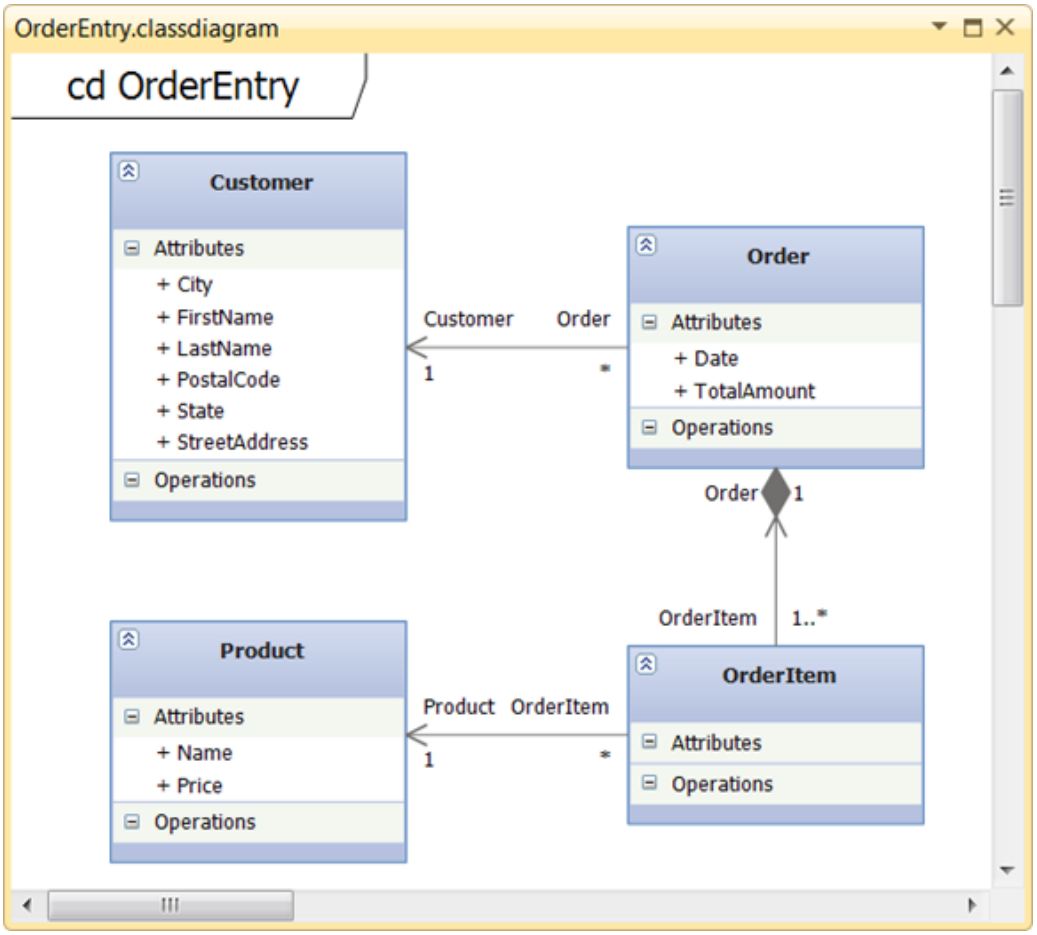


Figure 4.4 Class Diagram for Order Entity in Visual Studio UML Editor

All the UML diagrams and, layer diagrams are created in a modeling project. Figure 4.4 shows the partial view of the UML class diagram model. Visual Studio has an UML Model Explorer to view and edit the models. (How to: Create UML Modeling Projects and Diagrams, n.d.).

The visualization and Modeling Pack for Visual Studio 2010 provides the facility for programmers to generate code from the models and vice versa, explore the existing code, and use, and manage model elements (Microsoft Visual Studio 2010 Visualization and Modeling Feature Pack, n.d.).

The drawback of this modeling tool is that Microsoft did not provide the source code for the modeling module of Visual Studio 2010 and thus programmers are not allowed to make the changes into a modeling module itself. As the Visual Studio is not an open source IDE so programmers cannot extend the tool.

4.1.5 Xfig

Xfig is an interactive drawing tool written by university students. It provides a set of drawing objects, such as circles, boxes, lines, and spline curves and also provides basic UML controls. It is available for most of the UNIX-compatible platforms with Cygnus X server utility, but it is not available for Windows platforms. The last version was released in 2010 and has poor documentation (xFig User Manual, n.d.).

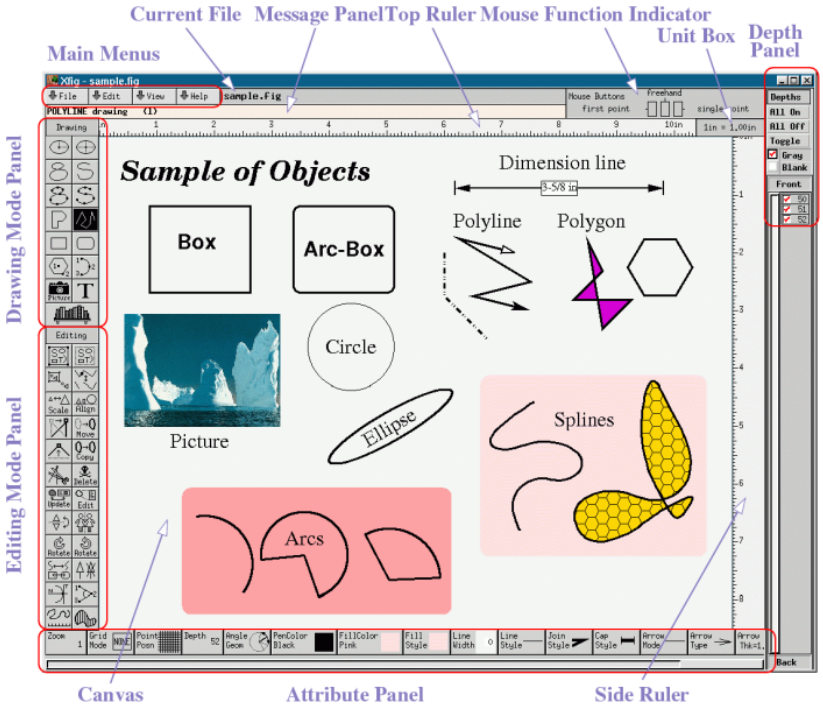


Figure 4.5 Xfig Drawing Canvas (Adopted from (xFig User Manual, n.d.))

4.2 Microsoft Visio 2010 and Microsoft Visual Studio 2010

To develop a working prototype, we built our plug-in for Microsoft Visio 2010 using Visual Studio 2010. The reason for choosing Visio 2010 is that it is widely used in industry for modeling and engineering drawing purposes; it has up-to-date documentation for each product and all published Application Programming Interface (APIs); it has a well-organized and managed API with plenty of examples; and it has an active and open forums for discussion.

Microsoft has published APIs online called "Visual Studio Tools for Office - VSTO" for developing add-ins for Microsoft Office suite using Microsoft Visual Studio. It enables the programmer to write a plug-in for the Microsoft Office 2003 suit and the later versions. We created a plug-in for our privacy controls using VSTO 3.0. There is a stand-alone installer that installs the required library files on the development machine in order to get started with the plug-in development. It also comes with many examples that are very useful during an actual development.

Visual Studio provides many project templates that help programmers to create add-ins for Microsoft Office Suite (Office development with Visual Studio (VSTO), n.d.). Using VSTO, programmers can build two types of solutions:

1. Solutions that focus on web integration
2. Solution which target .Net Framework and is integrated with MS Office's Object Model. (Office Development in Visual Studio, n.d.).

4.2.1 Tools Used

We used the following tools and applications to develop a working prototype:

- Microsoft Visual Studio 2010 (Professional Edition)
- Microsoft Visio 2010 (Professional Edition)
- Microsoft Office 2010 Primary Interop Assemblies (PIA)
- .NET Framework 3.5
- Microsoft Visual C#
- VSTO 3.0
- Microsoft Enterprise Library 5.0
- Microsoft Windows 7

4.3 Microsoft Visio 2010 Add-Ins Development

By writing add-ins a developer can automate Microsoft Office applications, extend default features, and customize the user interface of the application (Getting Started Programming Application-Level Add-Ins, n.d.). Visual Studio provides two types of project templates for MS Office development i) Document level customization, and ii) Application level add-ins (Office Solutions Development Overview, n.d.). These are described below in the following subsections.

4.3.1 Document level customization

Document level customization is associated with a single document, workbook, or template available in any of the applications in the Microsoft Office Suite. The customized

feature is available only to the associated document (Office Solutions Development Overview, n.d.). This type of customization does not allow to make application-wide changes, such as adding a new menu or adding an item to ribbon.

4.3.2 Application level add-ins

Application level add-ins are directly associated with the Microsoft Office application, which is Microsoft Visio in our case. The add-ins is loaded when the associated application is started. The add-in features are available to all the documents when opened by the application. Through application level add-ins we can add items to menus and ribbons etc. (Office Solutions Development Overview, n.d.). For our prototype purpose we have written an application level add-in.

4.3.3 Developing Add-ins Project

When we select an add-in project template in Visual Studio, it automatically creates a *ThisAddIn.cs* file, which acts as the foundation for writing add-ins. The following code snippet shows the basic structure of the file. The Startup and Shutdown event handlers enable the developer to hook up any necessary functionality at the start and shutdown of the Microsoft Visio application (Office Solutions Development Overview, n.d.).

