Evaluation of Web 2.0 Technologies for Developing Online Telehealth Systems

Jaspaljeet Singh Dhillon, Czarina Ramos, Burkhard C. Wünsche, Christof Lutteroth
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
{jran055,cram021}@aucklanduni.ac.nz,{burkhard,lutteroth}@cs.auckland.ac.nz

Abstract

Telehealth and telecare applications are a promising technology for improving the quality of care while using healthcare resources more effectively. Major obstacles to a more widespread use are the high initial costs and a vendor specific design, which makes it difficult and expensive to add new functionalities. The Internet offers an opportunity to make telehealth applications more accessible, while also adding social aspects and the opportunity for third-party developers to add content. A preliminary user study confirmed that elderly are interested in such an application, and provided guidelines for the user interface design and required functionalities. In this paper, we evaluate technologies for developing online telehealth platforms, and present a first prototype which is extendable and has social networking capabilities. Our results show that a combination of open web standards such as OpenSocial and a CMS such as Drupal represents a suitable design. We illustrate the capabilities of our design and prototype by developing a memory game, which can be submitted by third party developers, similar to a Facebook application, and which utilises the social context of our telehealth application.

Keywords: Web 2.0 technologies, social networks, health informatics, telehealthcare.

1 Introduction

Telehealth and telecare systems are a promising approach to use healthcare resources more effectively. However, usage is constrained by high initial costs and a design often centered on the requirements of the clinical user, healthcare provider, and the equipment vendor. Most existing systems cannot be extended by third parties, require extra costs to add new functionalities, are designed to manage diseases rather than prevent them, and do not address the social and psychological needs of the patient.

A suitable concept to overcome these shortcomings is Web 2.0, commonly known as "the web as a platform" (O'Reilly 2005). The term refers to web applications and services that facilitate interactive information sharing, rich user experience, dynamic content, and user-centred design. The open-ended nature, interconnectivity and large user community of popular social networks such as Facebook, MySpace and Orkut has enabled third-party developers to offer functionalities and content, which would be difficult to achieve with a stand-alone application.

The use of Web 2.0 in healthcare is rapidly evolving as more applications and services targeting health professionals and patients are being developed. With that trend, the term Health 2.0 is becoming popular with health management systems such as PatientsLikeMe, CureTogether, SugarStats and MyFitnesspal. Existing Health 2.0 applications provide useful functionalities such as diet and exercise monitoring, and formation of support groups. However, they do not offer a comprehensive suite of functionalities and do not replace traditional telehealth platforms (Dhillion et al. 2011).

The above analysis shows that current telehealth technologies represent two extremes. On the one side are very expensive vendor specific telehealth platforms. These are targeting patients with chronic diseases (e.g. diabetes, asthma, COPD, heart failure) requiring a high level of health support, which justifies the high costs of such systems. The systems are doctor centric and in many cases the patient is a passive provider of monitoring data and recipient of doctor advice, with few opportunities to participate in treatment and intervention plans. On the other side are Web 2.0 applications, which are mostly free and generally focus on one functionality only. These applications usually do not involve clinical users, and allow the patient to take control of their health, e.g. by devising diet plans or sharing with fellow patients experiences such as side effects of medications.

In this paper, we analyse different technologies for implementing web-based extendable telehealth systems with social networking capabilities. Specifically, we investigate the potential of popular social networking APIs and web development tools against the requirements for a general affordable telehealth system. We illustrate the results in the design of Healthcare4Life, our Health 2.0 platform, and a memory game application developed for it. We discuss the implementation process and relevant issues which will benefit the development of online telehealth systems in general.

Section 2 summarises important requirements for a general affordable telehealth system. Section 3 and Section 4 present an evaluation of popular Application Programming Interfaces (APIs) of social networking and web development tools for the development of telehealth

systems. Section 5 discusses the design and implementation of Healthcare4Life, a working prototype whose architecture and design is motivated by the requirements specified in Section 2. We also discuss the design and development of an example application (a memory game), and how it is integrated into Healthcare4Life. Section 6 describes the results of implementing the technologies, and Section 7 concludes the paper and gives an outlook on future work.

2 Requirement Analysis

Current telehealth systems do not take into account the importance of patients' social needs. In previous work we showed that social interactions are essential for patients, especially the elderly, to improve their quality of life and to overcome social isolation (Singh et al. 2010b). Social networks can help users to get in touch with their family, make new friends, and discuss medical concerns with peers and support groups. Furthermore, social networks also help with motivating the patient, e.g. by achieving family support, or by patients performing monitoring task and exercises together via a video link or in a virtual environment. Therefore, we suggest that novel telehealth systems should include social networking capabilities.

Existing telehealth systems are predominantly standalone applications, which come with a specific functionality, e.g. a device for measuring vital signs such as blood pressure, weight, pulse, and blood glucose levels. These applications cannot be extended by third parties and require the users to pay more to add new functionalities. In order to intervene early in the development of serious diseases, a larger proportion of the population needs access to telehealthcare services, and a wider range of functionalities must be provided. Examples are support for diet programs or physiotherapy exercises (Dhillon et al. 2011). This also reduces the risk that patients becoming bored with a limited range of content and functionalities. Moreover, for elderly people, it is often difficult to work with many different systems, and a single integrated user interface is necessary. Hence, telehealth systems should be designed with a plug-in architecture to enable third-party application developers to add content or health applications easily.

Current systems are mostly designed for patients to transmit health parameters to clinicians, i.e. do not encourage active participation of users in their healthcare. A general telehealth system should enable the patient to take control of their health. This requires tracking of health parameters (e.g. vital signs, exercise performance), feedback and alerts, and graphical representations to easily track progress and compare it with goals (Singh et al. 2010b).. Lee et al. (2011) and Fischer et al. (2011) have shown that the use of patient specific visualizations can alter patient behavior and support rehabilitation.

