Contents
Introduction .................................................................................................................. - 3 -

SMArt Project ............................................................................................................. - 3 -

This section gives a brief introduction to the SMArt platform. We give an overview of the project, explore the concepts of SMArt containers, introduce SMArt apps and give some information on the SMArt app challenge .................................................................................. - 3 -

Overview .................................................................................................................. - 3 -

SMArt Container ...................................................................................................... - 4 -

SMArt Apps ............................................................................................................... - 6 -

Challenge ................................................................................................................. - 6 -

SMArtSync ............................................................................................................... - 7 -

Meds in EMR ............................................................................................................. - 8 -

Meds and Supplements in Google Health .............................................................. - 9 -

Google Health Data Synchronization ...................................................................... - 11 -
Introduction

This report motivates and summarizes the work carried out on the SMArtSync application and its surrounding development framework during the spring semester 2011.

The first part is dedicated to introducing the SMArt (Substitutable Medical Apps, reusable technologies)\(^1\) Platform. The in 2010 started and currently rapidly evolving project aims to create a standard platform for the development of medical apps that extend the features of electronic patient data sources such as electronic medical records (EHR) or personal medical records (PHR).

The second part describes the SMArtSync application, which is medication reconciliation tool we developed for the SMArt platform. The goal of SMArtSync is to give patients the option to extract their medication data from a SMArt enable source (e.g., their local hospital), review them in an easily accessible format, and merge them into their PHR. As development progresses, this step will also include a basic form of medication-medication and medication-supplement conflict warnings.

The third part is reviewing the interaction of SMArtSync with a PHR. Currently we are focusing on interaction with the Google Health PHR, however since this part of SMArtSync is encapsulated into a separate Google Health connection server, extensions towards other PHRs will be possible in the future.

SMArt Project

This section gives a brief introduction to the SMArt platform. We give an overview of the project, explore the concepts of SMArt containers, introduce SMArt apps and give some information on the SMArt app challenge.

Overview

The goal of the SMArt project is to provide a uniform, well-defined and reusable infrastructure for diverse applications to interact with medical-record data. The creators of SMArt motivate their work with the argument that an environment that is constantly changing and evolving such as the health care system has inherit need for information technology infrastructures that

\(^1\) [http://www.smartplatforms.org/](http://www.smartplatforms.org/)
are of general purpose nature rather than monolithic and predesigned\(^2\). SMArt is promoting a concept in which a collection of simple applications each encapsulating a single task can be selected to serve in a common environment. This means that a health care institution using SMArt does not need to buy one bulky application covering all its needs, but instead can adjust its system on a very granular level. Furthermore, in the SMArt paradigm, applications are replaceable entities, meaning that as a better application for a task enters the market, a health care organization can easily replace their existing solutions without having to go through the pain of data migration. This is in some sense similar to platforms such as the Apple App Store and the iPhone. In this analogy, the organization’s SMArt container would be equivalent to the iPhone hardware that is the common ground for all developers selling software through the App Store.

SMArt is one of the projects funded by The Office of the National Coordinator for Health Information Technology through the Strategic Health IT Advanced Research Projects (SHARP) program.\(^3\)

We provide more details on SMArt containers and applications in the following sections. For further implementation details we refer the reader to the SMArt Developers Documentation\(^4\).

**SMArt Container**

In the context of SMArt platforms, the SMArt container is the entity that holds medical record data. Essentially, any medical data source (e.g., an EMR or a PHR) can be turned into a SMArt container by exposing the SMArt API (and its RDF based data model) which is used by applications for interaction. The SMArt API can be accessed in two ways, either through SMArt Connect or through SMArt REST; the former is JavaScript based and suitable for purely browser based apps since it requires to be logged in into a container dashboard (see Figure 1). The later is based on HTTP REST and allows building applications that are independent from the dashboard, although it is generally more complicated to implement.

---


\(^3\) [http://healthit.hhs.gov/portal/server.pt/community/strategic_health_it_advanced_research_projects/1436/home/16979](http://healthit.hhs.gov/portal/server.pt/community/strategic_health_it_advanced_research_projects/1436/home/16979)

\(^4\) [http://wiki.chip.org/smart-project/](http://wiki.chip.org/smart-project/)
At the current stage of the SMArt project the container of main interest is the reference container provided by the project for development purposes. However, there are several other projects that are working on container implementations at this time, so that we are expecting more sophisticated examples soon (e.g., i2b2\(^5\) or Indivio PHR\(^6\)).

As part of our project, we obtained all components of the reference container (SMArt Server, User Interface Server, and Demo Application Server) and built a local test environment. This has proven to be a challenging task at time, since the Python based implementation is under currently still under constant development and requires a fair amount of tuning during an installation. The code and installation instructions can be found in the project’s source repository\(^7\), an important additional source of information at this stage is the developer

\(^5\) https://community.i2b2.org/wiki/display/SMArt/

\(^6\) http://indivohealth.org/

\(^7\) https://github.com/chb/
discussion group[^8]. For pure development, the hosted development sandbox[^9] of the project can be used as an alternative to a local container installation.