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml.Linq;
using Visio = Microsoft.Office.Interop.Visio;
using Office = Microsoft.Office.Core;

namespace SampleVisioAddInDevelopment
{
    public partial class ThisAddIn : Microsoft.Office.Tools.AddInBase
```

```
    {
        private void ThisAddIn_Startup(object sender, System.EventArgs e)
        {
        }

        private void ThisAddIn_Shutdown(object sender, System.EventArgs e)
        {
        }

        #region VSTO generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InternalStartup()
        {
            this.Startup += new System.EventHandler(ThisAddIn_Startup);
            this.Shutdown += new System.EventHandler(ThisAddIn_Shutdown);
        }

        #endregion
    }
}
```

We are interested in developing a ribbon in Microsoft Visio, which will help the software developers to select privacy controls from a ribbon to model the privacy components in use-case diagrams. The following section describes the development of ribbon.

4.3.4 Extending Visio Ribbon

Microsoft has introduced a ribbon interface element in Office applications since its 2007 version. The ribbon is a way to organize related commands so that it becomes easy for the user to find them. Commands appear on the ribbon as controls and are organized into groups. Figure 4.6 shows the sample screenshot of the ribbon in Microsoft Visio. Visual Studio allows us to create a custom ribbon that can be plugged in during Microsoft Visio startup.

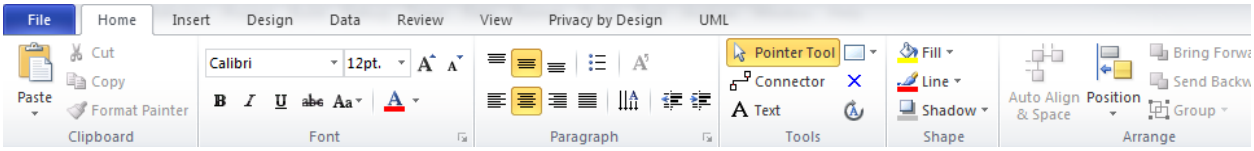


Figure 4.6 Microsoft Visio ribbon example

We have created a Privacy by Design ribbon in Visual Studio as shown in Figure 4.7. We called it a Privacy by Design ribbon as it supports incorporation of privacy into software in early phases of development as per principles stated in (*Privacy by Design*). The ribbon is automatically loaded at Visual Studio application startup. Visual Studio generates a code file that corresponds to the ribbon shown in Figure 4.7. Through that code file, we can access the Microsoft Visio Object Models that will expose many internal Visio classes, interfaces, and types (Visio Object Model Overview, n.d.). We have used these classes, interfaces, and types to automate the privacy modeling task.

Figure 4.7 Privacy UML ribbon

Visual Studio generates the following code when we choose a ribbon item to be added into the project.

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using Microsoft.Office.Tools.Ribbon;

class PrivacyByDesignRibbon : Microsoft.Office.Tools.Ribbon.RibbonBase
{
       Visio.Application visioApp;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        public PrivacyByDesignRibbon()
            : base(Globals.Factory.GetRibbonFactory())
        {
            InitializeComponent();
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
        ///  otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.tab1 = Factory.CreateRibbonTab();
            this.group1 = Factory.CreateRibbonGroup();
            this.tab1.SuspendLayout();
            this.group1.SuspendLayout();
            this.SuspendLayout();
            //
            // tab1
            //
            this.tab1.Name = "tab1";
            this.tab1.ControlId.ControlIdType =
                        Microsoft.Office.Tools.Ribbon.RibbonControlIdType.Office;
            this.tab1.ControlId.OfficeId = "TabAddIns";
            this.tab1.Groups.Add(this.group1);
            //
            // group1
```

```
            //
            this.group1.Label = "group1";
            this.group1.Name = "group1";
            //
            // Ribbon1
            //
            this.Name = "Privacy by Design";
            this.RibbonType = "Microsoft.Visio.Drawing";
            this.Tabs.Add(this.tab1);
            this.Load += new
                    Microsoft.Office.Tools.Ribbon.RibbonUIEventHandler
                    (this.PrivacyByDesignRibbon_Load);
            this.tab1.ResumeLayout(false);
            this.tab1.PerformLayout();
            this.group1.ResumeLayout(false);
            this.group1.PerformLayout();
            this.ResumeLayout(false);

        }

        private void PrivacyByDesignRibbon_Load(object sender, RibbonUIEventArgs e)
        {
              visioApp = Globals.ThisAddIn.Application;
        }

        #endregion

        internal Microsoft.Office.Tools.Ribbon.RibbonTab tab1;
        internal Microsoft.Office.Tools.Ribbon.RibbonGroup group1;
    }

    partial class ThisRibbonCollection
    {
        internal PrivacyByDesignRibbon PrivacyByDesignRibbon
        {
            get { return this.GetRibbon<PrivacyByDesignRibbon>(); }
        }
    }
```

The *PrivacyByDesignRibbon_Load* event provides an entry point to access many Visio application objects with which we can interact. The objects are organized in a hierarchy style, wherein at the top level there is *Microsoft.Office.Interop.Visio.Application* object – it is a pointer to the currently running application instance. This object contains reference to the other useful objects such as:

1. *Microsoft.Office.Interop.Visio.Document*

   This is core collection of objects for programming for Visio and it represents drawing canvas, stencils, and template files. Each time we create a new Visio document, internally an object of this class is created and appended to the *Microsoft.Office.Interop.Visio.Documents* collection object – which is hosted by the *Microsoft.Office.Interop.Visio.Application* class instance (Visio Object Model Overview, n.d.).

2. *Microsoft.Office.Interop.Visio.Page*

   This object represents the drawing area of the Visio application into which the use-case diagrams are dropped.

4.3.5 Creating Privacy Policy Control Stencil

In Microsoft Visio, each shape is recognized as a 'master', and multiple related shapes are saved on a stencil. In this implementation we have created a stencil that contains the basic UML shapes that are dropped on the Visio page. We have added object lifeline type shapes for all 10 privacy services as shown in Figure 4.8. This helps the software developer or architect to create a sequence diagram that shows privacy components object communication, and lifetimes.

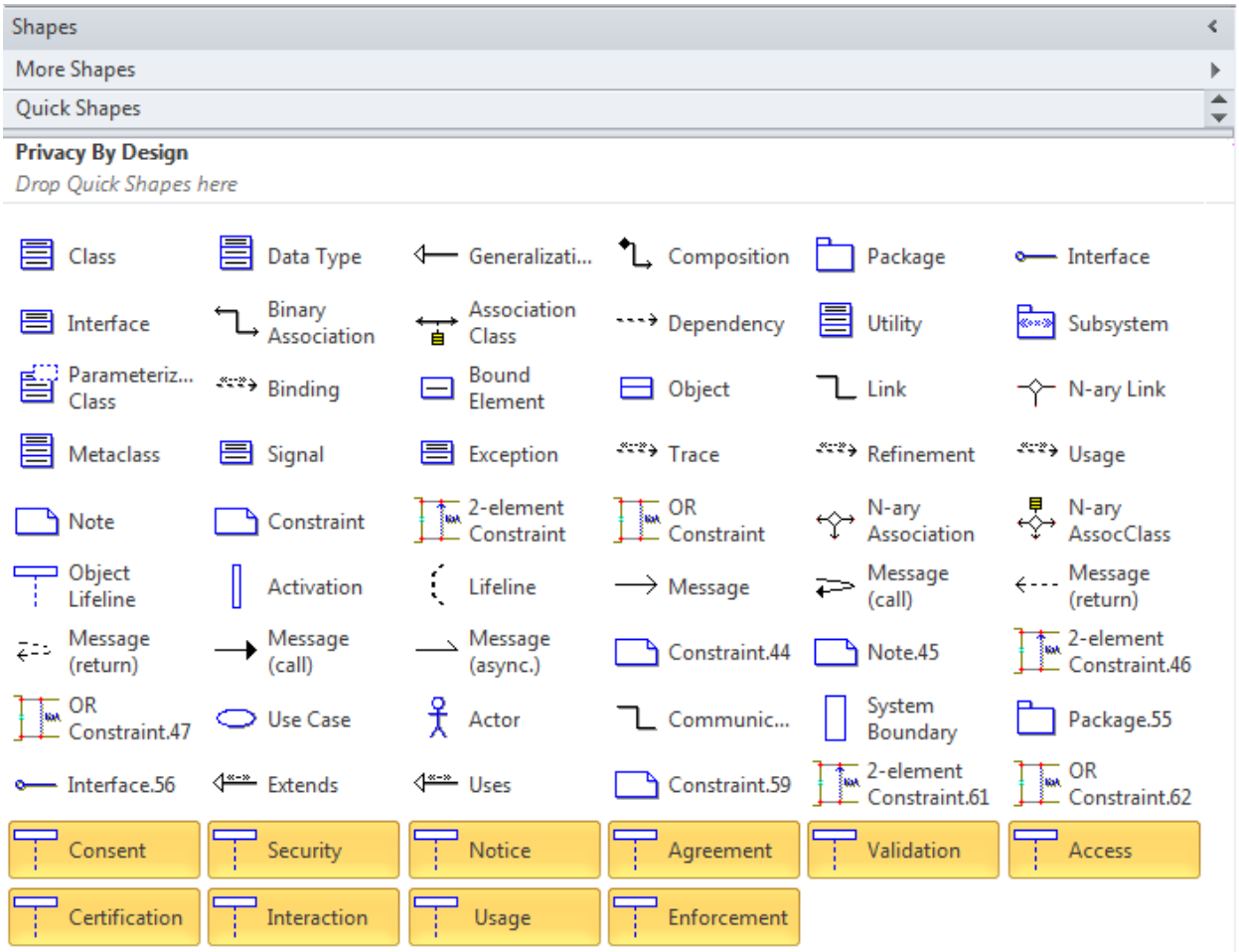


Figure 4.8 Privacy Controls in UML Stencil

4.3.6 Windows Form Programming

The prototype uses the Windows Forms. When a user clicks on any of the privacy by design controls in a 'Privacy by Design' ribbon, a form is shown to the user as depicted in Figure 4.9. This form allows the user to choose a privacy control and define the action that should be coded by the software developer. In our implementation prototype we have provided the privacy detailed privacy operations for the Notice, Consent, and Security controls. The 'Privacy by Design' form, shown in Figure 4.9, also allows for the software developers and architects to provide their own operations and their description by choosing the 'Other' operation from the

Select Operation dropdown control. The feature is useful if the architect or software developer needs to define new operations.

