Project Report

TDT4290 - Customer Driven Project
"Privacy Advisor"

NTNU, Fall 2011

Contents

Introduction

Bla. blabla

Project Directive

Contents

0.1 Purpose

This document describes the mandate, background, resources available and organizational structure of the project A Privacy Advisor, henceforth referred to as the project.

0.2 Mandate

The purpose of this project is to implement the key functionality of a privacy agent as described in Nyre and Tndel (2010), that provides users with advice in making Internet privacy decision.

0.2.1 Background

This project is a part of a larger research project at SINTEF ICT that studies approaches to handling Internet privacy related issues. The underlying idea is that while users are often concerned about the way various websites and services handle private information about them, obtaining information about this is very costly as privacy policies tend to be very long documents formulated in an inaccessible language. This has led to the idea that Internet privacy can be handled by machine learning techniques, where a particular decision is based on the users past behavior and the behavior of similar users.

Nyre and Tndel has then proposed a Privacy Agent structure that uses the case based reasoning (CBR) method for giving privacy advice. CBR is in many ways similar to the way human experts reason about problems, and is usually described as a process in four stages (here related to the privacy decision problem):

- Retrieve: Given a new site, the agent will retrieve from its knowledge base, the set of cases deemed the most similar to the one at hand. This means, that if presented with the site Facebook, for instance, the agent finds Twitter, Google and LinkedIn to be the sites that have the most similar policy to Facebook.
- Reuse: Look at the decisions made about the cases that were found and adapt this decision to the problem at hand. In this case, the agent needs to see if there are strong enough indications toward a particular behavior with respect to the type of site that is at hand. If for instance, the user has accepted the policies of all the similar sites found, he is also likely to accept that of Facebook.
- Revise: Once a conclusion is reached, it is presented to the user (along with the background for why it is reached). The user may then choose to accept the conclusion, or to overrule it, providing the system with directions as to why it was wrong. This may in turn cause the agent to update its parameters accordingly.
- Retain: Finally, the new case is stored to the database along with the users decision as a reference for the next time the same site is opened, and as a case to employ when evaluating new sites.

Nyre and Tndel also describes this local CBR approach to be complemented by a community database where the same information is stored, allowing for a second lookup that uses a collaborative filtering, that is, making a decision based on the behavior of similar users.

0.3 Objectives

This project identifies three key objective, arranged by order of importance: Implementing a testing framework of CBR based privacy agent that is able to make privacy decisions based on previous user behavior. Implement the community system/collaborative filtering part of the agent. Extend the system to other standards for machine readable privacy policies. Implement the system as a browser plugin. This is considered least important, and is contingent on the success of early testing. It is also given a low priority given the relative small portion of major websites that implement P3P.

0.4 Resources and Duration

The system in its complete form is to be demonstrated on November 24 2011. For the project period, a total of 25 hours per week per project member is planned. With seven group members and a project spanning 13 weeks, this adds up to approximately 2300 hours.

0.5 Organization

The project group organization is based on the modules of the system that is being implemented. One group member is responsible for developing one particular feature. This organization is shown in Table 1.

0.6 Planning

A project plan has been developed for the purpose of communicating expectations and progress within the group and to the customer and the advisor. The plan also serves as an aid in identifying problems and project management. For the software development process, a hybrid waterfall model has been chosen.

0.7 Limitations and Scope

The primary focus of this project is on developing a framework that allows for testing the CBR privacy agent framework. This entails building a module for parsing policy documents in XML format, a data structure for holding policy information in memory (henceforth policy objects), a set of exchangeable distance metric that compares policy objects, a generic retrieval algorithm (such as k Nearest Neighbors) that works with any distance metric and methods to store and update a knowledge base. Being a part of an ongoing research project, reusability and modularity are important success factors for in evaluating the project. This means that it should for instance be simple to swap P3P with

