Design and implementation of the temperature component in Personal Healthcare Slow Intelligence System

Zichuan “Jerry” Ye

Department of Computer Science, University of Pittsburgh

Introduction:

Technical challenges in our current healthcare system are increasing rapidly due to the heavily accumulating amount of data in forms of text, image, video and numeric data streams. Conventionally, fast-intelligence systems that handles large incoming data to make medical decisions are widely used in domains of medical studies. However, the lack of initial attempts and biased assumptions give rise to potential inaccuracy in predictions and judgment of these system. As an answer to such problem, Slow-Intelligence System (SIS) in healthcare is considered as one competitive candidate in precisely handling a variety of medical scenarios that are not timely-urgent and is actively changing. Such advantages come from the feature of a slow intelligence system that (i) solves problems by trying different solutions, (ii) is context-aware to adapt to different situations and to propagate knowledge, and (iii) may not perform well in the short run but continuously learns to improve its performance over time.

The Personal Healthcare Slow Intelligence System of our project is a system that collects, processes healthcare data and delivers messages among multiple functional components. This system is built upon a component-based architecture, in which each component is responsible for a certain source of healthcare data or other functions such as uploading and graphic user interface. The components are distributed in a network that follows the typical centralized client-and-server architecture in which a central center serves as the critical node in terms of managing, connecting and integrating the peripheral components. In the design principle, the components that are connected to the SIS-server do not know the presence of each other; they only communicated with the central server "SIS-server" by sending SIS-messages. Each type of message among components is numbered and well-defined. By having access to the number of the messages that are carried by SIS-server, the components recognize the ones that are needed and fetch them from the server. This message-oriented communication supports the flexibility and scalability of the system. It also helps keeping the integrity of the system and expedites the creation, management and termination of the components.

Development of Temperature Component in SIS:

A healthcare system often includes multiple sensors that are designed to handle different types of data that need to be measured and processed separately. According to the concept of "component-based architecture", sensors must be applied in a fashion that each individual sensor does not know the existence of the others and operations of this individual sensor does not rely on other components help. Such component must also be flexible, meaning that once being taken out, other components in the healthcare system need to function properly. In this case, any component designed for SIS-Healthcare system must be able to run independently and be woken up/terminated only by the messages sent out from SIS-server.

The SIS-Healthcare System is a multi-threaded component-based system in which a central server is built to deliver messages sent from components. This centralized server does not involve into the tasks that are needed to receive, parse, process and store healthcare data. Instead, it only plays as a message delivery server that handles incoming messages and either passes the messages to other components or operate system functions such as creating/killing designated components upon receiving default messages. Previous implementation of this system includes a list of component such as "InputProcesser", "SPO2", "BloodPressure", "EKG" and so on. These components are used to handle the types of healthcare data according to their names.

In this project report, implementation of a component that is aimed to handle the temperature data is proposed. This component is names as "Temperature Component" in this report, and labeled as "Temp" in SIS Healthcare System. Since it is a part of the entire SIS healthcare system, several other components are involved in handling scenarios that require functionalities from Temperature Component. Detailed illustration will be presented in the following paragraphs. Please be noted that there is an additional software that Temperature Component needs to run above. The “TEMPerV21.ext” executable files on Windows X86/64 machines is required to have Temperature Component function properly, as we do not have resources to purchase a generic temperature sensor associated with un-complied OEM driver.

Technical Diagram of Temperature Component

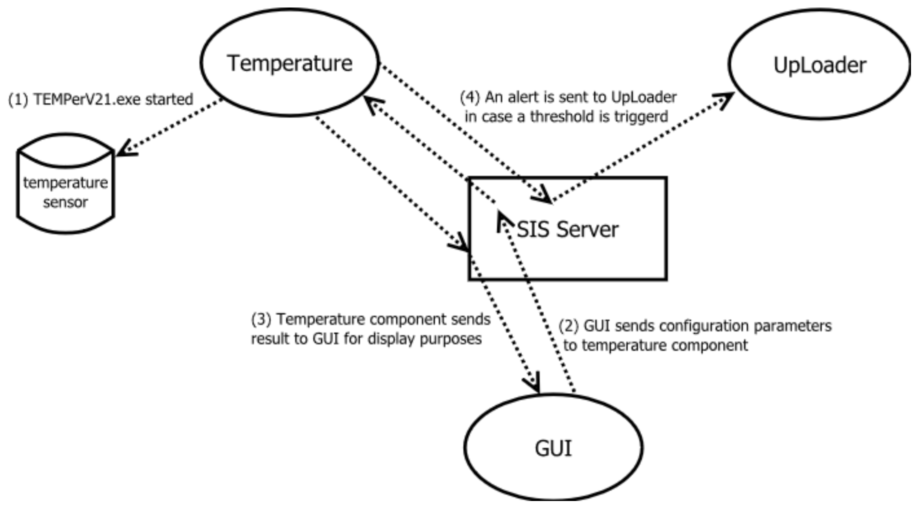
Temperature
UpLoader
(1) TEMPerV21.exe started
temperature sensor
(4) An alert is sent to UpLoader in case a threshold is triggerd
SIS Server
(3) Temperature component sends result to GUI for display purposes
(2) GUI sends configuration parameters to temperature component
GUI

Figure 1. A brief flow diagram describes how temperature component works and communicated with other components.