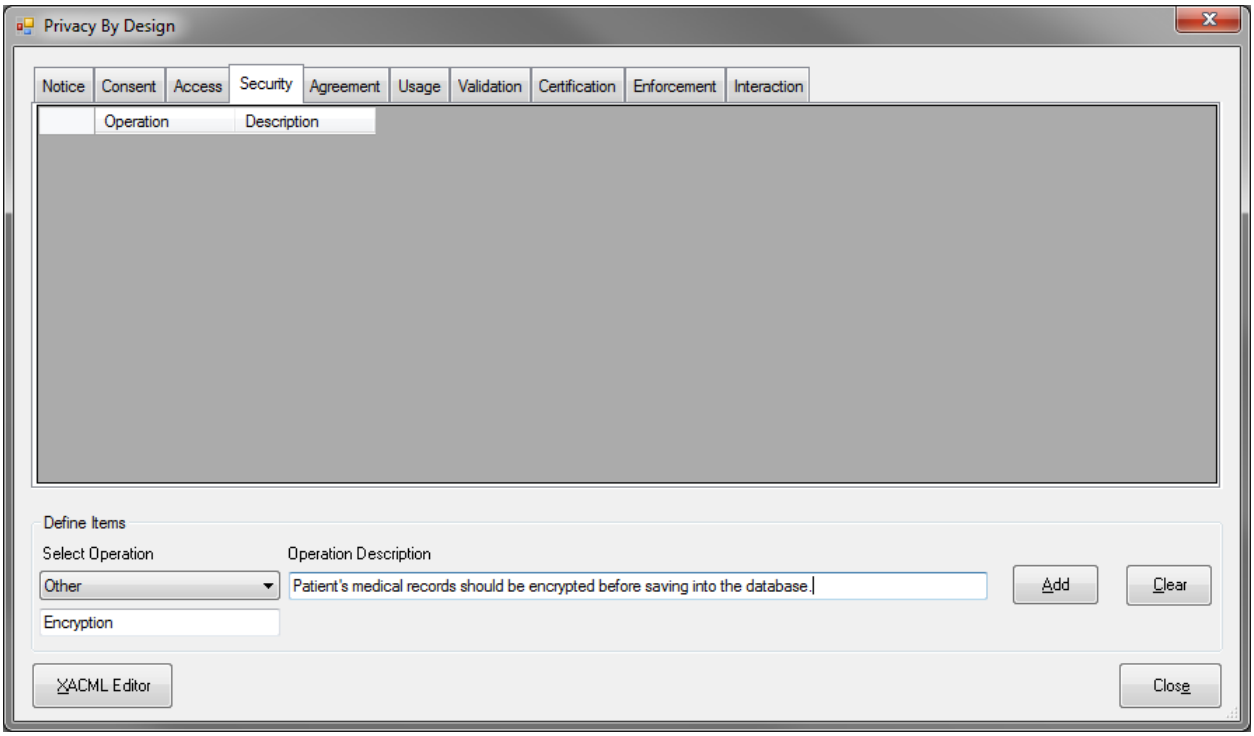


Figure 4.9 Privacy for UML User input form

4.4 Test Driven Development

The prototype was created using a test-driven development methodology. We wrote the test cases first and then wrote the actual code to pass the test cases. We have written unit test cases to cover all of the prototype's aspects. This helped us to write the code that fits our requirement and made it easy for us to refactor and manage the code. A unit test method from our actual implementation is shown below:

```
[TestMethod]
public void TestNoticeOperationCount()
```

```
{

    PrivacyServiceItems servicePrivacyItemList = new PrivacyServiceItems();

    PrivacyManagementForm frmPrivacy = new
                              PrivacyManagementForm(servicePrivacyItemList,
                              ValueObjects.PrivacyServiceComponents.Access);

    frmPrivacy.AddPrivacyItem( new PrivacyItems {
                                   Operation = "Privacy Notice",
                                   OperationDescription = "Provide user with Privacy
                                                           notice........"
                                   },
                                   PrivacyServiceComponents.Notice);

     Assert.AreEqual(servicePrivacyItemList.lstNotice.Count, 1);

 }
```

4.5 Integration of XACML Editor

In this implementation we have integrated the XACML editor – an open source implementation provided by the University of Murcia. The XML based policy language is used to define the general access control requirements to access the protected resources such as PII (A Brief Introduction to XACML, n.d.) (Yagüe, 2006). The objective of this integration is to provide the backward compatibility to the software developers and architects to make use of their existing policies defined in XACML. By clicking on the XACML Editor button, shown in Figure 4.9, will open up a XACML Editor application as shown in Figure 4.10. .Net Framework provides the *Process* class that helped us to host the XACML Editor application in our application.

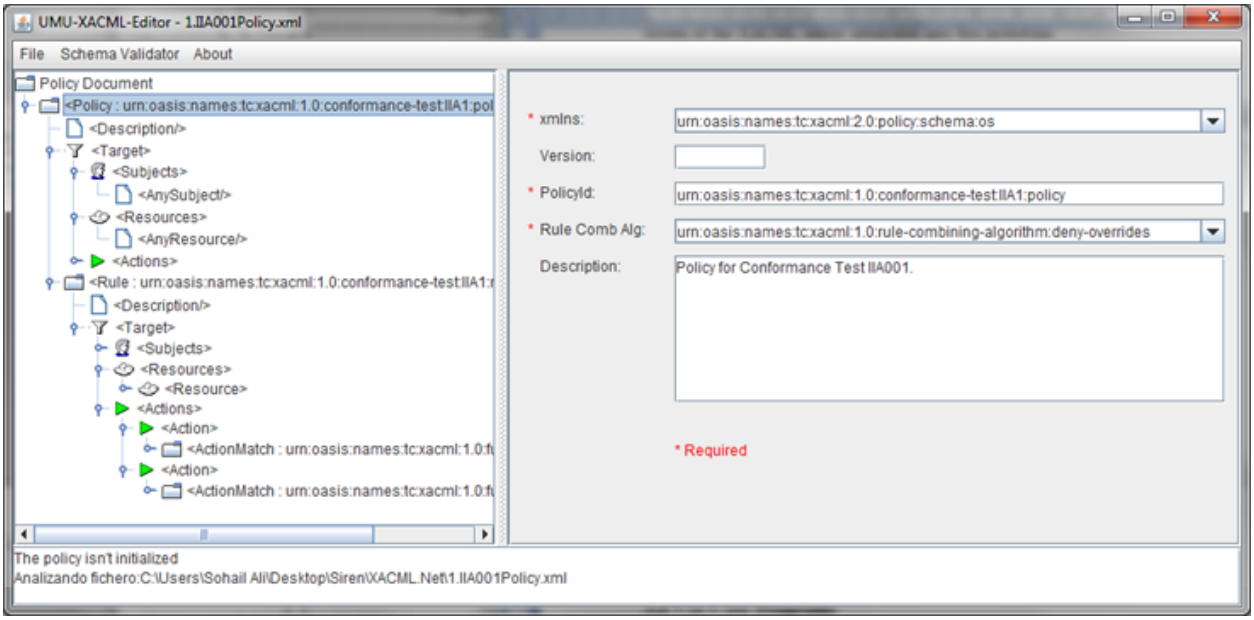


Figure 4.10 XACML Editor

CHAPTER 5 : USE-CASE SCENARIOS

In the previous chapters 3 and 4, we presented a proposal for extending UML with privacy controls and described a UML prototype for MS Visio, respectively. In this chapter we utilize our UML prototype to model privacy requirements for two real use-cases and thus demonstrate the feasibility of our approach. For each of the two scenarios, we first develop the UML use-case without utilization of our privacy controls and hence without considering the privacy requirements. This is a typical approach still practiced in the software industry in that the software requirements are developed early in the software development life-cycle, while privacy aspects are typically tackled after the fact, that is, after the software capturing the functional requirements, but not the privacy requirements, has been developed. We then examine the use-case and highlight privacy violations. Next, using our prototype, we develop the UML use-case while also using our proposed privacy controls to model the privacy requirements and thus demonstrate the feasibility of our approach. Finally, for the second use-case, we also develop a UML sequence diagram in order to show how the use-case diagram is interpreted by a software developer in creation of the sequence diagram, which represent the flow control of the actual code when it is implemented.

As the use-case diagrams are real-life scenarios, they are not simple. All the use cases discussed in this chapter are working scenarios taken, with permission, from the Agha Khan University Hospital (AKUH), in Karachi, Pakistan. In order to simplify the presentation and concentrate on the privacy aspects, we only discuss the most important aspects of the uses-cases.

5.1 Use-Case Scenario for the Unit Receptionist

Figure 5.1 shows the use-case scenario that describes the overall system behavior for a receptionist who manages the patient's appointments. This use-case has many sub-cases; they are shown in Figure 5.1 as ovals.

The Manage Appointment use-case, which is a sub use-case of Figure 5.1, allows the unit receptionist to manage appointments for any scheduled clinic held by any doctor. The unit receptionist can view appointments, book a particular appointment, edit an appointment, or delete it. The receptionist can also accept advance payments either in cash or as an online transaction. The details of the Manage Appointment use-case are provided in Appendix A.

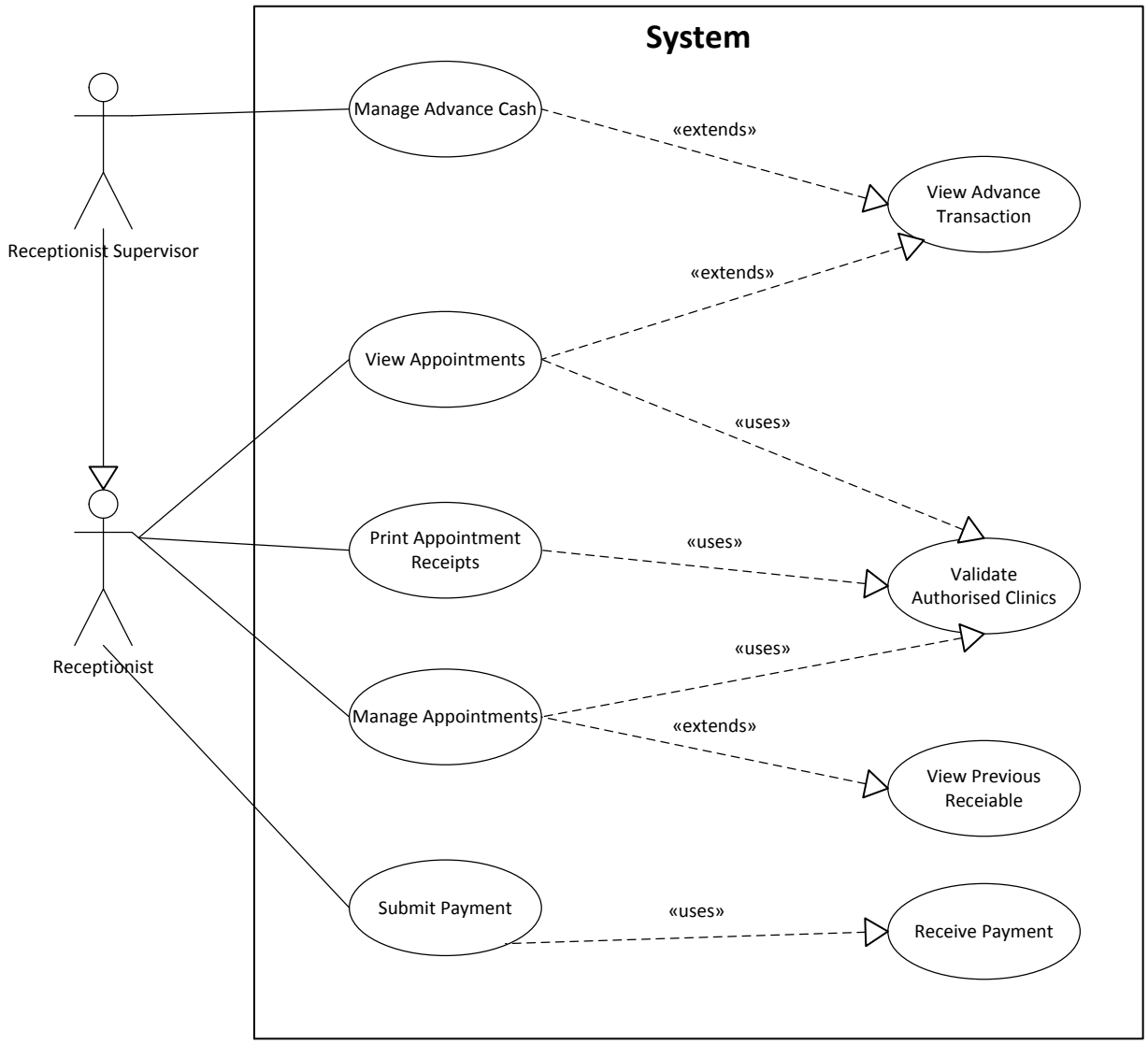


Figure 5.1 Unit Receptionists (UR) - Appointment Management System Use-Case

5.1.1 Application of the Prototype to the Manage Appointments Use-Case

Consider the UR-Appointment Management System use-case (described in Appendix section A) while keeping in mind privacy requirements. Furthermore, consider the Manage Appointment use-case – a sub use-case, represented in Figure 5.1 by an oval labeled "Manage Appointments", of the System Use-Case shown. It violates a number of the patient's privacy