One of the core requirements of telehealth systems is user-friendliness. The resulting requirements include the use of large font sizes, an easy-to-follow linear structure, and the use of a horizontal menu at the top of the screen to make it easy to identify and choose key functionalities (Dhillon et al. 2011).

It is essential that monitoring data can be shared between different applications. This will avoid the need for users to re-enter data (e.g. patient parameters), and it allows implementation of more powerful functionalities (e.g. total calories burned or overall fitness). Such features will also help users to keep track of the total amount of time spent in performing the health related activities, and data can more easily be compared with other users to increase motivation.

Another important requirement of telehealth systems is the ability to integrate with consumer level HCI devices, e.g. iPhones and Wii remotes. These devices contain motion sensors such as accelerometers that can measure positions, velocity and direction vectors, which can be leveraged in creating rich applications, e.g. pedometers, fall detection, or guided rehabilitation activities to improve the condition of the user (Dhillion et al. 2011).

3 Evaluation of Social Networking APIs

Social networking APIs enable developers to integrate social features into their systems. The Facebook and OpenSocial APIs are the two most popular examples. In this section, we describe and compare these APIs for the development of telehealth systems based on the requirements discussed in Section 2.

3.1 OpenSocial

OpenSocial provides a set of common APIs for developing web-based solutions, with a focus on social applications. It is currently managed by the non-profit OpenSocial Foundation, is developed by Google along with MySpace, and is supported by a number of other social networks and well known software vendors such as IBM and SAP. The principle idea of OpenSocial is to make applications widely available to more users by enabling application developers to deploy the same application across multiple platforms with no or minimum modification. Nevertheless, developers are increasingly exploring OpenSocial for other development needs, moving from traditional social networking concepts to enterprise-level software.

OpenSocial allows the development of an open platform, also known as an OpenSocial container, where third-party developers can contribute applications written using the OpenSocial API. OpenSocial applications share the same structure as Google gadgets, therefore are also known as OpenSocial gadgets. These gadgets are actually XML documents containing HTML and JavaScript code along with metadata. There are two types of gadgets that can be built using OpenSocial: gadgets that live within the hosting container, and gadgets that rely on an external server. The latter is widely used in realising open platforms, where developers integrate XML specifications located on their own external web servers with the hosting OpenSocial container.

The contents of a gadget can be displayed in the different views supported by the container, e.g. profile, canvas, home and preview (Häsel 2011). Gadgets can be specified to switch between these views to enable the users to interact with applications in different sizes and layouts. Most containers support the canvas view, which displays the rendered gadget by itself in a full screen page within the container.

MySpace, Hi5 and Orkut are some of the popular OpenSocial containers that take advantage of the services provided by the API. Examples are methods to access information about people, friends, and data, within the context of a container.

To become an OpenSocial container that can render remote or embedded gadgets and support social networking features, a system must comply with both the Core Gadget Container Specification and Social Gadget Specification (OpenSocial 2011a). Developers can make use of Apache Shindig, a reference implementation of the OpenSocial standards, to host OpenSocial applications with little effort. It provides the code to render gadgets and proxy requests, as well as handle REST and RPC requests. Communications between the Apache Shindig and the application take place via standardised AJAX requests, defined in the OpenSocial JavaScript API (OpenSocial 2011d). Apache Shindig is currently written in both Java and PHP. The hosting process of the container is made possible through its four components: Gadget Container JavaScript, Gadget Rendering Server, OpenSocial Container JavaScript and OpenSocial Data Server (Shindig 2010). Apache Shindig also provides a variety of security level options to secure requests and responses, i.e. to enable developers to make applications more secure. It uses Shindig user security tokens, two and three way handshakes, OAuth, and various encryption technologies.

Any HTML page can be fetched and displayed in an OpenSocial container using a gadget mechanism called Proxied Content. This implies that developers can specify the Uniform Resource Identifier (URI) of any existing online application in the XML specification to turn it into a gadget that can be rendered by the container (OpenSocial 2011b). However, the outcome of wrapping existing applications in OpenSocial can be less rewarding, as these applications may not be designed to provide social interaction, unless the developer made significant use of the user's social context including friends lists and activity streams (Hinchcliffe 2011).

Although OpenSocial was not ready for productive use when it was launched in November 2007 (Schonfeld 2007), it is rapidly evolving with more improvements and significant features (OpenSocial 2011a). Recently, the OpenSocial Foundation has launched OpenSocial 2.0 (OpenSocial 2011c), which includes features such as embedded experiences, activity streams standardisation, support for mobile devices, OAuth 2.0 and OpenSearch support (Hinchcliffe 2011).

3.2 Facebook

Facebook is the most prominent social network and has nearly 700 million users worldwide (Eldon 2011). Similar to OpenSocial, the API allows applications to utilise profile, friend, photo, and event data to add social context. It also allows the applications to publish activities to the news feed and profile pages of Facebook. Increasingly, it is used by people and company sites as an identity provider with its support for OAuth 2.0. This avoids the need to register or create a new user account on each site individually. The large user base of the Facebook attracts many third-party developers who build new products and services on this platform.

The API supports the RESTful API and the Graph API (Facebook 2011). The Facebook platform is based on a URL-addressable, RESTlike server API, i.e. it assigns unique IDs to each social object in the system, which can be invoked by a URL. OpenSocial gadgets are rendered by the surrounding container (e.g. Apache Shindig) and can communicate with their backend servers via JavaScript calls, whereas Facebook applications rest entirely on their developers' web server. In contrast to OpenSocial platforms, Facebook restricts developers to proprietary language requirements such as FBML (an evolved subset of HTML), FQL (an SQL-style interface for querying social data), and FBJS (a solution to enable developers to use JavaScript in their FBML applications). The engineering team of Facebook has released and maintains open source SDKs for Android, C#, iPhone, JavaScript, PHP, and Python (Facebook 2011).

The use of the Facebook API for telehealth systems makes it possible to access millions of users of this social network, including family and friends of a patient. According to Norval et al. (2011), this makes it easier to connect people known to a patient to provide care or social support. However, elderly users remain a clear minority and in 2010 only 2% of Facebook users were in the 65+ age bracket (Socialbakers 2011).