In this section, implementation details of the component “Temperature” is illustrated in details. The basic design diagram of this component is shown in Figure 1. In this report, Temperature Component is developed and works together with two other SIS-components

“UpLoader” and “GUI”. The “UpLoader” is used to send emails to designated users as well as upload data to cloud databases. “GUI” is the component that a user directly interacts to. It contains a panel of elements such as text field, buttons and view windows for displaying, setting and other functions. It is noted that communications among all components are message-based and can only be carried by SIS Server.

A common scenario that involves the Temperature Component is as follows: A user sets up all parameters associated with temperature sensor in GUI at the beginning. Temperature readings are collected by the physical sensor and data is stored as a local file in the PC by using “TEMPerV21.ext” before starting the Temperature Component. Temperature Component accesses to the local file (.csv) and parse readings from the files according to the date each entry associates in this file. It finds out the current temperature reading and sends the reading to SIS-server in the form of an SIS-message. If some certain values of temperature threshold are reached according to the parameters set by the user, an internal trigger in Temperature Component starts and sends out alerts. As a result, “UpLoader” collects the messages and sends out alarm emails accordingly.

Stepwise description can be seen below. All communications among components are accomplished by sending SIS-messages to SIS-server. Diagram that enumerate the steps of this scenario is depicted in Figure 1.

1. The temperature component is coded and created from as an individual component (CreateTemp.java). It starts an executable program (TEMPerV21.exe) when begins to run. It needs to be mentioned that “TEMPerV21.exe” is compiled

runnable that handles I/O of the temperature sensor and writes temperature data to a local file. By default, the data files is located at "D:\Temp". Due to the limit of software/hardware resources, we must use this executable to receive data from a temperature sensor and store them on local disk.

2. A user set up parameters of the temperature component, including sample refresh rate, sensor start time, sensor end time, temperatures to trigger high/low temperature alerts.
3. After GUI completed reading the user input, the parameters are packed into an SIS-message (#1019) and sent to SIS-server, which later delivers the message to Temperature Component.
4. The temperature component receives the message and decode it. It then starts processing the data according to the parameters that are carried in the message. During its execution, outgoing messages are created and sent out in two scenarios: (a) the temperature component reports to the GUI via messages (#1018) periodically to for current temperature readings, and (b) when the temperature triggers some threshold (high or low temperature), it sends a messages (#38) to the UpLoader.

SIS message:

SIS-messages are the key element in SIS-Healthcare System. Components function independently and communication among them can only be accomplished by SIS-messages that are centralized and distributed by the SIS-server. Numbering the message and assigning them to proper usage is therefore critical in the entire SIS ecosystem. In this study, there are multiple SIS-message that are used to implement a temperature component. The scenarios of how they are used that what they stand for will be discussed in the following paragraphs. For the simplicity reasons, default message are not discussed in the section below.

To handle this scenario, we design a system that works accordingly. The GUI component first takes a user's input parameters of the Temperature Component. It packs the parameters into an SIS-message (#20) to initiate the Temperature Component. Before running the Temperature Component, "TEMPerV21.ext" is triggered so that it starts execution and writing current temperature onto the local disk of PC. SIS-server receives Message #20, which is a system default message, and initiate a standalone Temperature Component using the parameters packed in this message. Once started, Temperature Component reads the local files and sends Message (#39) to GUI for displaying purposes and Message (#38 and #40) to the UpLoader in case a certain temperature threshold is reached. UpLoader then update current temperature reading to the database at cloud and/or sends the temperature reading to the input email address.