principles, such as security, access, privacy notices, and consent. The use-case does not provide any details about how the system will secure the patient's private information, such as medical records and payment details. The patient is not presented with any consent form and also is not informed about the collection of her/his private data. The use-case also does not inform the patient about the organization's authority to collect private information and fair use of the collected private information. Even if the software engineer, who is using UML to determine the requirements by developing the use-case, is cognizant of the privacy requirements, specifying them is cumbersome as there are no dedicated elements in UML to support privacy services. The analyst is thus required to use stencil elements targeted for other purposes instead and then provide annotation explaining how these modeling elements are used to specify the privacy requirements.

We have applied the prototype on the Appointment Management System Use-Case shown in Figure 5.1 when developed without privacy services controls. The result of using our privacy controls to represent the required privacy services is shown in Figure 5.2. The figure immediately brings into the viewer's attention the privacy services that are required to support the patient's privacy. The diagram specifies to the software developer/programmer that there is code represented by each privacy component and thus ensures that privacy is built into the system by design and not bolted-on as is the current practice. Specifically, Figure 5.2 makes it clear that all communication between the unit receptionist and the system has to be made over an SSL connection. It should be noted that the storage of the private data would require encryption – but this requirement would be represented in appropriate sub use-cases, which describe the functionality of storing private data. The diagram also shows that the unit receptionist has to

provide to the patient a privacy notice and collect consent. The letter 'R' on the communication line between different use-cases is associated with the receptionist, while RS1 shows that the communication made by the Receptionist Supervisor. The receptionist's communication line is directly connected to the supercontainer, which means that all of the receptionist's communication lines, which are labeled with R, that are connected to the sub cases will require all privacy services contained in the supercontainer. The Receptionist Supervisor actor is directly connected to the security component, as opposed to the supercontainer. This is because the actor is only associated with the sub use-case Manage Advance Cash, which requires secure communication – it does not require the privacy services represented by the Notice and Consent controls/components. The sub use-cases View Appointments and Print Appointment Receipts, which are associated with the Receptionist, also require only the security privacy component.

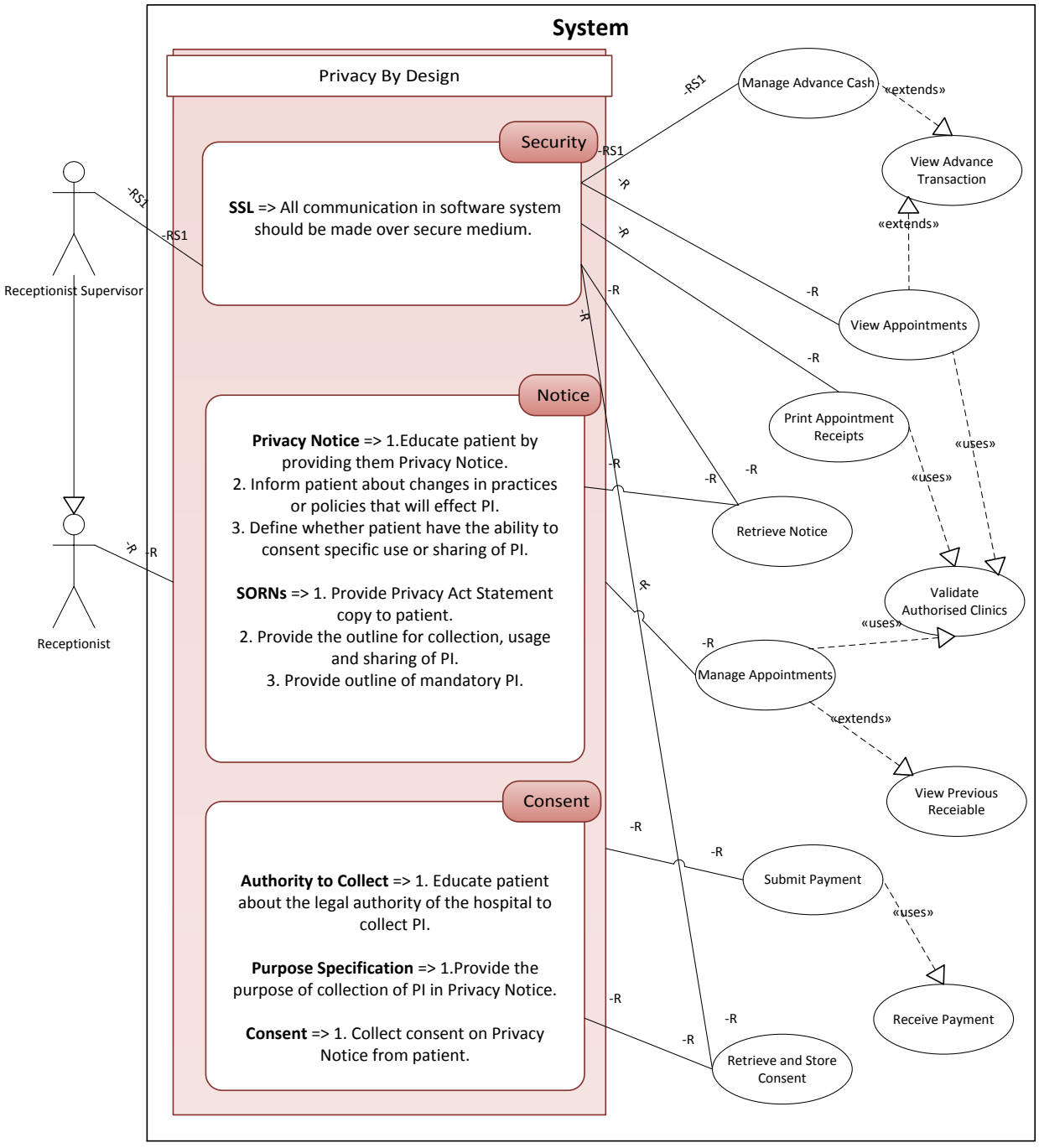


Figure 5.2 Unit Receptionists (UR) - Appointment Management System Use-Case Using Privacy Controls

5.2 Use-Case Scenario 2 - Doctor Clinical Activity Use-Case

This use-case allows the doctor to record the purpose of the patient's visit and previous background/history.

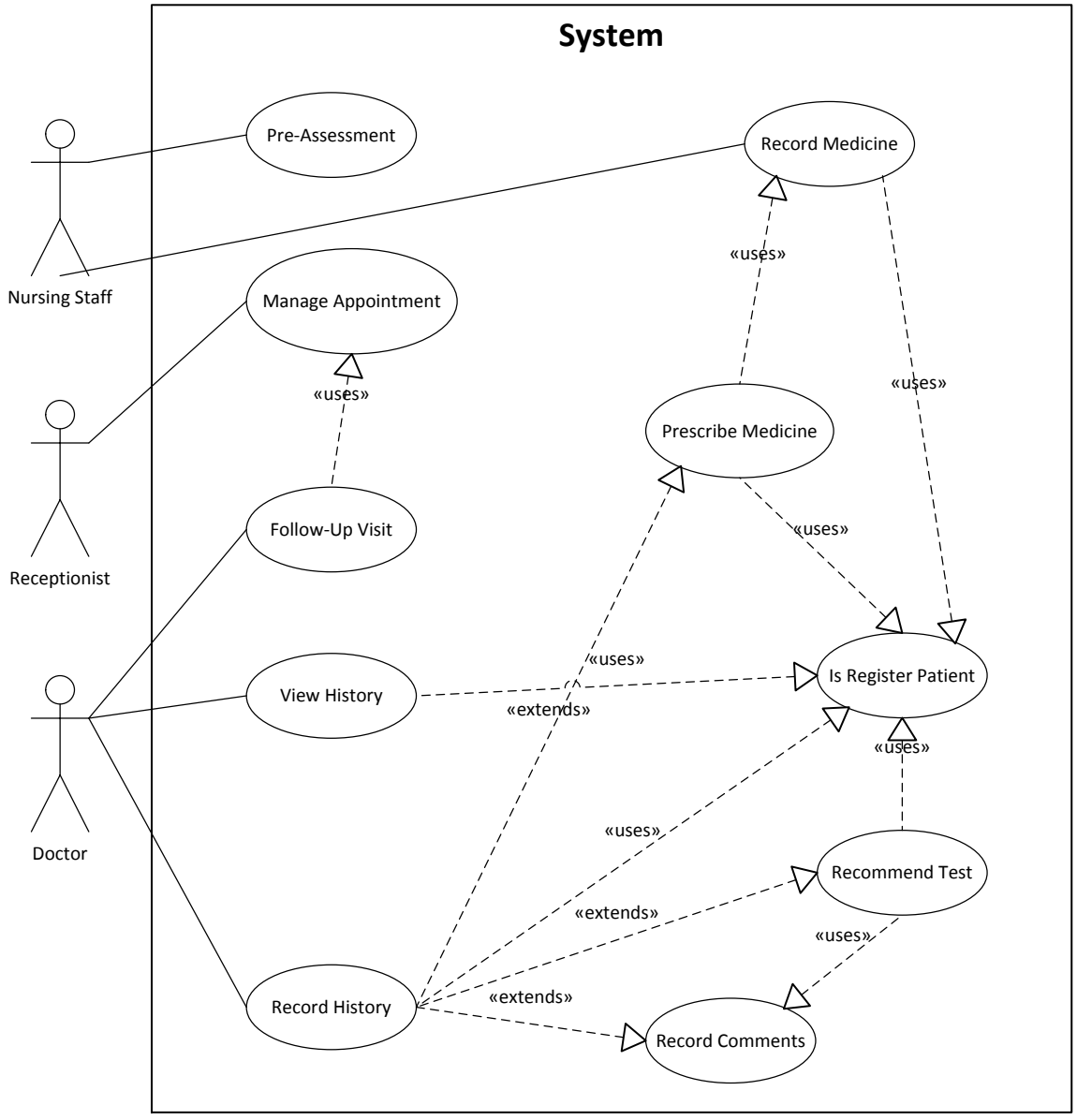


Figure 5.3 Doctor Clinical Activity Use-Case

The Doctor Clinical Activity UML use-case, shown in Figure 5.3, was created without the privacy controls, while Figure 5.4 shows the UML use-case created while using the privacy

controls to specify the privacy requirements. In our discussion, we will concentrate on the Record History and View History sub cases. Details of the Record History and the View History sub cases are available in Appendix in section B and C, respectively.