Despite the huge success of Facebook, it has been reported recently that traffic is dropping (Eldon 2011) and significant challenges exist to archive the users' personal data (McCown 2009). A growing number of Facebook users are switching over to Google+ due mainly to its apparent integration with a variety of Google services (Sullivan 2011). The latest statistics from SocialBakers (2011) show that the three top categories of applications in Facebook are Games, Entertainment and Lifestyle. Health applications (a sub-category of Lifestyle) has just over 1% of the total available applications. Therefore, although Facebook is known to be the most popular social networking site, it is not necessarily an appropriate platform for health related applications.

3.3 Discussion

Section 2 showed that two major requirements of telehealth systems are to make the systems extendable by third parties and to include social networking capabilities. There are two general approaches to realise such features: 1) creating a new API from scratch and sharing it with health application developers, and 2) deploying and adapting existing tools that support development of an open platform with social interactions. The former is more difficult because it takes time for an API to mature and to be accepted as a standard for developers. When leveraging existing Web 2.0 technologies to create open-ended systems (Dhillion et al. 2011), developers need to make a well-informed choice about the API used.

Social networking APIs are the core technologies needed to realise the open platform and social aspects of such a system. After reviewing the current social networking APIs, it is clear that both OpenSocial and Facebook allow people to keep up with friends, upload an unlimited number of photos, share links and videos, and

learn more about the people they meet. Both APIs support the development of third-party applications. However, the APIs were created with different objectives. Table 1 shows a comparison between the OpenSocialAPI and the Facebook API.

OpenSocial	Facebook
A specification	A social network
Standard API for social applications to run on multiple social networks	Single network API
Open with no proprietary regulations	Strict proprietary regulations
Applications hosted are commonly client-side JavaScript-oriented (gadgets) as well as server-oriented	Applications hosted are all server-oriented
Allows portability of an application into various OpenSocial containers	Applications can only run within the Facebook platform
Uses common languages (e.g. HTML, XML and JavaScript)	Uses proprietary languages (e.g. FBML, FQL and FBJS)
Full control over the social network functionalities and user policies	Little control over the social network functionalities and user policies

Table 1: OpenSocial versus Facebook

OpenSocial is not a technology but a specification. By following the OpenSocial specification any system can be turned into an open platform, which can interact with other applications in a standardised way. The platform and other applications will have a common set of interfaces and processes in order to communicate seamlessly. OpenSocial's social API can be leveraged to incorporate social networking features into a new system. In contrast to OpenSocial, Facebook is a social network and does not provide an open platform. It uses a plug-in architecture to enable developers to create applications, which can only run within the Facebook platform.

Developers using the Facebook API have little control over the social network functionalities and user policies (Norval et al. 2011). Unlike Facebook, OpenSocial also allows the development of web-based telehealth systems without constraining the user with proprietary regulations. Developers will have full control over their system and the freedom to integrate it with other OpenSocial containers. The ability to run applications on various containers will encourage potential developers to contribute health applications. However, OpenSocial allows this to happen only if the applications are programmed to be generic, and do not use their own proprietary API (Häsel 2011).

The reference implementation of OpenSocial, Apache Shindig, enables a telehealth system to be transformed into an OpenSocial container. Development of gadget applications is easy and attractive, since developers are not required to learn new programming languages and specific platform traits associated with its proprietary mechanism (McIlrath 2010). Instead, common languages such as HTML and JavaScript can be used. Developers can create application using a variety of technologies, including CSS, OpenSocial Templates, Flash, PHP, Python, Java, Perl, .NET, and Ruby.

The ability to easily embed existing health applications into an OpenSocial container is a great advantage for telehealth systems. Although these applications may have limited social features, developers will be able to embed proper existing health applications into the telehealth system. In addition, this will allow users, especially the elderly, to interact with existing health related applications within the same interface.

The idea of leveraging OpenSocial for telehealth systems was initiated by Weitzel et al. (2009), who described the use of this Web 2.0 technology in providing extended care networks for chronic disease management and elderly care. Furthermore, Weitzel et al. (2010) have discussed a Web 2.0 model for patient-centred health informatics applications. The suggested model uses open technologies such as OpenSocial, REST, and Open Authentication.

Based on the reviews and analysis, the OpenSocial APIs meet the requirements to develop online telehealth systems which are extendable and contain social aspect.

4 Web Development Tools

Web Development Tools are necessary to construct the web-based systems and its functionalities. The two common approaches are Content Management Systems (CMSs) and Web Development Frameworks (WDFs). In this section, we will highlight the strengths and weaknesses in their ability to design, develop and maintain web-based telehealth systems.

4.1 Content Management Systems

CMSs support developers with setting up rich and dynamic websites. With a CMS, the content is stored in a database and the templates, styles or themes that determine how the content is presented are maintained separately. Most CMSs leverage the power of Cascading Style Sheets (CSS) to easily update or make changes to the look and feel of websites. The main advantage of employing a CMS in developing a web-based system is the variety of ready-made modules, which can be directly used or adopted to add desired features to the website. CMSs can be integrated with existing social networking APIs. For example, Drupal has a module to integrate the OpenSocial API. Most CMSs use popular programming language such as PHP to enable developers to create their customised modules for specific features of their site. The three major open-source CMSs are Drupal, Joomla and WordPress (water&stone and CMSWire 2009).

Drupal is one of the most popular and powerful CMS available to develop dynamic state-of-the-art Web 2.0 sites. Drupal is supported by a large and active community of developers, and offers a large number of open source extensions, modules, and themes. It is based on a customisable framework which enables its site visitors to contribute content. Provided functions go beyond those of a CMS, as it also acts as a framework for developing web applications and is used in a wide variety of deployments. Drupal is increasingly used for developing social networking sites (Purham 2010) and healthcare systems, including sites connecting patients to health services (Drupal 2011a).