Table 1: Responsibilities

Role	**Description**	**Responsible**
Administrative	Customer relations Requirement specification Requirement specification Planning documents Meeting minutiae Project report	Ulf Nore
Software design and Architecture	UML modeling. Design report. User documentation. Technical Decisions.	Nicholas Gerstle
Data Storage/Databases	Flat file data storage system. Database systems .	Amanpreet Kaur
CBR - Algorithms and Data structures	Data structures for storing privacy policy information. Define and implement similarity metrics. Retrieval and learning algorithms. Parameter storage.	Dimitry Kongevold, Neshahavan Karunakaran
Testing and Evaluation	Design test cases. Criteria/methodology for model testing.	Henrik Knutsen
GUI	Implement a simple GUI for testing model framework.	Ulf Nore
Version control	Set up and maintain code repository.	Einar Afiouni
XML/P3P Parser	Implement P3P parser that produces inputs to CBR	Einar Afiouni
Version control	Set up and maintain code repository.	Einar Afiouni

some other privacy policy standard, that different distance metrics should be applicable, new metrics could easily be implemented and so forth.

Planning Phase

Contents

0.8 Purpose

This document details the different phases in the development process. It presents the development model that has been chosen, relating the choice of model to the particular challenges faced by the nature of the project. It also describes the set of activities included in each phase and a set of activities that are ongoing throughout the lifetime of the project.

0.9 Development Model

As the project is an early implementation phase of a larger research project, there is an inherent uncertainty as to several details regarding both the implementation and the appropriateness of the underlying model. This has several implications for the development process.

Following the above discussion, a hybrid waterfall model is settled upon. Here the design, implementation and testing phases follow a cyclical pattern. In concrete terms, this means that a first prototype will be developed and then put through some tests, and depending on the results of these tests, there may be revisions in the software design.

Furthermore, there are clearly overlaps between these three phases. A key part of the documentation work consists of documenting the source code, which is basically commenting during development. Furthermore, as implementation is underway, it may turn out that the design needs adjustments, due to unforeseen factors.

0.10 Phases

0.10.1 Planning

0.10.2 Prestudy and Research

In this phase the aim is for each project member to acquire a certain level domain knowledge in the field of Internet privacy and to learn the necessary technology and tools required to implement the model as proposed by the customer. This entails having a working knowledge of the Java programming language, version control using Git and the CBR framework.

0.10.3 Requirement Specification

The requirements specification is a document listing the functional and non-functional requirements of the software to be developed, which is a standard that the results is to be measured against, thus serving as not only a contract between the customer and the project team, but as a basis for developing testing methods.

0.10.4 Design/Architecture

This phase consists of a broad structuring and specification of the overall system. It defines the program structure in terms of program flow, modules, classes and interfaces as well as coding standards and other conventions that will serve as guidelines for the implementation phase.

0.10.5 Implementation

In this phase the design is realized as a working Java program according to the models developed in the Design phase.

0.10.6 Evaluation and Documentation

This phase consists of testing the system and documenting the structure of the system and how it is operated. From a software engineering perspective, the primary testing grounds are against the standards prescribed by the requirements specification rather than applicability of the underlying models performance. As mentioned, among the primary objectives of the project is to provide a testing framework to verify the applicability of the given system in making privacy decisions.

0.10.7 Ongoing Activities

Reporting and Administrative Tasks

Under this heading are more project management related activities, such as routine organizational work (ie. arranging meetings and writing status updates), more refined distribution of tasks as the project is underway, and preparation of the project report (this document).

Study and Lectures

To solve several of the problems posed by this project, most group members have had to learn new tools and technologies. This includes, but is not limited to Case Based Reasoning, version control (Git), certain features of Java and so on. Lectures on project management and software development are also subsumed under this heading.

Preliminary Study

Contents

0.11 Introduction

Requirements Specification

Contents

0.12 Purpose

This requirements specification has been prepared for and accepted by SINTEF ICT (the customer) stating the requirements for the software system to be developed for the course TDT4290: Customer Driven Project. The requirements span two categories; functional requirements, describing the functionality the software needs to supply, and non-functional requirements, describing the development process.