In the use-case shown in Figure 5.3, the doctor records the patient's medical history that also includes private information. The use-case does not make sure that patient's information is accurate, complete, and relevant thus and thus violating the Validation component of Privacy by Design guidelines. Another violation is that when viewing the patient's history, the system should verify the identity of the doctor as (s)he should have the appropriate right to view and update the old information. From the security aspect, the use-case does not mention anything about how the patient's information will be stored in database. Patient private information and medical records are sophisticated pieces of information that must be encrypted and stored in database. Another shortcoming in this use-case is that it does not inform or educate the patient about how organization will share and use their information with third parties.

In Figure 5.4, we have re-created the same use-case using our prototype and addressed the privacy violations discussed under Figure 5.3. In Figure 5.4, communication lines labeled with 'D' belongs to the Doctor actor, lines labeled with 'NS' are for the Nursing Staff actor, and lines labeled with 'R' are for the Receptionist actor. The receptionist use-case was already covered in section 5.1. Figure 5.4 makes it clear to the software developer to code the respective privacy components at the development time.

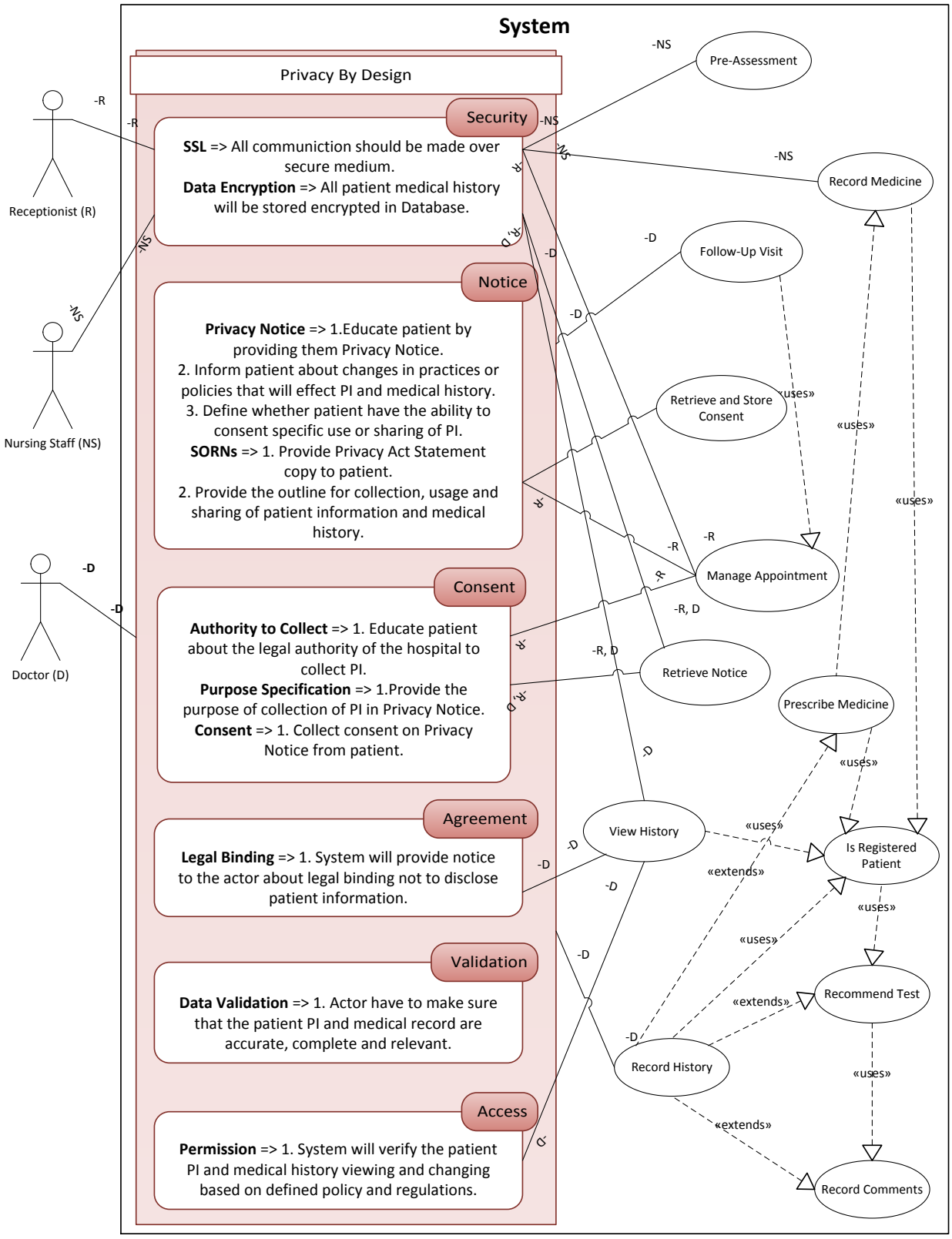


Figure 5.4 Doctor Clinical Activity Use-Case using Privacy Controls

The identified privacy violations in Figure 5.3 has been removed as can be seen in Figure 5.4. At the design time, we have made sure that before recording any medical history, the Doctor actor will be notified about the information validation and integrity. In this way, the patient's record will be kept up-to-date. Before any actor collects the data from patient, the system will provide a legal binding notice to the actor reminding them about the legal procedure and for further security, the system will collect a digital signature from actor for audit purposes. Before patient's private and personal information is displayed, the system will verify that the logged-in user has appropriate access rights to view such information.

5.3 Sequence Diagram – Doctor Clinical Activity

To further demonstrate the use of the UML use-case diagrams that include the privacy controls, we have developed two sequence diagrams for the use-case of section 5.2. The first sequence diagram, shown in Figure 5.5, has been developed from the specification represented by the UML use-case diagram shown in Figure 5.3, which is without any privacy services. The other sequence diagram, shown in Figure 5.6, was developed from Figure 5.4 that has appropriate privacy services.

Figure 5.5, shows the actors' lifelines and their performed operations. In Figure 5.5, it is clear that all actors are performing their operations without taking into account the patient's privacy. For example, the system is allowing the doctor to fetch the patient's history without verifying the role and rights of the doctor. Another example is that the doctor is not making sure that the patient's history is accurate, relevant, and up-to-date.

We have re-designed the sequence diagram by using specifications represented by the UML use-case with the privacy services as shown in Figure 5.4. The redesigned sequence diagram, shown in Figure 5.6, eliminates the privacy and security violations that were identified before the re-design. The sequence diagram shown in Figure 5.6, helps the software developer to include appropriate privacy in the implementation phase. Objects, representing controls, drawn in a red color in Figure 5.6 will help to eliminate the privacy violations. The Access, Validation, Security, Agreement, and Privacy Notice controls will make sure that related requests pass through them and in case of violation will return error to the actor object. For example, if the doctor object requests a patient's medical record, then that request will first go through the Access object instance that will verify that the doctor's instance has appropriate access permissions. If the doctor instance has appropriate access rights, then the system will allow the doctor instance to complete the 'GetHistory()' request. In case the doctor instance does have the access permissions, then the system will deny the 'GetHistory()' request. This way, embedding the privacy into the design will help the organizations to improve their software development processes to incorporate appropriate privacy services right from the beginning of the software development phase.