Joomla is another strong alternative for rich web development. It is easy to use, but most of the customisations required by the user are built around paid plug-ins and themes. It lacks important features such as a powerful blogging engine, nested categories, a built-in download manager/document repository, Content Construction Kit (CCK) abilities (functionality to easily move content around), and many other features already found in Drupal. WordPress has a strong focus on blogging, although a large number of open source plug-ins are available to extend its functionality. It is ideal for fairly simple blog-style web sites, but is not suitable for more complex site requirements.

4.2 Web Development Frameworks

Web Development Frameworks (WDFs) support developers with building websites, web applications and web services. There are many frameworks available for web development, written in various programming languages, with varying technical and conceptual differences (Singh 2010). For example, Yii, CodeIgniter, Zend, CakePHP and Symfony are just a few of the popular ones from the vast selection of PHP frameworks available to code web-based projects. Although each framework is different, they generally provide a variety of useful features. These frameworks provide functionalities that are common to most web applications, e.g. database access, sessions management and templating systems. WDFs help in providing a basic structure to develop web-based systems, which enables developers to reduce repetition and write code in a shorter amount of time. For instance, a framework enables developers to avoid the need to re-code the same features for each web application they create.

Most WDFs are based on the Model View Controller (MVC) architecture. The MVC implements a "separation of concerns", i.e. distinct features without overlapping functionality. Examples include isolation of the application logic from the user interface and separation of database access code from the application logic. Separation of tasks, such as web programming from user interface design, allows a development team to focus on specific objectives and use their individual strengths (DocForge 2010). With MVC developers can focus and work on individual elements. Hence, the concept of MVC helps to break the development process of an application into manageable tasks.

4.3 Discussion

Using web development tools can considerable reduce development times, which is essential for complex incremental systems. The choice of using CMSs or WDFs depends on the complexity, requirements and duration of the telehealth project. Table 2 shows a comparison between CMS and WDF. Drupal offers more functionality over other available CMSs and is easier to use, but it has a steeper learning curve.

Although Drupal helps developers to create specific modules for features lacking in their system, it encompasses nearly 8500 contributed modules (Drupal 2011b), including modules to integrate HCI devices such as webcams (Drupal 2011d). Therefore it is likely that the needed functionality is already available. As it was stated earlier, in order to host OpenSocial applications, Apache Shindig needs to be installed to render these applications. Drupal is preferred over other CMSs and WDFs because it includes the OpenSocial Shindig-Integrator, which can be used to integrate the Apache Shindig container with any Drupal-based system (Drupal 2011c). Hence, Drupal can be used to construct a full-featured and extensible web-based telehealth system that can host OpenSocial health applications.

Content Management Systems	**Web Development Frameworks**
Low learning curve	High learning curve
Easier system updates	Longer update and upgrade times
Planning is useful	Prior proper planning is essential
More consistent and controlled outcome	Lower quality control over the outcome
Provides back-end support such as modules and themes	Build from scratch using available classes and libraries
Suitable for small to medium projects, including system prototypes	Suitable for large and complex projects
Flexibility of adding customised modules	Unlimited flexibility
Suitable for projects with a small development team	Suitable for projects with a large development team
Reduce development times	Typically requires longer development time

Table 2: Content Management Systems versus Web Development Frameworks

As mentioned earlier, Drupal makes it easier to update a system. This is important for a telehealth system, since patient needs, available knowledge, and required interactions with other health providers can change. The module and theme feature help to set up web interface components. In addition, there are lots of user interface templates available to improve the look and feel of a system. Moreover, Drupal has a large and active user community, which is helpful during the development process.

In comparison to CMSs, WDFs frameworks are suitable for large and complex projects which require maximum flexibility. CMSs provides the user with back-end support (such as modules and themes) to develop and manage a website (front-end). By contrast, users deploying a WDFs framework have to build their sites from scratch by using the readily available classes and libraries. The initial learning curve for some of the WDFs can be quite steep and it requires the user to have solid knowledge of Object Oriented Programming (OOP). Other known shortcomings of employing a WDF such as a PHP framework include: 1) lower quality control over the outcome, 2) longer update and upgrade times, and 3) a stronger need for proper planning (Cheng 2009). Using CMSs makes system updates easier than when using WDFs. This is especially important if future content updates or changes will be done by non-technical users.

5 Design and Implementation

Section 2 summarised requirements for developing a web-based telehealth system, which addresses the

shortcomings of existing telehealth applications. In this section, we explain how we used Web 2.0 technologies in realising a working prototype of a web-based telehealth system called Healthcare4Life.

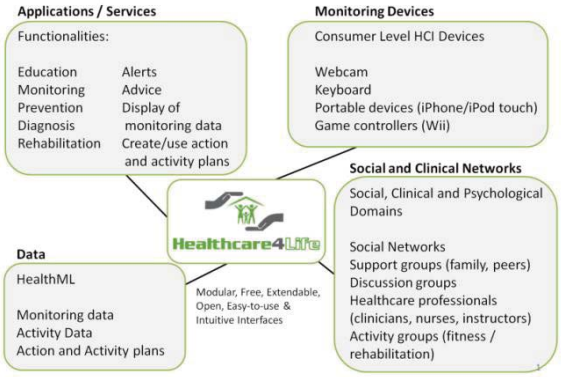

Figure 1: Framework for *Healthcare4Life* (Singh, Wünsche and Lutteroth, 2010a)

In previous work, we presented a theoretical framework for Healthcare4Life (Figure 1), a novel web-based telehealth system that combines the power of social media with telehealth systems to enable patients to take charge of their own health (Singh et al. 2010a). Its goal is to transform the restricted nature of traditional telehealth systems by making them widely available, affordable and extendable. The framework has an open OpenSocial-like architecture, which allows third-party providers to add new content and functionalities. This allows users to choose new monitoring and exercise tools if they get tired of existing ones or develop new needs. It also makes it possible to incorporate emerging technologies such as new HCI devices. Functionalities include applications for monitoring, rehabilitation, education, and advice in the form of, e.g. serious games, interactive web pages and expert systems.

