Introduction

- Telehealth: information and communications technologies used to transmit electronic health information and data across physical distances through the local telecom network.
- Examples of data transmitted: text, audio, video, graphics and other forms of data.
- Telehealth mitigates differences in time, through synchronous or asynchronous messaging methods.
- Telehealth can have many applications, but is particularly advantageous for those who lack convenient access to professional medical help. (Rural residences etc.)

- TeleMedCare (TMC) specialises in providing solutions in Telehealth through its proprietary award-winning Health Monitor system.
- Its current generation is an aging platform that is becoming increasingly obsolete.
- Android tablet technology is a potential platform to replace the aging hardware.
- A new server backend is also needed to complement the hardware.
- A health messaging standard is needed, that should be mobile-friendly and promote interoperability.

This would facilitate communication between the server and Health Monitors, and possibly other health devices and applications.

HL7® FHIR® (Fast Health Interoperability Resources)

- Health Level Seven International (HL7®) is a non-profit, ANSI-accredited standards developing organisation.
- FHIR® is their newest iteration for a messaging standard.
- FHIR® is used for the definition and exchange of clinical data.
- It builds upon the principles of HTTP and RESTful services.
- Its basic unit is the modular resource.
- Resources can be integrated with healthcare and clinical systems to facilitate the solving of clinical and administrative problems.

Development Environment

- Development in C#, using Visual Studio 2015.
- ASP.NET Web API framework used to create the REST-based FHIR API.
- Support provided by the FHIR DSTU2 core support library.
- SQL Server Express used to host and manage the database for the FHIR server.
- Postman, an API testing app used to test the FHIR API throughout development.
- Rigorous unit and integration testing performed on the server using the MStest framework.
- Moq framework facilitated the mocking of dependencies for unit testing.

Implementation of the FHIR®-powered server

Requirements

- Requirements for this project were motivated by a desire to produce a modernised backend for the TMC Health Monitor system.
- 1. Compliance to the FHIR® standard
- 2. Compliance to REST constraints
- 3. Server Maintainability
- 4. Record-keeping of operations performed by server
- 5. Correct handling of requests by server

Implementation of core FHIR® resources

- FHIR resources were chosen for development based on their relevance to Telehealth.
- Each resource contains specific fields of data.
- The Patient, Device and Observation resources were chosen. (Figure 3)

Requirements and Constraints:

- Code-first approach with Entity Framework (EF)
- Each resource has its own Model and Controller (Figure 3).
- A resource model comprises a set of Plain Old CLR Objects (POCOs) which define the structure of the database that is generated by EF.

Figure 4: A REST API was created, allowing users to interact with the data of FHIR® resources stored in the database.

- A REST API was produced which allowed interactions with data stored in the database, through HTTP actions.
- Payloads come in XML or JSON, in a specific FHIR®-specified format.
- GET: Retrieve specific resource
- POST: Create new resource
- PUT: Update existing resource. If non-existent, create new resource.
- DELETE: Delete existing resource.

Repository Pattern

- Repository pattern allows resource controllers to be agnostic to the DAL or type of database used.
- Repositories contain data-access logic, and are introduced to the API controllers via Dependency Injection (DI).

Records System

- Health information is sensitive; documentation integrity of the system is important to ensure accuracy and prevent abuse.
- Every time an interaction is performed that modifies data in the database, a record is created for the affected resource.

Security

- API is designed to be accessible only via HTTPS.
- Enforcement of HTTPS usage is done via data annotations for specific controller methods.

Client Demonstration

- A simple client program was developed to demonstrate how a client with rudimentary UI can be used to access data from the API without explicitly requiring the use of specific URLs.

Future Work

- Only a fraction of a ‘full’ FHIR® implementation was achieved.
- Many more resources can be implemented to flesh out the FHIR® framework.
- Full version-control of resources can be implemented to augment the record-keeping system.
- An account-based system should be implemented to enhance security by defining different tiers of system privileges and to control access to data.
- Explore the use of resource extensions, which provide an avenue for defining additional information stored in each resource (outside of the FHIR® specification).

Conclusion

- The REST API that FHIR® piggybacks on provides the server with a high degree of support for mobile devices.
- FHIR® is an up-and-coming health messaging standard that opens up many options for interoperability between the Health Monitor system and other medical applications using the same standard.
- The server fulfilled all requirements of compliance to FHIR® and REST guidelines, correctness and maintainability of the server, as well as record-keeping.
- A strong argument can be made for a FHIR®-powered server to be a suitable Telehealth framework for the next generation of Health Monitors.

Figure 1: Telehealth breaks physical barriers to provide reliable health care across distances.

Figure 2: A flow diagram of the steps taken during the entire process of development.

Figure 3: The server implementation encompasses the DAL and BIL. The UI is handled by the client.

Figure 4: A REST API was created, allowing users to interact with the data of FHIR® resources stored in the database.

Figure 6: A record is created containing metadata specific to the HTTP action performed.

Figure 5: Different databases, and repositories containing different implementations of the DAL can be swapped around without affecting the behaviour of API controllers.

Figure 7: The simple client program provides a rough proof of concept of how a Health Monitor unit might be able to exchange data with the FHIR® server.