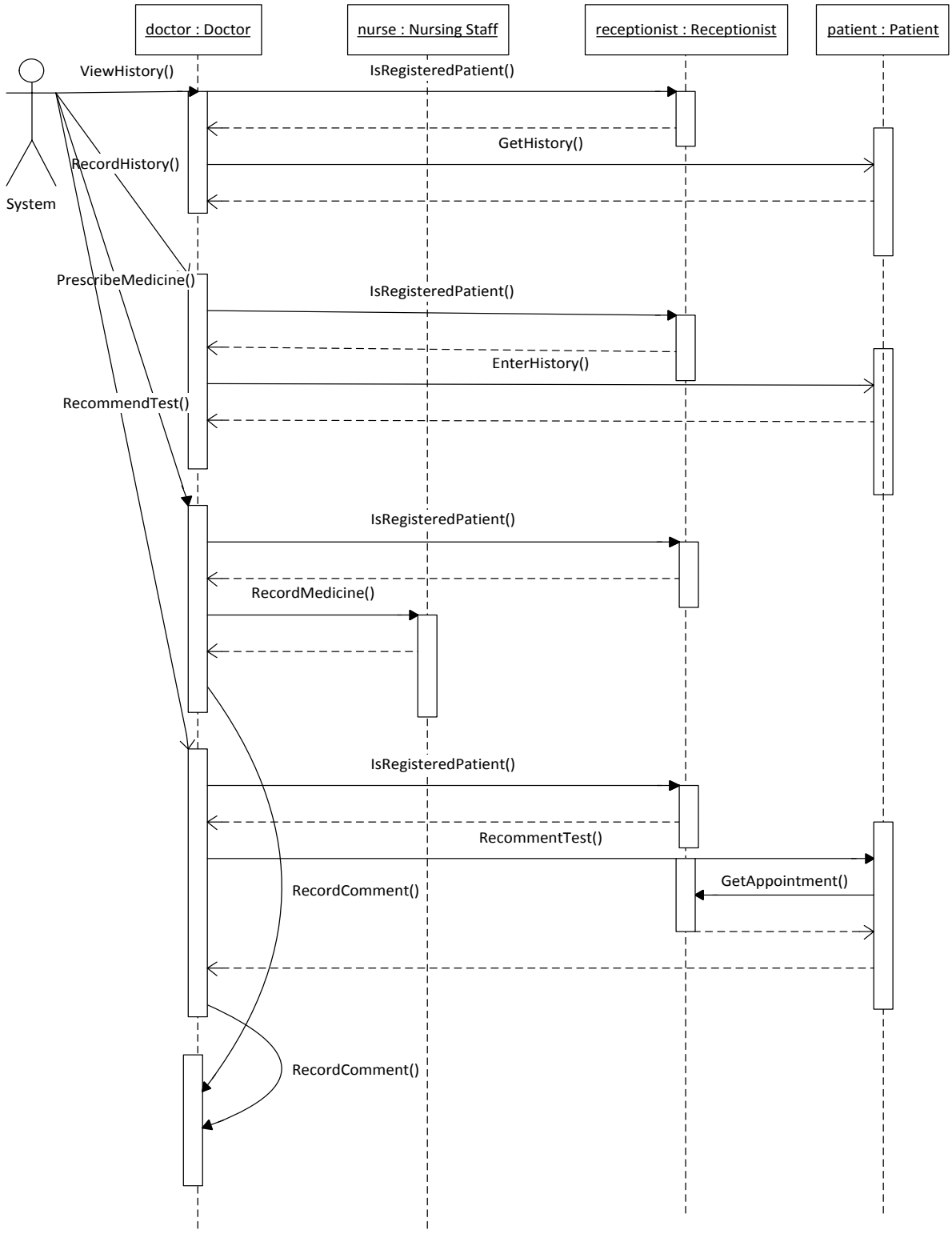
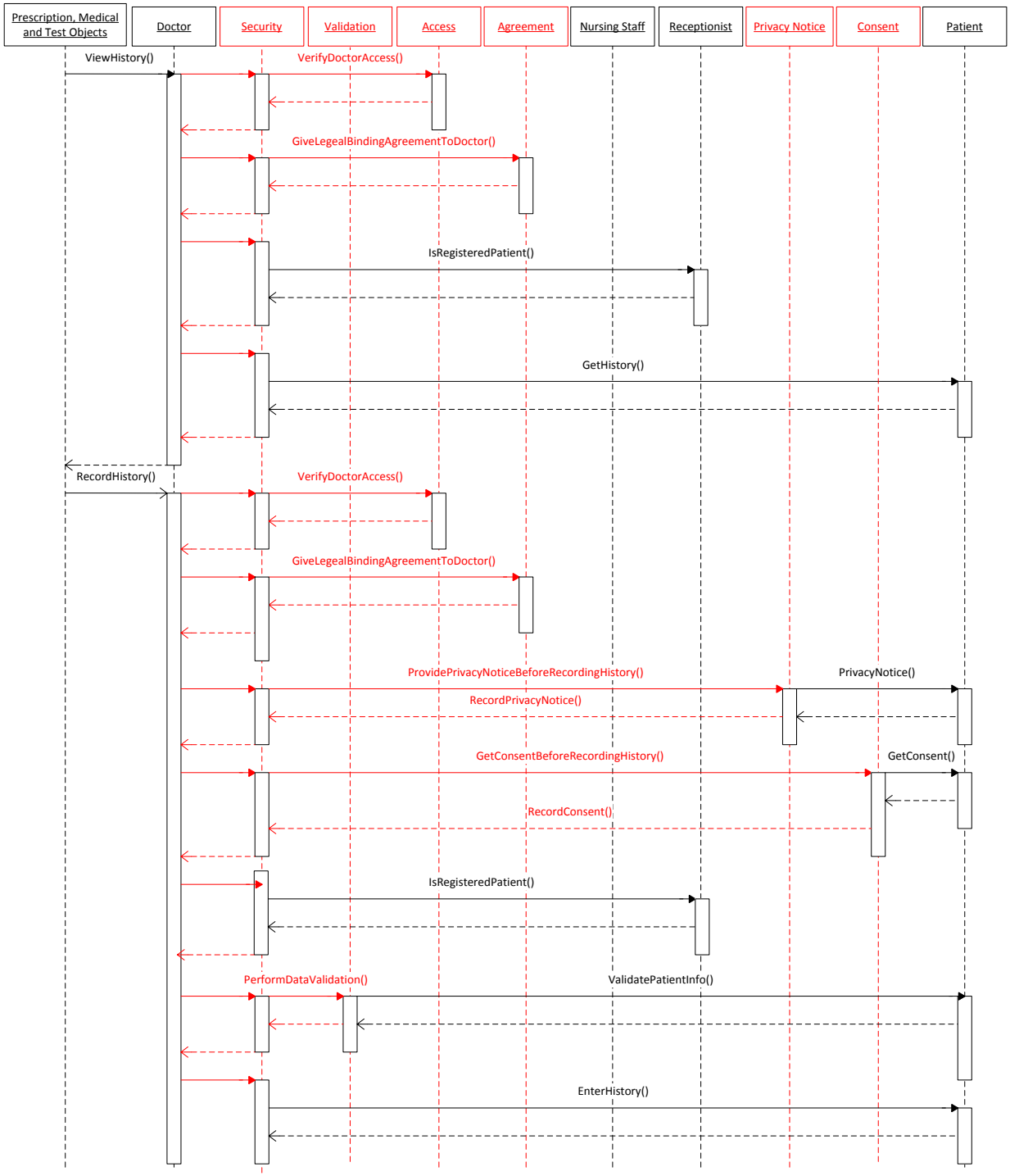


Figure 5.5 Doctor Clinical Activity Sequence Diagram


Prescription, Medical and Test Objects
Doctor
Security
Validation
Access
Agreement
Nursing Staff
Receptionist
Privacy Notice
Consent
Patient
ViewHistory()
VerifyDoctorAccess()
GiveLegealBindingAgreementToDoctor()
IsRegisteredPatient()
GetHistory()
RecordHistory()
VerifyDoctorAccess()
GiveLegealBindingAgreementToDoctor()
ProvidePrivacyNoticeBeforeRecordingHistory()
PrivacyNotice()
RecordPrivacyNotice()
GetConsentBeforeRecordingHistory()
GetConsent()
RecordConsent()
IsRegisteredPatient()
PerformDataValidation()
ValidatePatientInfo()
EnterHistory()

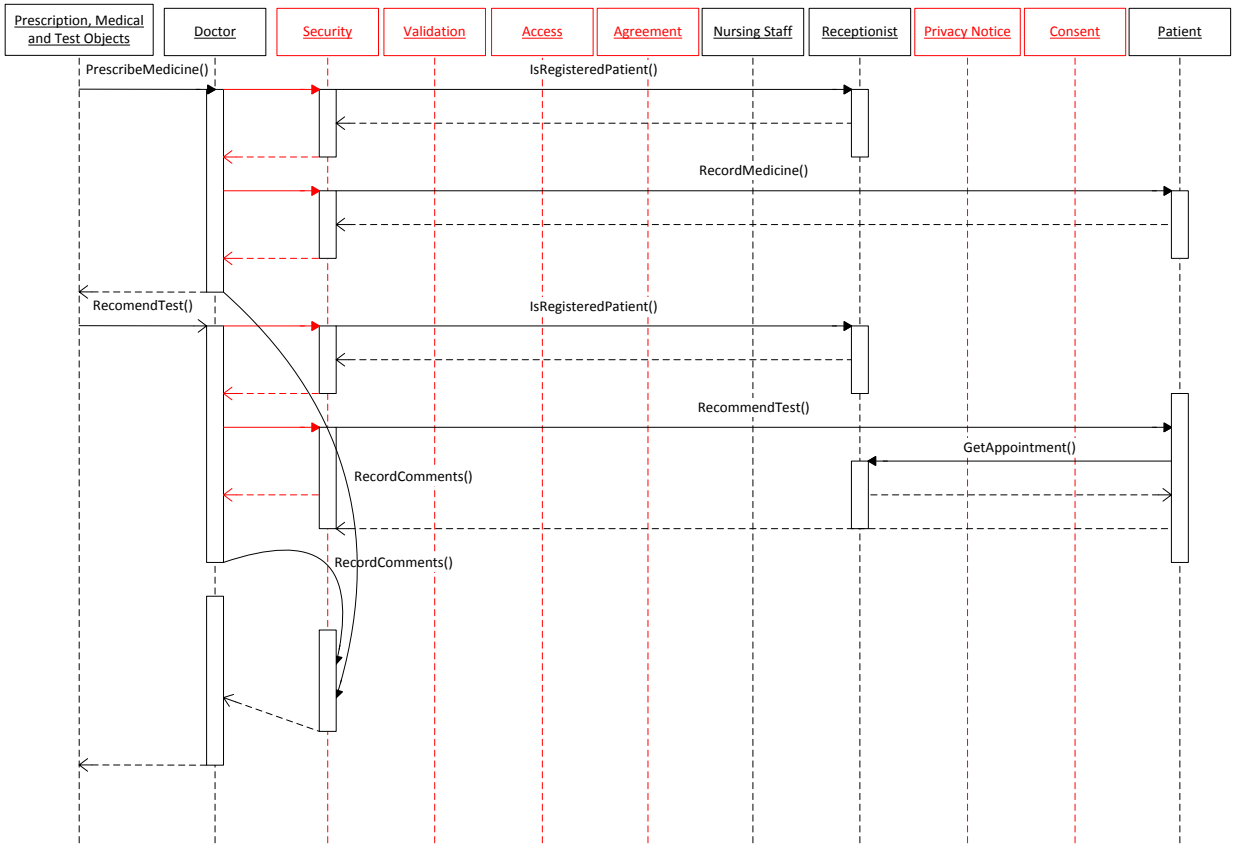


Figure 5.6 Doctor Clinical Activity Sequence Diagram using Privacy by UML Controls

CHAPTER 6 : CONCLUSIONS AND FUTURE WORK

Privacy embedded by design in software implementations can give users freedom of choice, personal control and self-determination of private information. This empowerment provides a win-win situation to business owners and to their customers by providing a solution which is closer to the Fair Information Practices and on the same token educating users and providing users control over the collection, use and sharing of private information with third parties. Adoption of Privacy Enhancement Technologies (PETs) such as demonstrated in this prototype may present exciting social opportunities to organizations. Privacy-respecting approaches will help organizations to build their trust with their customers and will more likely improve the business.

The utility of extending UML with privacy icons and controls is visible from the use-case scenarios discussed in chapter 5. Adoption of such PETs will make it very convenient for the organization to integrate user privacy from the beginning of the software development phase rather than focusing on privacy after the product or service is coded. This will help organizations to produce cost effective solutions and help them to minimize privacy-related violations.

This work can be extended to other development IDEs as mentioned in the Chapter 4. It may be extended to create a privacy design framework for major programming languages such as Java, .NET, PHP and many others. A newer version of UML may be introduced with privacy controls allowing the software developers and architects to facilitate modeling of privacy at design time.

REFERENCES

Object Management Group - UML, (n.d.) . Retrieved January 22, 2013, from Object Management Group - UML: http://www.uml.org/

Getting Started Programming Application-Level Add-Ins. (n.d.). Retrieved October 15, 2012, from Getting Started Programming Application-Level Add-Ins: http://msdn.microsoft.com/en-us/library/ms268878.aspx

NetBeans IDE - Overview. (n.d.). Retrieved October 15, 2012, from NetBeans IDE: http://netbeans.org/features/index.html

Office Solutions Development Overview. (2012). Retrieved October 18, 2012, from Office Solutions Development Overview: http://msdn.microsoft.com/en-us/library/hy7c6z9k.aspx

A Brief Introduction to XACML. (n.d.). Retrieved February 18, 2013, from Oasis-open.org: https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html

Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., & Rjaibi, W. (2005). Extending relational database systems to automatically enforce privacy policies. *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference*, 1013-1022.