Our solution does not put any constraint on the physical location of users. Since we do not provide medical advice, we are not limited by government policies on this. System feedback will contain general health information and advice to contact a user's healthcare provider if unusual patterns in the monitoring data are detected.

Patient privacy is important and users are encouraged not to share clinical information. However, like with existing social networking and patient support websites it is ultimately up to the user to decide what information to store and share. For example, in order to find peers suffering from the same disease it is necessary to specify this information, and other members of such a patient group will hence implicitly gain this information.

Our technology evaluation has shown that OpenSocial and Drupal are suitable technologies to realise a web-based telehealth platform. Both technologies are open-sourced with a large and active community, and access to the resources and support for implementing our system. We therefore decided to use these two technologies for our system. The ideas presented by Weitzel et al. (2009 & 2010), along with our review on current social networking APIs, affirms the value of leveraging OpenSocial in the development of Healthcare4Life. Since we are using the PHP version of Apache Shindig and Drupal, which is also written in PHP, we adopted PHP as our main programming language.

5.1 Container

The resulting system must support both application users and application developers. Both groups are presented with distinctive functionalities based on their role in the system. The application users or patients are presented with a clear horizontal menu at the top, with six icons and descriptive text: Home, Applications, Profile, Mail, Friends and Search (see Figure 3). Table 3 describes the main purpose of each page and respective Drupal modules used in implementing the features. The core functionality provided to application developers is to embed their gadgets applications into Healthcare4Life. All applications will be listed in the applications directory of the system for users to interact with them.

Page	Purpose	Drupal Module
Home	To share their status, view and comment status of friends within the network.	*Facebook-style Statuses*
Applications	All health applications added by developers will be listed at this page, as icons. Users are required to click on the respective icon to interact with a health application in canvas view.	*Shindig-Integrator*
Profile	To view and edit profile information. It will also consist of summary of latest activities such recent health applications used by the used.	*Content-profile and Flag*
Mail	To send a mail to friends within the network.	*Mail*
Friends	To access friends' profile page.	*Flag*
Search	To find new friends within the network.	*Search*

Table 3: Functionalities provided to application users

We started the implementation process by designing a customised theme for Healthcare4Life using HTML and CSS, which was imported into Drupal. Although Drupal comes with lots of design templates, we opted for a new theme based on the interface design requirements specified by elderly people (Dhillon et al. 2011). We then adapted existing Drupal modules for many of the functionalities of Healthcare4Life. For instance, we employed the Flag module for storing data, e.g. information of friends and activities, and Webforms module for developer and user registration. We also created customised modules for features not supported by the existing modules. For instance, Drupal does not support different types of registration for developers and normal users, which is required for Healthcare4Life. Social networking functionalities currently implemented include the ability to create profile pages, send email to others users, add friends and search for friends using specific keywords (e.g. username, age and hobbies).

Upon implementing the basic functionalities, we integrated OpenSocial with Drupal to transform Healthcare4Life into an open platform for third-party developers. Initially, we installed Apache Shindig within the Healthcare4Life environment and then integrated it with Drupal. Figure 2 depicts the architecture of our system.

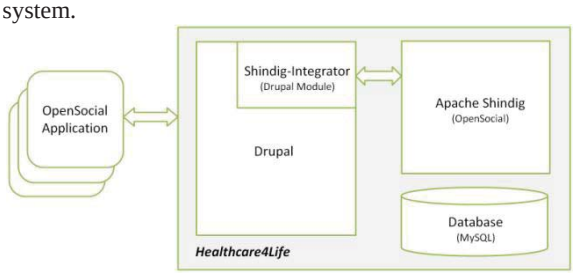

Figure 2: Architecture of *Healthcare4Life*

In order to make Healthcare4Life an OpenSocial compliant platform, the Shindig-Integrator module was used. This module originally used the User Relationships module for a user's friends' data, and the Profile module of Drupal to store a user's profile (Drupal 2011c). In Healthcare4Life, we have used different modules to achieve the same. Hence, to integrate the Shindig-Integrator with our platform, we had to adjust the code to use the Flag and Content-profile modules for user friends and profile, respectively. The Shindig-Integrator has a class which talks to the database for retrieving a user profile or user friends from the database based on the user ID. We changed the code, which was referring to the Profile and User-relation module's database, and made them retrieve data from the Flag and Content-profile module database.

With OpenSocial, developers are able to add applications residing on their own web servers by simply specifying their URL, i.e. the location of the XML file of the application, in our system. This XML specification of an application will be rendered by Apache Shindig and integrated into our system. The gadget applications added here are automatically listed in the applications directory of the site. We have tested this with existing applications from Labpixies (www.labpixies.com). Apart from specifying the URL of their application, application developers need to select a suitable category for their application. This helps users with selecting applications, e.g. for monitoring, education and rehabilitation.

We have also connected Healthcare4Life with Shindig's OpenSocial Service Provider Interface (SPI) to allow gadget applications to access our site's social data. The SPI implements: 1) retrieving people information, 2) storing and retrieving activities, 3) storing and retrieving persistent data, and 4) sending messages.

5.2 Sample Application

We have designed and developed a simple memory game in order to demonstrate the capabilities of OpenSocial's client side API. This involved transforming an existing JavaScript application into an OpenSocial application that can access the social context of our Healthcare4Life container. Applications that are appropriate for the Healthcare4Life platform must be beneficial to the user's health. As mentioned earlier, applications plugged into the platform are grouped into specific categories based on their purpose. Rehabilitation is one of the application categories available in Healthcare4Life and will contain applications that aim to improve cognitive or physical functions.

The design and implementation of the new OpenSocial memory game was heavily based on a study by Ijsselsteijn et al. (2007) investigating the needs and motivation of elderly gamers. The original JavaScript memory game did not have social features that the elderly might find interesting, such as personalised, challenging and collaborative game play. In addition to this, the game design did not consider the special requirements of the elderly, e.g. readable fonts, larger images and familiar terminology. For testing purposes, our design aimed to make the new memory game challenging and personal to the user by allowing the application to access the personal data of the user's friends in the Healthcare4Life network.