**SMArt Apps**

At its core, each SMArt application is a normal web application. It can be either embedded as a frame within a SMArt container’s dashboard interface (see Figure 1), with access to the SMART API for interacting with the container’s health data, or call the API through HTTP REST without using the dashboard. Typical examples of SMArt apps are risk assessment and decision support tools, medication and prescription lists, reminders, or data collection and monitoring interfaces.

A fundamental idea of the SMArt paradigm is to keep applications 100% replaceable. For example, if a physician is using a SMArt app to review his patient’s medication list and at some point decides to get a more sophisticated list (e.g., a list that provides common packaging images for each medication to allow easier confirmation in discussions with patients), then the complexity of moving to this list should be limited to the action of replacing the old one on the dashboard. SMArt apps are required to be independent from the underlying data by being fully conform to the SMArt API and the SMArt data model.

**Challenge**

In order to promote the platform, the SMArt project launched a $5000 dollar SMArt app challenge. Because of the early stage project, the rules of the challenge turned out being limiting to some extent. One major drawback was that developers were not allowed to use the writing capabilities of the API and were restricted to use cases that called for a write only behavior.

In this context we decided on implementing the **SMArtSync** app, a medication and supplement reconciliation tool that allows patient to extract their medication data from SMArt containers, review it, and merge it into PHRs such as Google Health.

The remainder of this report covers details of the **SMArtSync** app and its interaction with a PHR.

[^8]: [http://groups.google.com/group/smart-app-developers](http://groups.google.com/group/smart-app-developers)

[^9]: [http://sandbox.smartplatforms.org](http://sandbox.smartplatforms.org)
**SMArtSync**

Medication reconciliation is a universally recognized challenge in health care. Patients are able to receive care from different sources. This brings several conflicts, especially with medication prescriptions. Duplicate, conflicting and unsuccessful medication could be prescribed to patients without previous knowledge of their previous prescription history. Another dimension is added to the problem by possible interactions with medical supplements (e.g., herbal extracts) that are commonly used by patients, but are rarely documented in a way accessible to physicians. Thus, it remains an important goal to solve the medication reconciliation problem.

**SMArtSync** is an application that aims to solve that problem, putting the patient at the center of the responsibility. In a general description, **SMArtSync** gives the patient the capabilities of compiling a full list of medications available in the official electronic medical record and the patient’s personal health record. This approach gives the patient the ability of not only adding medications, but adding supplements that could have been bought over the counter without any prescription. When the patient has a list of all the medications s/he has taken, he could show it to the health provider for the most up to date information.

The way **SMArtSync** achieves this goal is by providing a “parallel” view of the patient’s medication in the EMR, with SMArt as the front-end, and Google Health as the PHR solution. **SMArtSync** also works as another user interface to Google Health if the patient desires to use it as so. It supports the addition and removal of medications and supplements.

![SMArtSync Medication Flow Diagram](image-url)
Meds in EMR

This view allows the patient to see the medications stored in the official SMArt enabled EMR. These medications have information such as name, code, instructions, start and end date, supplied date and the amount of dates the medication was prescribed for. This is the default view when the patient requests access to the application.

Figure 3: Example EMR medication list for the patient Amy Young
Meds and Supplements in Google Health

This view allows the patient to see the medications stored in their PHR provided by Google Health. These medications have information such as name, code, instructions, start and end date, supplied date and the amount of dates the medication was prescribed for. To access this view, patients must have a working Google account. In here they can add and delete medications from the PHR.
After they are granted access to Google Health, the patients are able to see all their stored medications in Google Health. In addition, new buttons appear in the Meds in EMR tab, which allows the patient to transfer medications from the EMR to Google Health.
Google Health Data Synchronization

In this section we provide implementation detail of the way SMArtSync interfaces to Google Health. The communication with Google Health was realized through a separate server application. We hope that this module based approach will simplify future implementations that might require support for additional PHRs.

Generally, synchronizing data includes three kinds of requirements: add, update, and delete. Since Google Health requires authentication for accessing a user’s health records, a supporting login operation also becomes a requirement. In our implementation, users send requests to the Java server via HTTP requests (AJAX), which is responsible for the interactions with Google Health. The four operations are described as follows:
1. Login:
A login request including Google email and password is sent to Google Health for authentication. If authentication passes, a Google Health connection will be established and the user’s medication list will be returned. The medication list is stored in the form of a HashSet in the server’s memory at run time. It will be used for adding, updating or deleting medication records in the future. The process is illustrated in Figure 8:

![Figure 8: Login Operation](image)

2. Add medication record:
When user requests to add a new medication record, the server will first look up the medication list (HashSet) stored in its memory and check if this record is already in the list. If it already exists, an update operation will be performed. Otherwise, the new record will be inserted into Google Health and the medication list. This process is illustrated in Figure 9:
3. Update medication record:
When user requests to update a medication record, the program will first look up the medication list (Hashset) stored in the server memory and check if this record is already in the list. If the record does exist, it will be updated in both Google Health and the Hashset. Otherwise, this medication record will be inserted as a new record into Google Health and the Hashset.

This process is illustrated as Figure 10:
4. Delete medication record:
When user requests to delete a medication record, the program will first look up the medication list (Hashset) stored in the server memory and check if this record is already in the list. If the record does exist, it will be deleted in both Google Health and the Hashset. Otherwise, the program returns without doing anything. This process is illustrated in Figure 11:

Figure 11: Delete Medication Record Operation