Ashley, P., Hada, S., Karjoth, G., Powers, C., & Schunter, M. (2003). *Enterprise Privacy Authorization Language (EPAL 1.2)*. IBM Research.

Ashley, P., Hada, S., Karjoth, G., Powers, C., & Schunter, M. (2003). *Enterprise Privacy Authorization Language (EPAL)*. Zurich, Switzerland: IBM Research.

Beresford, A., & Stajano, F. (2003). Location Privacy in Pervasive Computing. *Pervasive Computing, IEEE*, 46-53.

Bodorik, P., & Jutla, D. N. (2008). Privacy with Web Serivces: Intelligence Gathering and Enforcement. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 546-549.

Bodorik, P., Jutla, D. N., & Dhillon, I. (2009). Privacy compliance with Web Service. *Journal of Information Assurance and Security*, 412-421.

Brodie, C., Karat, C.-M., Karat, J., & Feng, J. (2005). Usable Security and Privacy: A Case Study of Developing Privacy Management Tools. *SOUPS '05 Proceedings of the 2005 symposium on Usable privacy and security*, 35-43.

Byun, J.-W., & Li, N. (2008). Purpose based access control for privacy protection in relational database systems. *The VLDB Journal — The International Journal on Very Large Data Bases*, 603-619.

Canny, J. (2002). Collaborative Filtering with Privacy. *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium*, 45-57.

Cavoukian, A. (2011). *Privacy by Design.* Toronto, Ontario: Information and Privacy Commissioner of Ontario.

Cavoukian, A. (2013). *About PbD*. Retrieved January 5, 2013, from Privacy by Design: http://www.privacybydesign.ca/index.php/about-pbd/

Cavoukian, A. (n.d.). *About PbD*. Retrieved December 5, 2012, from Privacy by Design: http://www.privacybydesign.ca/index.php/about-pbd/

Cranor, L. F. (2002). *Web Privacy with P3P.* O'Reilly & Associates.

Denker, G., Kagal, L., & Finin, T. (2005). *Security in the Semantic Web using OWL.* Information Security Technical Report.

Facebook Data Use Policy. (n.d.). Retrieved January 7, 2013, from Facebook: https://www.facebook.com/full_data_use_policy

Ferraiolo, D., & Kuhn, R. (1992). Role Based Access Controls. *15th NIST-NCSC National Computer Security Conference*, 554-563.

Ghazinour, K., & Barker, K. (2009). A Lattice-based Privacy Aware Access Control Model. *International Conference on Computational Science and Engineering*, 154-159.

Ghazinour, K., Majedi, M., & Barker, K. (2009). A Model for Privacy Policy Visualization. *International Computer Software and Applications Conference*, 335-340.

Ghazinour, K., Majedi, M., & Barker, K. (2009). A Model for Privacy Policy Visualization. *33rd Annual IEEE International Computer Software and Applications Conference*, 335-340.

Godik, S., & Moses, T. (2005, 02). Extensive Access Control Markup Language (XACML) version 2.0. Oasis Standard. OASIS.

Goudalo, W., & Seret, D. (2008). Toward the Engineering of Security of Information Systems (ESIS): UML and the IS Confidentiality. *The Second International Conference on Emerging Security Information, Systems and Technologies*, 248-256.

How to: Create UML Modeling Projects and Diagrams. (n.d.). Retrieved October 1, 2012, from How to: Create UML Modeling Projects and Diagrams: http://msdn.microsoft.com/en-us/library/dd409445.aspx

IBM developerWorks - Are you new to Rational? (n.d.). Retrieved September 20, 2012, from IBM developerWorks: http://www.ibm.com/developerworks/rational/newto/

Jutla, D. N. (2012a). *Report on Privacy Governance for Software Organizations and Mobile App Developers.* Commissioned by the Office of the Privacy Commissioner of Canada.

Jutla, D. N. (2012b). Presentation to the Privacy Commissioner of Canada and her staff. Ottawa: Office of the Privacy Commissioner on contents of reports on Privacy Governance for Software Organizations and Personal data that mobile apps developers are monetizing and sharing with others, March 8, 2012.

Jutla, D. N. (2012c). Privacy Governance for Software Organizations. *European Identity and Cloud Conference.* Munich.

Jutla, D. N. (2012d). In OASIS Privacy by Design for Software Engineers' Call for Participation: https://www.oasis-open.org/news/announcements/call-for-participation-privacy-by-design-documentation-for-software-engineers-pbd.

Karjoth, G., & Schunter, M. (2002). A Privacy Policy Model for Enterprises. *15th IEEE Computer Security Foundations Workshop (CSFW'02)*, 271-281.

Karjoth, G., & Schunter, M. (2002). A Privacy Policy Model for Enterprises. *Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE*, 271-281.

Lobo, J. (2009). CIM Simplified Policy Language (CIM-SPL). Distributed Management Task Force (DMTF).

Massacci, F., & Naliuka, K. (2008). Towards Practical Security Monitors of UML Policies for Mobile Applications. *The Third International Conference on Availability, Reliability and Security*, 1112-1119.

Md. Moniruzzaman, Ferdous, M., & Hossain, R. (2010). A Study of privacy policy enforcement in access control models. *f 13th International Conference on Computer and Information Technology (ICCIT 2010)*, 352-357.

Microsoft Visual Studio 2010 Visualization and Modeling Feature Pack. (n.d.). Retrieved 10 02, 2012, from Microsoft Visual Studio 2010 Visualization and Modeling Feature Pack: http://msdn.microsoft.com/en-us/library/dd460723(v=vs.100).aspx

Miles, R., & Hamilton, K. (2006). *Oreilly Learning UML 2.0.*O'Reilly Media.

Morcillo, P. G., & Lázaro, A. J. (2012). *UMU XACML Editor*. Retrieved November 8, 2012, from University of Murcia: http://xacml.dif.um.es/

Naedele, M. (96-98). Standards for XML and Web Services Security. *Security*, 2003.

Ni, Q., Bertino, E., & Lobo, J. (2008). An obligation model bridging access control policies and privacy policies. *SACMAT '08 Proceedings of the 13th ACM symposium on Access control models and technologies*, 133-142.

Office Development in Visual Studio. (n.d.). Retrieved October 15, 2012, from Office Development in Visual Studio: http://msdn.microsoft.com/en-us/library/d2tx7z6d.aspx

Office development with Visual Studio (VSTO). (n.d.). Retrieved September 8, 2012, from Office development with Visual Studio (VSTO): http://msdn.microsoft.com/en-US/office/hh133430

Olivier, M., & Oberholzer, H. (2005). Privacy Contracts as an Extension of Privacy Policies. *Data Engineering Workshops*, 1192-1193.

Papyrus. (n.d.). Retrieved September 5, 2012, from Papyrus: http://www.eclipse.org/modeling/mdt/papyrus/

Pitofsky, R., Anthony, S., Thompson, M., Swindle, O., & Leary, T. (2000). *Privacy Online: Fair Information Practices in the Electronic Marketplace, A Report to Congress.* US Federal Trade Commission.

Rissanen, E. (2010). XACML v3.0 Privacy Policy Profile Version 1.0. Oasis.

Rodríguez, A., & Mario Piattini, E.-M. (2006). Security Requirement with a UML 2.0 Profile. *First International Conference on Availability, Reliability and Security (ARES'06)*.

Ross, R., Porter, E., Stine, K., Stoneburner, G., Hodge, B., Fabius, J., et al. (2012). *Security and Privacy Controls for Federal Information Systems and Organizations.* National Institute of Standards and Technology, U.S. Department of Commerce.

Sabo, J., Willett, M., & Janssen, G. (2013). OASIS Privacy Management Reference Model (PMRM) TC. Oasis.

Sabo, J., Willett, M., Brown, P., & Jutla, D. N. (2012). Privacy Management Reference Model and Methodology (PMRM) Version 1.0. Oasis.

Sabo, J., Willett, M., Brown, P., & Jutla, D. N. (2012). Privacy Management Reference Model and Methodology, OASIS PMRM TC Standards Track Committee Draft, March 26, 2012.

Sabo, J., Willett, M., Brown, P., & Jutla, D. N. (2013). Privacy Management Reference Model and Methodology, OASIS PMRM TC Standards Track Committee Specification, March 25 2013.

Sandhu, R., Ferraiolo, D., & Kuhn, R. (2000). The NIST model for role-based access control: towards a unified standard. *RBAC '00 Proceedings of the fifth ACM workshop on Role-based access control*, 47-63.

Spiekermann, S., & Cranor, L. F. (2009). Engineering Privacy. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 67-82.

Spiekermann, S., Grossklags, J., & Berendt, B. (2001). E-privacy in 2nd Generation E-Commerce: Privacy Preferences versus actual Behavior. *Third ACM Conference on Electronic Commerce*.

Sun, W., France, R., & Ray, I. (2011). Rigorous Analysis of UML Access Control Policy Models. *Policies for Distributed Systems and Networks (POLICY)*, 9-16.

UML. (n.d.). Retrieved August 10, 2012, from UML - NetBeans Wiki: http://wiki.netbeans.org/UML

UML, business process, requirement and database design plug-in for NetBeans. (n.d.). Retrieved Auguste 12, 2012, from UML, business process, requirement and database design plug-in for NetBeans: http://www.visual-paradigm.com/resource/netbeans-plugin.jsp

Visio Object Model Overview. (n.d.). Retrieved August 19, 2012, from Visio Object Model Overview: http://msdn.microsoft.com/en-us/library/cc160740.aspx

Visual Paradigm for UML 10.1 Community Edition. (n.d.). Retrieved January 22, 2013, from Free UML Tool with full UML, ERD and SysML Supports: http://www.visual-paradigm.com/product/vpuml/editions/community.jsp

Wenning, R. (2007). *Platform for Privacy Preferences (P3P) Project*. Retrieved October 15, 2012, from http://www.w3.org/P3P/

Xfig User Manual. (n.d.), Retrieved 04 09, 2013, http://xfig.org/userman/frm_introduction.html

Yagüe, M. (2006). Survey on XML-Based Policy Languages for Open Environments. *Journal of Information Assurance and Security*, 11-20.

Yang, N., Barringer, H., & Zhang, N. (2007). A purposebased access control model. *Third International Symposium on Information Assurance and Security*, 143-148.

Yu, W., & Murthy, S. (2007). PPMLP: A special modeling language processor for Privacy policies. *Computers and Communications, 2007. ISCC 2007. 12th IEEE*, 851-858.