The original JavaScript memory game is able to run in the OpenSocial container using the Proxied Content technique mentioned earlier. However, this does not provide the memory game with any interaction to the user's social data. The original game simply used cartoon images as objects of the game, which is monotonous and leads to the elderly being less motivated to use the game again. It became obvious that educating users on the benefit of the game to their health is not enough to motivate them to keep playing the game. To make the game more interesting, we have added a new "Card Deck" called "Friends". Terms such as "Card Deck", "Friends" and "Cartoon Images" are used because they are common terminology familiar to the elderly. Choosing the "Friends" game mode allows users to play using their friends' images as objects of the game. This makes the game more personal to the user, giving it a social aspect.

Figure 3 illustrates the memory game running in Healthcare4Life. Developers are free to use their own web servers in hosting their applications, however, we chose to host our OpenSocial application in the iGoogle gadget server and embed it in our platform under the rehabilitation category. The new memory game is an XML document with HTML and JavaScript bodies, much like the architecture of a Google gadget. The main difference is that the game uses OpenSocial to gain access to the social data of the Healthcare4Life container. Developers can use their favourite editor to create their OpenSocial gadget. When developing our memory game, we used the Eclipse IDE, as it provides a readily available OpenSocial plug-in for ease of testing and deployment of applications.

Our game consists of multiple functions which are mixed with calls to the OpenSocial API. Examples are a function for retrieving user's information, a function that randomises and loads the user's friend's thumbnail images into the game, and a function that saves the user's score into the network's persistent data storage. The OpenSocial REST API allows the application to retrieve information about the user by calling the HTTP GET request to our server, in this case, the list of friends they have in the Healthcare4Life network. Healthcare4Life allows users to save small amounts of data about a

particular user, such as the scores achieved in the memory game application, using the storage mechanism of OpenSocial called Persistence API. Likewise, with this mechanism, each of the players is able to load their scores from the last time they used the application. These persistent data are key/value pairs which only accept a string format. Non-string data values such as the scores in our memory game need to be converted into a string format before being stored and then parsed back to its original format. Players are able to compare their achievements with other players by posting on their friends' walls, which effectively adds more social aspects to the game.

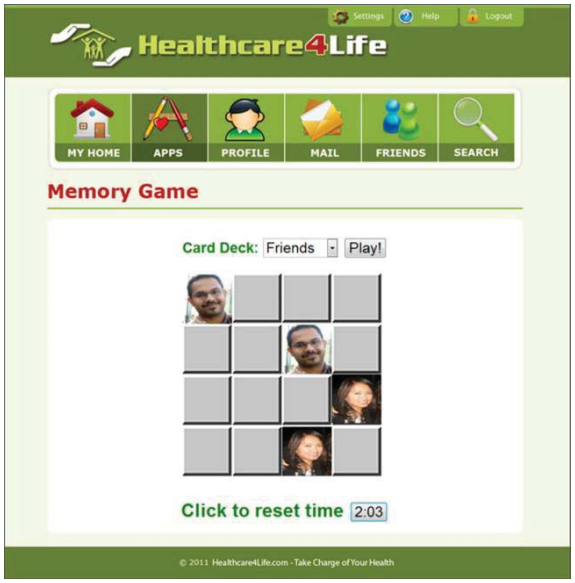

Figure 3: Memory game in *Healthcare4Life*

Healthcare4Life was able to render the game smoothly in canvas view and allowed access to its social data. Using standard HTML, JavaScript and CSS styling, we were able to change the old memory game into a more usable game with larger elements, a more elaborate interface, and more importantly, more interesting game play. The memory game was also tested in other OpenSocial containers, Orkut and iGoogle, which ran the game without requiring any complex modifications. There are a number of other possible social aspects that can be added to the memory game using the OpenSocial API, such as collaborative game play and scoreboards.

6 Results

Our evaluations of functionalities, and the subsequent design of a working prototype and a sample application, showed that OpenSocial and Drupal integrate well and can be used to develop extendable online telehealth systems with social capabilities. Based on our analysis, implementation experience and understanding, almost all of the requirements described in Section 2 can be achieved by leveraging these technologies. Below, we discuss the strengths and related issues in employing OpenSocial and Drupal.

6.1 OpenSocial

The availability of Apache Shindig, the reference implementation of OpenSocial, has helped to turn Healthcare4Life into an OpenSocial container that can host health applications created by external developers. Theoretically, OpenSocial compliant websites can be programmed using any programming language as long as the OpenSocial specifications are satisfied. In practice, however, the Apache Shindig will be used for most projects, which supports only Java and PHP.

It is fairly easy to develop gadget applications since common languages such as HTML and JavaScript are used. There is a lot of documentation available to get started with the development of a gadget application. However, it is difficult to find tutorials to develop rich and complex gadget applications, such as a multi-player memory game. Furthermore, applications must implement a valid version that specifies the features that we want the hosting container to interpret (OpenSocial 2011b). Issues with compliance arise when rendering applications implementing the 0.9.x version of the API, as the Shindig-Integrator of Drupal is only available in version 1.0. Also, it is found that designing a gadget too specific to a container requires more modification when running in other containers. Hence, the idea of running the same application across multiple OpenSocial containers is possible, but it is not as simple as documented.

OpenSocial provides various libraries to support a variety of applications and technologies. For instance, it provides a .NET client library, which enables the communication with the RESTful APIs of the OpenSocial container using Microsoft .NET based technologies (e.g. Kinect based applications). XML, HTML and JavaScript are also the foundations of Silverlight therefore conversion is easy. OpenSocial provides communications between Flash ActionScript and OpenSocial JavaScript API through its External Interface mechanism. Likewise, our investigation so far suggests that it is possible to develop gadget applications to achieve almost anything typically seen in healthcare related applications. For example, patient parameters (e.g. weight) can be stored in a database with the data acquisition date, and plotted using JavaScript and Google chart APIs. Note, however that in our experience, the development of new functionalities is time consuming due to the lack of suitable tutorials and documentation.

