Design and Implementation Issues of a Secure Cloud-Based Health Data Management System

Frank Steimle, Matthias Wieland, Bernhard Mitschang, Sebastian Wagner, and Frank Leymann

Funded By:
- Introduction to the ECHO project
  - Problem description
  - Solution
- Design and Implementation Issues
- ECHO: An Active Health Data Management System
  - System Architecture
  - RESTful Interfacing the Health Data Management Layer
  - Health Data Management
  - Automated Provisioning and Management of the Health Data Management Layer
- Summary and Future Work
• Chronic disease patients may **not have regular communication with physicians**
  – *Lack of time, economic difficulties or negligence*
• This may **induce exacerbation** of their condition, possible hospitalization
• But closer connectivity can ensure **uninterrupted care** and **crisis avoidance**

• Field of Application is **COPD** - chronic obstructive pulmonary disease
• **Combine** human medical expertise with state-of-the-art technologies, like cloud computing, data mining, and smart phones

• **Enable regular monitoring** of patients and avoidance of medical emergencies
  
  – *Patients enter Data on a daily basis*
  
  – *The system analyzes the incoming data*
  
  – *If the system recognizes a dangerous situation it notifies the patient and the doctor*
Design & Implementation Issues
• Data Privacy & Security
  – *Health data are sensitive data and have to be protected by means of encryption, access control, SQL injection prevention, input verification...*

• Support of User Roles
  – *Separation of patients and physicians*

• Different Access Channels
  – *Support many possibilities how a client can access the system: web, mobile, etc.*

• Support of multiple hospitals
  – *The system has to be easily adaptable to different hospitals.*

• Cloud readiness
  – *Enable automatic management and deploying in cloud environments.*
• Easy Development of Applications
  – *Provide Uniform API, good documentation, and tool support.*

• Scalability
  – *System should react fast even if many users access it. Plus, it has to deal with more and more data.*

• Automatic Health Data Analysis and Active Behavior
  – *The System should automatically analyze data provided by the patient and react to possible health problems by generating notifications.*

• Data Quality
  – *The system has to implement mechanisms that ensure data quality.*

• Extensible service-based Architecture
  – *It should be possible to create new services by composing existing services, like Health Services which manage patients’ data or Analytic Services which analyze patients’ data.*
The ECHO System:
An Active Health Data Management System
• Health API used to store data/query the system.
• RESTful HTTP-API to simplify application development
• Resources:

```
<table>
<thead>
<tr>
<th>Subresources of Patient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily Reports</td>
</tr>
<tr>
<td>CATs</td>
</tr>
<tr>
<td>CCQs</td>
</tr>
<tr>
<td>Charlsons</td>
</tr>
<tr>
<td>Readings</td>
</tr>
<tr>
<td>Treatments</td>
</tr>
<tr>
<td>Death</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notification</td>
</tr>
<tr>
<td>Accounts</td>
</tr>
<tr>
<td>Patients</td>
</tr>
</tbody>
</table>

```

```
| 1
| * |
| 1
|   |
| 1
|   |
| 0..1
|   |

```

```
<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Daily Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>CAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>CCQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Charlson</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```

```
<table>
<thead>
<tr>
<th>Death</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
</tr>
</tbody>
</table>

```
- Transport Security: SSL
- Authentication: Token-Based
  - OAuth 2.0 Library
  - JSON Web Token
- Token is digitally signed and contains user information
- Token has to be send in the Authentication-Header
- Token can expire
- Refresh Token can be used to get a new Token
• Implemented using node.js & Express-Framework
• Calling a Health Service via the Health API:
• Swagger is used for Documentation of the Health API
• It is a Description Language for REST Services based on JSON
• Swagger enables:
  – Service Discovery
  – Model Definitions
  – Generation of Client and Server Stubs
• ECHO System comes with SwaggerUI
  – interactive documentation
• MySQL for Health Data Management
  – *RDBMS because it enforces data integrity and has security features like views*

• Health Data Model:
• Data entered by the patient:
  – Q1: *Did your shortness of breath increase?*
  – Q2: *Did your cough increase?*
  – Q3: *Did your sputum change?*
  – Q4: *Did you have chest pain or discomfort?*
  – Q5: *Did you take the same medications? Or increased them?*
  – *...and some measurements, like heart rate and temperature*

• Rule-based Analysis:
  – e.g., Q1, Q2 & Q3 answered with “yes” → “Call your physician!”
Health Data Management: Security

- Database user for each system user
  - *This user is used by the health service to store/query data*
  - *Views filter data depending on logged in user*
  - *Right management of Database is used to secure data*

- User Roles
  - *Available roles: patient, doctor, admin*
  - *MySQL does not support user roles*
  - *Stored Procedure sets appropriate access privileges*

- Injection Prevention
  - *Node.js has no support for prepared statements*
  - *Stored procedures for “writing” Health Services*
Automated Provisioning and Management of the Health Data Management Layer

- System has to be easily deployable in different IT environments
- Management tasks (e.g., backup/workload-dependent scaling) should be automated.
- We used the TOSCA standard to achieve this
  - Topology and Orchestration Specification for Cloud Applications
  - TOSCA is used to describe the platform and the management functions in a machine-readable and self-contained manner
- TOSCA descriptions can be deployed in any supported cloud environment
ECHO TOSCA Topology

- Health API (JavaScript)
- Health Service (JavaScript)
- Health Data DB (Database)

- Node.js 0.10.25 (Node.js)
- Ubuntu 14.04 (Operating System)
- OpenStack VM (Server)

- Tomcat 7.0 (Apache Tomcat)
- Ubuntu 14.04 (Operating System)
- OpenStack VM (Server)

- Analytics Service (Axis2 Web Service)
- Service Orchestration (BPEL Process)
- WSO2 (BPEL Engine)

- MySQL 5.5 (DBMS)
- Ubuntu 14.04 (Operating System)
- OpenStack VM (Server)

(calls)
Summary
### Design and Implementaion Issues: Revisited

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Security</td>
<td>Views, Encryption, Token-based Authentification</td>
</tr>
<tr>
<td>User Roles</td>
<td>Roles: doctor, patient and admin</td>
</tr>
<tr>
<td>Access Channels</td>
<td>Mobile/Web via REST Interface</td>
</tr>
<tr>
<td>Multiple Hospitals</td>
<td>HDML can provide Questions</td>
</tr>
<tr>
<td>Cloud Readiness</td>
<td>TOSCA</td>
</tr>
<tr>
<td>Easy App Development</td>
<td>REST Interface, Swagger Descriptions</td>
</tr>
<tr>
<td>Active Behavior</td>
<td>System analyzes Data and creates Notifications</td>
</tr>
<tr>
<td>Scalability</td>
<td>REST, Node.js can easily be scaled up</td>
</tr>
<tr>
<td>Extensible Architecture</td>
<td>New services can be composed from existing ones</td>
</tr>
</tbody>
</table>
We have built an active Health Data Management System.

Our system can be deployed in many different cloud environments.

We identified issues and showed how to solve them.

Future Work:
- HL7 support
- Data Mining
- Customizable Analyzes