Yu, W., & Murthy, S. (2007). PPMLP: A Special Modeling Language Processor for Privacy Policies. *Computers and Communications 12th IEEE Symposium*, 851-858.

APPENDIX A : USE-CASE DETAILS

A. Manage Appointments

The use-case allows the UR to manage appointments for any scheduled clinic held by any doctor. The UR can view appointments, book a particular appointment, edit an appointment or delete it. The use-case allows the UR to view the appointments in three different ways i.e. by Day, by Week or by month.

A.1 Flow of Events

A.1.2 Basic Flow

This use-case starts when the Unit Receptionist (UR) wishes to manage appointments of a particular Doctor held in a clinic.

1. The system displays the default clinic of user. Extend **<Validate Authorized Clinics>**.
2. The UR selects the Doctor for the particular Clinic.
3. The system fetches all the appointments for the doctor at that clinic from today onwards.
4. The system fetches all the active daily and exceptional scheduled clinics for the doctor for all days and dates following today respectively.
5. The system fetches all the scheduled clinics from today onwards.

6. The system fetches the next available slot for Initial and Follow up Appointments for the doctor at the specified clinic.
7. The system displays the next available slot dates for Initial and Follow up Appointments.
8. The system selects the earliest of the next available Initial OR next available Follow up appointment date and displays as the date selected.
9. The system selects the 'Week View option' and 'view all slots' by default.
10. The system displays the appointments for the doctor for next five scheduled clinic starting from first available slot date.
11. In case the UR is a Ward UR, the system displays all slots that belong to the ward only.
12. The UR enters the Date for which he wishes to view the appointment.
13. The system validates the date by checking that this date is not before today's date. If it is, the system gives a warning message and does not allow the UR to proceed.
14. The system checks if the day is a holiday or the clinic is cancelled and displays a warning message.
15. The system retrieves appointments for the day selected and displays on screen.
16. Upon display of appointments, the UR can either select an empty slot or a particular appointment to edit or delete or insert a new slot. For each of the scenarios mentioned a separate sub-flow is executed as described below.
17. If the UR selects a slot that belongs to the ward and appointment has already been given on that slot, the system warns the UR.

Sub Flow A – Edit Appointment

1. If the clinic is cancelled or on public holiday without exceptional schedule, the system will not allow the UR to edit the appointment.
2. If the UR selects a slot system displays the following information.
    - Clinic
    - Doctor
    - Appointment Date – Day
    - Appointment Time
    - Appointment Booking Date
3. If the UR selects a slot that has a booked appointment, the system displays the details of the appointment as follows:
    - MR# (if exists)
    - Name
    - Work Phone
    - Home Phone
    - Temporary Phone
4. If appointment was booked with patient MR No. then system displays the following information.
    - Address Line 1
    - Address Line 2
    - Address Line 3
    - Email Address
    - Sex

- Area
- City
- Postal Code
- Mobile No
- Birth Date

5. The system displays following appointment information. User may edit any of this information.
   - Appointment Nature
   - Referring Location
   - Advance Amount
   - Payment Mode
   - Remarks
6. The system displays previous receivables of patient. Extend use-case **<View Previous Receivables>.**
7. The system checks if the patient's cataract surgery has occurred in the past 30 days and checks if advance for the visit is present or not. If the advance is present the system picks the advance amount and displays it as Cataract advance.
8. The UR can edit any field in the record displayed above apart from Patient MR #. The system will not allow editing of this advance amount if it belongs to the cataract advance category. The system will also not allow change of appointment nature if changing of appointment nature is not allowed for the doctor.
9. The UR asks to save the appointment.
10. The system saves the appointment.

11. The system updates any change in the patient's phone number to the patient's demographic data in MRI system.
12. In case the system has picked the Cataract advance of the patient, it updates the cataract advance transactions for the patient and updates the number of free visits allowed now.
13. Control is transferred to step 19 of the Basic Flow.

Sub Flow C – Enter Appointment

1. If the clinic is cancelled, the system will not allow the UR to enter the appointment.
2. The UR enters the MR# of the patient. UR may search the MR# from external actor <MRI>.
    i. The UR can also book an appointment for a patient who does not have an MR#. The booking is made in the name of the patient instead of MR #.
3. System displays the information described in step 2-4 in Sub Flow C.
4. The system displays previous receivables of patient. Extend use-case **<View Previous Receivables>**.
5. The system checks if the patient's cataract surgery has occurred in the past 30 days and checks if advance for the visit is present or not. If the advance is present the system picks the advance amount and displays it as Cataract advance.
6. The system checks if an unadjusted payable exists for the same patient for the same clinic and doctor and if it does the system records it as the current advance and displays the message and advance amount on the screen.
7. The system displays the current date as the booking date for the appointment.

8. The system checks whether appointment nature is allowed to change on slots for the doctor.
    i. If change of appointment nature is allowed for the doctor, the UR can edit the appointment nature if he desires.
9. System prompts the UR to enter referring location.
10. The UR enters the remarks.
11. If the appointment is given by MR #, the system prompts the UR to enter the advance payment mode and amount.
12. The system checks if the patient's email address is present in the MRI System. If the patient's email address is present in the MRI System, the UR may also mark that the booking be emailed to the patient.
13. The UR asks the system to save the appointment.
14. The system saves the appointment.
15. The system updates any change in the patient's phone number to the patient's demographic data in MRI system.
16. In case the system has picked the Cataract advance of the patient, it updates the cataract advance transactions for the patient and updates the number of free visits allowed now.
17. If the UR has chosen to mail the booking to the patient, the system finds the patient's email address from MRI System.
18. The system generates an email to the patient's email address, specifying the booking details.

19. If the UR has chosen the selected clinic as the default clinic, the system saves the default clinic of the UR.

20. Control is transferred to step 19 of the Basic Flow.

Sub Flow B – Delete Appointment

If the UR selects to delete a particular appointment,

1. System prompts the user "Are you sure you want to delete the appointment", if user replies in affirmative then system removes the appointment from the schedule and an empty slot as a result of the cancellation of the appointment becomes visible to the UR.
2. If the deleted appointment contains advance, the system gives the message "Advance has been paid for this appointment. Are you sure you want to delete this appointment". If user replies in affirmative then system records the advance as patient payable amount to that clinic on that doctor and removes the appointment.
3. Control is transferred to step 19 of the Basic Flow.

Sub Flow D – Insert Appointment

If the UR wishes to insert a new appointment slot, the system does the following

1. The system checks whether a new appointment can be inserted on 'this' schedule or not. System will not allow user to insert slot in following cases
    - Cancelled Clinic
    - Public Holiday with no exceptional schedule
    - Slot insertion for the day is not allowed

2. Incase user has the rights to insert a slot then system will allow the user to insert slots even if slot insertion for that particular clinic and doctor is not allowed.
3. If a new appointment slot can be inserted, the system prompts the user to enter the new time slot. System will not allow the slot insertion if slot insertion for that time is not allowed.
4. Control is transferred to step 1 of Sub Flow B – Enter appointment where the UR is expected to enter a patient's MR # or name to book appointment.

Sub Flow F – Month View

1. System displays all the dates of current month and all the scheduled clinics are displayed with different color coding.
2. System retrieves and displays the following information for all the scheduled clinics.
    - Total Initial Slots
    - Available Initial Slots
    - Total Follow-up Slots
    - Available Follow-up Slots

Sub Flow G –Next available slot for Specialty

1. System prompts the user to identify the specialty.
2. The system extracts the doctors with the default specialty same as the specialty entered.
3. The system retrieves and displays the following information for all such doctors.
    - Clinic Id
    - Clinic Description

- Doctor
- Doctor Name
- Initial Slots
- Follow-up Slots

Sub Flow H – View Available slots only

If the UR selects to view only the available slots, the system displays all slots that do not contain any appointments.

Sub Flow I – View Next Appointments

If the UR selects the next option following actions will be performed in different views.

- Month View: Next month will be displayed
- Day View: Next scheduled day will be displayed.
- Week View: Schedule of next five scheduled days will be displayed.

Sub Flow J – View Previous Appointments

If the UR selects the previous option following actions will be performed in different views.

- Month View: Previous month will be displayed
- Day View: Previous scheduled day will be displayed.
- Week View: Schedule of next five scheduled days will be displayed.

Sub Flow K – Physician's Next Available Clinic

1. System will prompt the user to input Doctor Mnemonic. User may search doctor mnemonic using lookup.
2. System will display name of doctor on providing valid doctor mnemonic.

3. System will retrieve and display following information regarding all the clinics of doctor with their first available slot.
    - Clinic Id
    - Clinic Description
    - Appointment Nature
    - First Available Date
    - Day
4. User may select any authorized clinic and control will be transferred to step#4 of Basic Flow.

Basic Flow – Cotd.

1. The system updates the displayed appointments newly booked appointment.
2. The system re-fetches the next available slot for Initial and Follow up appointments and displays on screen.

The use-case ends here.

A.1.3 Alternative Flow

None.

A.2 Special Requirements

The system should follow the standard Graphical User Interface guidelines as described in the relevant document.

A.3 Preconditions

User must be logged on to the system.

A.4 Post Conditions

If the use-case is successful, the system updates the record otherwise the System State is unchanged.

A.5 Extension Points

None.

B. Record History

This use-case allows the doctor to record the patient's purpose of visit and previous background/history.

B.1 Flow of Events

B.1.2 Basic Flow

This use-case is initiated by doctor to enter/edit the background/history of the patient.

1. This use-case gets following inputs from calling use-case.
    - MR #
    - Visit Date
    - Problem
    - Family Background which includes:
        - Job History
        - Family Members Details
        - Family Members History
2. The use-case ends here.

B.1.3 Alternative Flow

None

B.2 Special Requirements

None

B.3 Preconditions

1. Doctor must have rights to view and modify patient history.
2. Doctor must be logged on to the system and patient must be a registered patient.

B.4 Preconditions

None

B.5 Extension Points

None

C. View History

This use-case allows the doctor to view the records of the patient's previous visit's history/treatment/medicine record etc.

C.1 Flow of Events

C.1.2 Basic Flow

This use-case is initiated by doctor to enter/edit the background/history of the patient.

1. This use-case gets following inputs from calling use-case.
    - MR #
2. System retrieves and displays the following information.
    - Visit #
    - Patient Personal Detail
        - First Name and Last Name
        - DOB
        - Contact Details
        - Address
        - Family Medical history
        - Patient Medical History
    - Test Record
    - Medicine Record
    - Comments of the previous visits
2. The use-case ends here.

C.1.3 Alternative Flow

None

C.2 Special Requirements

None

C.3 Preconditions

1. Doctor must have rights to view and modify patient history.
2. Doctor must be logged on to the system and patient must be a registered patient.

C.4 Preconditions

None

C.5 Extension Points

None