The OpenSocial specification does not say anything about data sharing (e.g. patient parameters) between two gadget applications. Since OpenSocial is a specification, it can be extended accordingly, i.e. we can write our own APIs to extend OpenSocial. Care must be taken when deciding whether functionality is provided by extending the OpenSocial API or by writing methods using a web development tool. For instance, sharing of patient parameters between applications can be achieved by using user profile data from CMS instead of using OpenSocial. When data is stored, it is stored in the platform like Healthcare4Life or Facebook. Security is always primary concern for the container and the applications it is hosting. OpenSocial is part of the container and depends on the services and data of the

platform. Therefore, the security aspects need to be implemented on the platform rather on the applications.

OpenSocial enables developers to embed non-OpenSocial-based applications into containers. However, if an application such as a hand tracking application requires a special plug-in such as Silverlight, this application will not execute within the OpenSocial container without the plug-in. Therefore, it is necessary to develop a testing environment (also known as sandbox) that enables developers to test their gadget application prior to submitting them to appear in the application directory of a system.

OpenSocial specifies a common standard to share social data between two social networks and with OpenSocial applications. Potential developers of Healthcare4Life will be able to host their application on various OpenSocial based container such as MySpace, Orkut and Hi5. However, some of the biggest social networking sites like Facebook do not support OpenSocial.

OpenSocial is a flexible specification that can be treated as a blue print to design large scale enterprise applications, but it mainly focuses on social networking, social media and related entities. If developers are interested to make an enterprise system such as a telehealth system, they have to define their own API and specification according to their requirements if they are not already present in OpenSocial specification.

6.2 Drupal

Drupal reduces the development time for realising our system. The flexibility offered by Drupal and the availability of its contributed modules make it easier to implement functionalities. These modules can be used directly or modified to implement the desired features. Most of the modules are sufficiently documented to understand the source code and its implementation.

The contributed Shindig-Integrator module enables Drupal-based social networking sites to become OpenSocial compliant. However, the Shindig-Integrator module is old and not maintained properly. It depends on other modules, which often get upgraded. It is challenging to understand the Drupal architecture and Shindig-Integrator module code, in order to be able to change it as per Healthcare4Life. Furthermore, the Shindig-Integrator module only supports Apache Shindig release 1.0.x-incubating, i.e. it does not support OpenSocial 2.0. Therefore, we will not be able to integrate the new enterprise and consumer features provided by OpenSocial 2.0 unless the Shindig-Integrator module is upgraded or we invest more time to improve it ourselves.

Drupal has evolved as a significant application development tool, but it suffers from several limitations. Firstly, because it is open source, we cannot rely on open source modules or plug-ins in the long run, as they may not be maintained as is the case for the Shindig-Integrator. Another important issue is selecting the right module for a project. Drupal comes with almost 8500 contributed modules and many of them have similar functionalities. Some modules are easier to use than others, e.g. require the user to write less lines of code to achieve a desired functionality (Buckman 2011). However, Drupal does not provide any guidelines to select the most suitable one. Users are expected to make their own selection by experimenting how each modules fits into their project. Furthermore, Drupal's architecture is quite complicated; it takes a good amount of effort to write new and complex features of a system.

7 Conclusion

We have reviewed popular Web 2.0 technologies for developing web-based telehealth systems. We have specifically investigated the OpenSocial and Facebook APIs and a range of Web Development Tools. Based on our findings, both OpenSocial and Drupal can be used to develop an extensible and dynamic telehealth system. The availability of Apache Shindig (a reference implementation of OpenSocial) and the Shindig-Integrator module of Drupal make it easy to convert ordinary systems to be able to host gadget applications developed by external developers.

We have deployed these technologies to create an online telehealth platform called Healthcare4Life. Results of our deployment show that OpenSocial and Drupal can be integrated successfully to realise a working prototype. We have also developed a gadget application to test the platform. Although, some known minor issues persist, there are many advantages of leveraging such technologies for developing online telehealth solutions. Since our Healthcare4Life prototype is ready, we are all geared to develop more OpenSocial-based health applications and to integrate common HCI devices with the system. We look forward to evaluate our telehealth platform with real users.

8 References

Buckman, B. (2011): Drupal's increasing complexity is becoming a turnoff for developers. New Leaf Digital's Developer Blog. http://benbuckman.net/drupal-excessive-complexity. Accessed 27 August 2011.

Cheng, M. (2009): PHP CMS vs PHP Frameworks. Philosophy in Programming Society, Singapore. http://www.scribd.com/doc/14288264/PHP-Meetup-Feb-09-PHP-CMS-vs-Frameworks. Accessed 10 August 2011.

Dhillon, J. S., Ramos. C, Wünsche, B. C. and Lutteroth, C. (2011): Designing a Web-based Telehealth System for Elderly People: An Interview Study in New Zealand. The 24th International Symposium on Computer-Based Medical Systems (CBMS 2011), Bristol, United Kingdom.

Dhillion, J. S., Wünsche, B. C. and Lutteroth, C. (2011): Leveraging Web 2.0 and Consumer Devices for Improving Elderlies' Health. Australasian Workshop on Health Informatics and Knowledge Management (HIKM2011), Perth, Australia.

DocForge (2010): Framework. DocForge - An Open Wiki For Software Developers, http://docforge.com/wiki/Framework. Accessed 24 August 2011.

Drupal (2011a): Healthcare sites using Drupal. Groups.Drupal homepage, http://groups.drupal.org/healthcare-sites. Accessed 10 August 2011.

Drupal (2011b): Modules. Drupal homepage http://drupal.org/project/Modules. Accessed 10 August 2011.

Drupal (2011c): OpenSocial Shindig-Integrator. Drupal homepage, http://drupal.org/project/ShindigIntegrator. Accessed 10 August 2011.