The requirement specification, once accepted by the customer, will serve as a contract between the parties involved, being a guideline for design and implementation and the standard against which the product is evaluated.

0.13 Introduction

0.13.1 Background and Similar Software

SINTEF ICT is currently investigating new approaches to privacy protection of end-users. Tndel and Nyre (2011) proposes a specific agent design for a machine learning approach to advice users on privacy actions based on:

- Past behavior using case based reasoning (CBR)
- Similar users behavior in similar situations using collaborative filtering (CF)

While there are systems for aiding users in making privacy related decisions, the majority of these systems rely in a large extent on the user pre-specifying his preferences and being prompted with messages about where the policy of a given site conflicts with the users preferences. Our design aims at being low profile or non invasive, that is able to make sensible decisions with as little interference as possible, and at the same time, given as little feedback as possible, able to cater for the dynamic nature of both web sites privacy policies and user preferences with respect to privacy.

0.13.2 Scope

The primary aim of this project is the implementation of the core classification system described in Tndel and Nyre (2011) to allow for testing the applicability of the suggested approach to predicting privacy preferences. Since the software is intended to be a part of a research project, a design that allows for testing of various hypotheses and models is required. This implies a highly modular design where the various components of the core system can be replaced.

Furthermore, given the research nature of the project, less emphasis is placed on developing a complete stand-alone application. The core focus for our project will be on developing the underlying system and an interface for testing and parameter estimation. Hence two distinct systems are to be developed, both built around the same core functionality (I.E. the privacy advisor agent). Owing to the fact that this software is developed around a research project, the emphasis is on the first system:

1. A testing system that can be fed a knowledge base consisting of input-output mappings (P3p + context -¿ decision), and run interactive tests on a sample where the user is allowed to give feedback to the system and see the explanation for the recommendation. We envision a dual CLI/GUI (command line interface and graphical user interface) solution for this. In a final product, this testing system can also be used for the purpose of calibrating the model.
2. An end-user system that can run as a browser plug-in giving real time advice to the user as he browses the web.

0.13.3 Overview

This document is organized as follows: Section 2 gives an overview over the system; its requirements and user characteristics. Section 3 presents four different use-case scenarios. Section 4 presents specific functional and non-functional requirements.

0.14 Overall Description

0.14.1 System Description

The overall structure of the system is detailed in Tndel and Nyre, and consists of the local CBR reasoning system, the remote/community collaborative filtering, both with their respective databases for storing information. This is in turn linked to an interface that is able to read and parse P3P policy files that are retrieved either from a local file (for the testing system) or by retrieving from the web.

Figure 1: End User System Flowchart

0.14.2 User Interface

Because of the research nature of the project, the customer considers the user interface to be of small importance. As the underlying algorithm/methodology is in an early development phase, the core focus is placed on producing a system for model testing and evaluation rather than an end user interface.

0.14.3 Hardware and Software Interface

Being written in Java, the software requires a local copy of the Java Runtime Environment (JRE) installed on the computer.

For the community functionality (collaborative filtering), a dedicated server running the filtering engine must also be available. Since this is basically a modified version of the local server, it has similar requirements, but as it presumably will hold a larger knowledge base, its hardware requirements will be greater, as both lookup time (computational demands) and storage demands will increase with the number of users. It may also require additional server/database software such as mySQL, CouchDB etc.

0.14.4 User Characteristic

For this project we distinguish between two groups constituting the users of the product.

Developers/Researchers

Firstly, developers/researchers that will be working on the testing and calibration of the underlying model and extending it to other policy types beside P3P etc. These users are the primary focus of our work. A research/developer is an expert user, and needs to be familiar with how privacy policies are coded in machine-readable form such as P3P, but also the software source codes in order to modify, extend and optimize the algorithms.

End Users

Secondly, the end user who will be using the software in the form of a browser plugin that provides advice with respect to the users behavior on the Internet. A key objective for the project is that the agent is to be able to make good decisions and require as little feedback as possible from the user. To the extent interaction is needed, it should be able to clearly state an explanation for its decisions and allow the user to override in a simple manner.