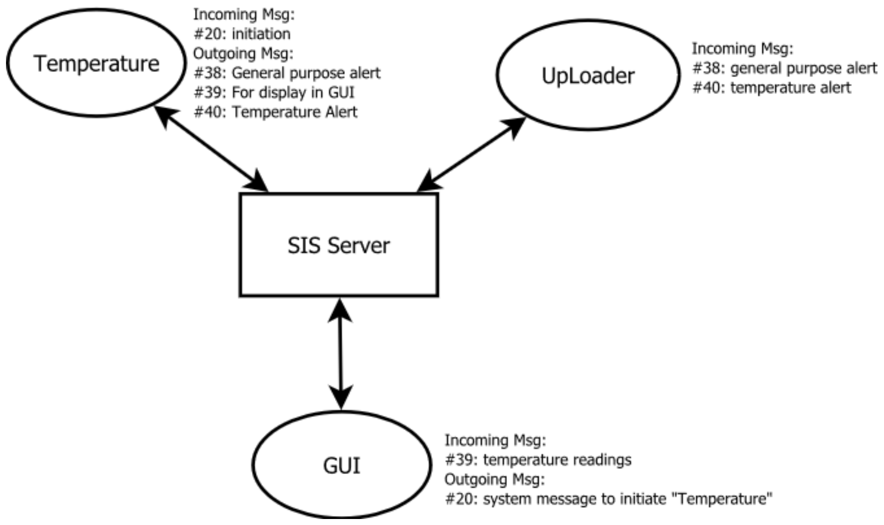

Figure 2. SIS-messages that are used in a component-based architecture that consists a Temperature Component. Multiple functional components communicate via SIS-messages that are delivered by SIS Server. For simplicity reason, the other components such as Blood-Pressure, SPO2 and ECG in SIS Healthcare are now shown here.

Programming perspectives:

In the programming point of view, the codes for Temperature Component, GUI and UpLoader are located in three different directories and have NO internal connection/call to each other. As parts of previous development, GUI (CreateGUI.java) and uploader (CreateUpLoader.java) components were written already. SIS-messages are taken care of in the "switch-case" statement in the infinite loop ("while(true)"). To expedite setting parameters of the temperature component, a class "TempSettings.java" is implemented to create a pop-out window for the main GUI panel and to record user inputs regarding to temperature settings. Parameters are stored as static variables in this class and can be access in GUI by directly calling its static fields.

The standalone temperature component is codes in CreateTemp.java. This program contains its own main method and can read SIS-message and adjust its settings according to the key/value pairs transferred from the message. It uses a buffered reader to read the local ".csv" file that "TEMPerV21.exe" uses to write current temperature on. Please be noted that due to some unknown I/O exception, it is better unplug and then re-plug the temperature sensor from the USB drive every time a new measurement is started.

More details of programming and implementation of Temperature Component in the SIS-Healthcare System is illustrated in the "readme" file.

Conclusion:

In this study, the Temperature Component is designed, implemented in SIS-healthcare system, following a component-based architecture concepts. Temperature readings can be obtained accurately and processed according to the user's input parameters with an easy-to-use GUI. The communications among different components are strictly isolated and can only be completed through the centralized SIS-server. Scenarios that mimic potential threats in temperature change was accomplished when temperature hit certain threshold. Alert messages can be sent to the input email address promptly.

Appendix:

Message #38: general alert message

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0-->
<Msg>
	<Head>
		<MsgID>38</MsgID>
		<Description>Alert Message</Description>
	</Head>
	<Body>
		<Item>
			<Key>DataStream</Key>
			<Value>date_stream_content</Value>
		</Item>
		<Item>
			<Key>GeneralAlertType</Key>
			<Value>context_of_some_alert_message</Value>
		</Item>
		<Item>
			<Key>Protocol</Key>
			<Value>protocol_of_alert</Value>
		</Item>
		<Item>
			<Key>Receiver</Key>
			<Value>some_doctor@upmc.edu</Value>
		</Item>
		<Item>
			<Key>DateTime</Key>
			<Value>"2014/11/18, 18:28.30"</Value>
		</Item>
	</Body>
</Msg>
```

Message #39: temperature reading for GUI display

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0-->
<Msg>
	<Head>
		<MsgID>39</MsgID>
		<Description>Temperature Reading</Description>
	</Head>
	<Body>
		<Item>
			<Key>TempEnable</Key>
			<Value>True/False</Value>
		</Item>
		<Item>
			<Key>Temperature</Key>
			<Value>sample_temperature</Value>
		</Item>
		<Item>
			<Key>DateTime</Key>
			<Value>sample_date</Value>
		</Item>
```

```
        </Body>
</Msg>
```

Message #40: temperature alert message

```
<?xml version="1.0" standalone="yes"?>
<!--Generated by SISProjectCreator Version 1.0-->
<Msg>
        <Head>
                <MsgID>40</MsgID>
                <Description>Temperature Alert</Description>
        </Head>
        <Body>
                <Item>
                        <Key>Alert Type</Key>
                        <Value>high temperature alert</Value>
                </Item>
                <Item>
                        <Key>AlertMessage</Key>
                        <Value>sample_temperature</Value>
                </Item>
                <Item>
                        <Key>DateTime</Key>
                        <Value>"2014/11/18, 18:29.15"</Value>
                </Item>
        </Body>
</Msg>
```