Drupal (2011d): Webcam trigger. Drupal homepage, http://drupal.org/project/webcam_trigger. Accessed 27 August 2011.

Eldon E. (2011): Facebook Sees Big Traffic Drops in US and Canada as It Nears 700 Million Users Worldwide. Inside Facebook. http://www.insidefacebook.com/2011/06/12/facebook-sees-big-traffic-drops-in-us-and-canada-as-it-nears-700-million-users-worldwide/ Accessed 10 August 2011.

Facebook (2011): Build the social and personalized web. Facebook Developers. http://developers.facebook.com/. Accessed 24 August 2011.

Fischer, S., Wünsche, B. C., Cameron, L., Morunga, E. R., Parikh, U., Jago, L., Müller, S. (2011), Web-Based Visualisations Supporting Rehabilitation of Heart Failure Patients by Promoting Behavioural Change, Proc. of the 34th Australasian Computer Science Conference (ACSC2011), Perth, Australia, Mark Reynolds Eds., pp. 53-62.

Häsel, M. (2011), "Opensocial: an enabler for social applications on the web", Communications of ACM, Jan 1 2011.

Hinchcliffe, D. (2011): OpenSocial 2.0 - Will key new additions make it a primetime player in social apps? Enterprise Web 2.0. http://www.zdnet.com/blog/hinchcliffe/opensocial-20-will-key-new-additions-make-it-a-prime-time-player-in-social-apps/1603. Accessed 10 August 2011.

Ijsselsteijn, W., Nap, H. H,, de Kort, Y., Poels, K. (2007): Digital game design for elderly users. Proc. of Futureplay 2007, Toronto, Canada, 14-18 November 2007), pp. 17-22.

Lee, T. J., Cameron, L., Wünsche, B. C., Stevens, C. (2011): A Randomised Trial of Computer-Based Communications Using Imagery and Text Information to Alter Representations of Heart Disease Risk and Motivate Protective Behaviour, British Journal of Health Psychology (16) 1, pp. 72-91.

McCown, F., and Nelson, M. L. (2009): What happens when facebook is gone? Proc. of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries, Austin, TX.

McIlrath C. (2010): Getting Started with Facebook JS (FBJS). Thinkclay. http://thinkclay.com/technology/getting-started-with-facebook-js-fbjs. Accessed 24 August 2011.

Norval, C., Arnott, J.A., et al. (2011): Purposeful Social Media as Support Platform - Communication Frameworks for Older Adults Requiring Care. ATTACH 2011 Workshop: Advances in Techniques and Technologies Assisting Care at Home, Dublin, Ireland.

OpenSocial (2011a): OpenSocial Specification Release Notes. http://opensocial-resources.googlecode.com/svn/spec/2.0/OpenSocial-Specification-Release-Notes.xml. Accessed 10 August 2011.

OpenSocial (2011b): OpenSocial Core Gadget Specification 2.0. http://opensocial-resources.googlecode.com/svn/spec/2.0/Core-Gadget.xml#Versioning. Accessed 10 August 2011.

OpenSocial (2011c): OpenSocial 2.0. http://docs.opensocial.org/display/OS/Home. Accessed 10 August 2011.

OpenSocial (2011d): OpenSocial API Reference. http://code.google.com/apis/opensocial/docs/0.8/reference/. Accessed 10 August 2011.

O'Reilly, T. (2005): What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software. http://oreilly.com/web2/archive/what-is-web-20.html?page=1. Accessed 10 August 2011.

Purham M. (2010): 30 Social Networking Websites That Were Created Using Drupal. Webdevtuts. http://www.webdevtuts.net/inspiration/30-social-networking-websites-that-were-created-using-drupal/. Accessed 10 August 2011.

Schonfeld, E. (2007): OpenSocial Still "Not Open for Business".Techcrunch. http://techcrunch.com/2007/12/06/opensoc ial-still-not-open-for-business. Accessed 10 August 2011.

Socialbakers (2011): Facebook in 2010: 7.9 new account registrations per second. http://www.socialbakers.com/blog/109-facebook-in-2010-7-9-new-account-registrations-per-second/. Accessed 10 August 2011.

SocialBakers (2011): Health applications Facebook Statistics. http://www.socialbakers.com/facebook-applications/category/105-health. Accessed 10 August 2011.

Shindig (2010): The Apache Software Foundation. Shindig – Welcome to Shindig! http://shindig.apache.org/. Accessed 10 August 2011.

Singh J. (2010): The Best Web Development Frameworks. WebDesignish. http://www.webdesignish.com/the-best-web-development-frameworks.html. Accessed 24 August 2011.

Singh, J., Wünsche, B. C. and Lutteroth, C. (2010a): Framework for Healthcare4Life - A Ubiquitous Patient-Centric Telehealth System. Proc. of CHINZ 2010, Auckland, New Zealand, pp. 41-48.

Singh, J., Wünsche, B. C. and Lutteroth, C. (2010b): Taxonomy of Usability Requirements for Home Telehealth Systems. Proc. of CHINZ 2010, Auckland, New Zealand, pp. 29-32.

Sullivan M. (2011): 9 Reasons to Switch from Facebook to Google+. PCWorld. http://www.pcworld.com/article/234825/9_reasons_to_switch_from_facebook_to_google.html. Accessed 10 August 2011.

water&stone and CMSWire (2009): 2009 Open Source CMS Market Share Report. http://www.cmswire.com/downloads/cms-market-share/. Accessed 10 August 2011.

Weitzel, M., Smith, A., Lee, D., Deugd, S., Helal, S. (2009): Participatory Medicine: Leveraging Social Networks in Telehealth Solutions. Lecture Notes in Computer Science, Series 5597: 40–47. Springer, Heidelberg.

Weitzel, M., Smith, A., Lee, D., Deugd, S., Yates, R. (2010): A Web 2.0 Model for Patient-Centered Health Informatics Applications. Computer 43(7): 43-50.