0.15 Use Cases

The first use case illustrates a research setting where calibration/testing interface allows the user to load in a dataset of P3P policies and test the performance of the underlying model.

The last three use cases illustrate the potential application of the system as a browser plugin that runs in the background monitoring the users activities and the web sites he is visiting. As previously stressed, the success of testing according to the first use case determines the extent to which the system described in cases 2-4 is implemented.

Case 1: Research/calibration

In this case a researcher wants to test the properties of the underlying model. Using the Calibration GUI, he imports 50 P3P policies that are parsed. Further he designates that 40 of these are to be stored immediately in the knowledge base along with a corresponding action for each policy. The user now specifies the distance metric he wants to apply to each of the different components. Finally, he can either set the (importance) weights assigned to each of the policy components, or he can load the weights from a flat text file. Now that the configuration is complete, the ten policies withheld earlier from the sample can be classified. For each of the ten policies, the user can choose either to accept, or reject, and provide a reason for his rejection before proceeding to the next policy.

Case 2: End user local query, recommendation accepted, site rejected

A user visits a previously unvisited website. The privacy agent tries to retrieve machine-readable privacy information from the site. When the policy is obtained it is parsed and a context object, consisting of the policy, domain, time of visit, and other contextual information, is created. The context object is compared to the local database for similar contexts. Since the user has visited sites with a similar policy previously, the comparison succeeds and the site is blocked based on data from the local database. The user agrees with this decision and navigates away from the site.

Case 3: End user local query, site approved by recommender, recommendation accepted

A user visits a previously unvisited website. As before, the system the system fetches the necessary data to do a local query. This query indicates, with sufficient confidence, that the sites policy is acceptable. The user is then allowed to continue browsing with no intervention from the Privacy Agent.

Case 4: End user global query, recommendation overridden

As before, but in this case, no sufficiently similar cases are found locally. In this case the system will query the global server for similar users that have visited the same site to base its decision on this. In this case, site is blocked, but the user disagrees. He selects an override feature and gives a reason for why he overrides.

0.16 Specific Requirements

0.16.1 Product perspective

As described in Section 1.2, as the main goal of the project is to develop a testing framework for the core reasoning system. The secondary goal is to implement a user interface that can work as a stand-alone application to allow for actual user testing.

0.16.2 Functional Requirements

- The system should be able to parse a P3P file to instantiate the data as a privacy case/event/instance.
- Based on past history (knowledge base), it should retrieve the cases most similar to the one presented.
- Given the degree of similarity to past cases and the uniformity of action taken in the past, the system can either
  - Give the user a recommendation a recommendation or
  - Pass the recommendation decision on to the community/CF system.
- If passed on to the CF, the system will query a server for the most similar users and use the data on their decisions in similar cases to make a recommendation (along with local/CBR recommendation)
- Update the database with the recommendation.
- Allow the user to view the explanation for the recommendation
- Allow the user to overrule a recommendation.
- When overruling a recommendation, the user must be allowed to explain why the decision is made, e.g. one time occurrence, permanent rule, etc.
- Allow the user, if making a new general rule, to backtrack and alter previous cases

0.16.3 Non-Functional Requirements

- Implementation
  - Code is written in Java following Sun Microsystems conventions (http://www.oracle.com/technetwork/j
    136057.html).
  - Third party libraries are to be documented with version numbers and to be included in the installation package.
- Maintainability:
  - Code repositories and version control: github is used as code repository and for version control.
  - User documentation is to be produced.
  - A well documented API is to be designed
  - English (US) is to be used as language for naming convention for source code and filenames, and in code comments and documentation.
  - The code is to be designed in a modular fashion.
- Performance:
  - For the final end-user product that will run as a browser plug-in, performance will be important, as the program should not be seen as a nuisance in getting work done.
- Portability:
  - The testing/design system should be portable to any system with a JRE.
- User interface:
  - Two UIs are to be implemented: A command line interface (CLI) as well as a GUI is to be designed using Java/swing.
  - These interfaces are meant to facilitate testing the model framework.

Design

Contents

0.17 Purpose

This document describes the design phase of the program, where the program architecture is established. Several critical decisions are made in this phase and the design and architecture decisions impacts the way the implementation phase proceeds as it defines how the final software system is decomposed into modules, and how these modules behave and interact with each other. Details regarding programming languages, algorithms, data structures, coding standards and other software engineering features must also be established prior to proceeding from this phase.

0.18 Development Tools and Technologies

This section details some of the choices that were made regarding development tools and technologies for this project.

0.18.1 Documents and Source Code Repositories

For software projects of a certain scale, version control is an important technology that allows project members to work simultaneously against the same files without causing inconsistencies. Version control systems also allow for comparison with older versions, tracking changes and restoring previous copies in the case of errors.

For source code repositories and version control, **Git** was selected. Git is a distributed, open source version control system that is available for all platforms. Git is also used for hosting the LaTeXdocuments that comprise this report. For other documents such as meeting minutes, agendas, status reports, time reporting and certain planning documents, Google Docs is used.

0.18.2 Programming Languages

Java is chosen as the primary programming language for implementing the majority of the code. Java is an object oriented programming language providing a level of abstraction appropriate for the task at hand in addition to a rich set of libraries, including the SWING library for GUI programming, and several libraries for networking. It also simplifies writing a browser plugin as major browsers such as Google Chrome and Firefox employ Javascript as

0.18.3 Databases

For testing purposes, it has been decided on using flat file storage of privacy policy data using Java's `Serializable` interface. However, the output functionality is to be written in a generic fashion to simplify use of database systems such as MySQL, CouchDB and so forth.

0.18.4 Third Party Libraries

For developing the CBR as well as P3P parsing components of the Privacy Advisor, a decision had to be made regarding the usage of third party libraries, either for components or for the entire CBR system. We basically considered two options with respect to the CBR system, the first was to use a full third party CBR system (jColibri) and the second was to use a third party system for the retrieval component of the CBR system (i.e. a k Nearest Neighbors (kNN) implementation).

Third Party CBR System

The customer (SINTEF ICT) suggested looking into an open source CBR library developed at the Universidad Complutense de Madrid.

jColibri is a CBR system that has been under development for well over 10 years and is a very comprehensive system allowing for database interfaces and several other features, and is according to the customer, a popular choice in academia for CBR projects. It is also written in Java, which of course makes interfacing it simple from our own Java project.

However, its comprehensiveness also means that it takes more reading to understand and properly apply to the project at hand, and due to its size and poor documentation, jColibri was ultimately deemed unfit for the Privacy Advisor project. Due to the limited time resources available to this project, the risks associated with spending a large amount of time on a third party library that eventually would not be running was to high.

Third Party k Nearest Neighbors Implementations

Since kNN is a standard classification algorithm, there are several open source implementations available. Limiting the search space to Java implementations, a library called The Java Machine Learning Library (JavaML) was the primary candidate, as it provided a clean and simple interface and allowed for extracting confidence measures.

The problem with this library relates to the nature of distance metrics used in classifying privacy policies which is compositional in a way that is non-trivial to handle in JavaML. Furthermore, JavaML seems to operate only on arrays of floating point numbers, which means the distance metric must be defined in two stages; first mapping from policy domain to real numbers, then in terms of a metric on real vectors.

XML Parser

???

0.19 Standards

To achieve clean and reusable code, the project has adopted Oracle's Coding Conventions for the Java Programming Language[1]. This is mentioned in the requirements specification

[1] http://www.oracle.com/technetwork/java/codeconvtoc-136057.html

due to the high likelihood of the customer having to change the source code for later adaptations.

0.20 Architecture

To implement the Privacy Agent, a class structure is built around the CBR agent model discussed in Tndel and Nyre with certain additions and refinements to handle data structures, algorithms, IO and so forth.

Full Title

0.21 Introduction