Japanese HWR

Steven B. Poggel
steven.poggel@gmail.com

January 26, 2010

Contents

Chapter 1

Technical Design of the Application

The focus of this chapter is on the general architectural choices made during the development of the system. In this chapter, the technical design aspects of the application are described. The general system architecture is layed out in section 1.1. It contains the global view on the software architecture in section 1.1.1, the data flow in within the system in section 1.1.2 and describes the design of the individual modules in section 1.1.3. Section 1.2 describes the technical set-up and framework choices. However, the handwriting recognition engine is described in detail in a separate section (see chapter **??**).

1.1 System Architecture

The system architecture of the Kanji Coach follows the requirements of an e-learning environment dealing with the specific difficulties for learners of the Japanese script (see chapter **??**) and those of an on-line handwriting recognition. Techniques of handwriting recognition are reviewed in chapter **??**. The general requirements of an e-learning application are presented in chapter **??**. The resulting specific conceptual design choices have been layed out in chapter **??**. This section deals with the technical aspects of the system design.

1.1.1 Global Architecture

The global architecture of the application follows the Model-View-Controller (MVC) design pattern. This paradigm is used as a general model, however, it is not implemented the strict way proposed by (Krasner and Pope 1988). Figure (1.1) shows the general set-up of the MVC design pattern after (Krasner and Pope 1988). xxx: Also figure (1.2) - decide how it should be done and use the appropriate gfx. xxx! In the MVC paradigm the *model* is a domain-specific software, an implementation of the central structure of the system. It can be a simple integer, representing a counter or it could be a higly complex object structure, even a whole software module. The *view* represents anything graphical. It requests data from the model and displays the result. The *controller* is the interface between the model and the view. It controls and scheduls the interaction between the input devices, the model and the view (Krasner and Pope 1988).

A global overview of the system architecture can be seen in figure (1.3).

1.1.2 System Data Flow

The system data flow is shown in figure (1.4). The controller lies in the centre of the application, it runs on the stationary device. It contains a web service that is used as an interface to receive data from the mobile view. The desktop view is the main interaction point for the user. The model contains the logic, while the data access layer provides a reusable interface for storing data.

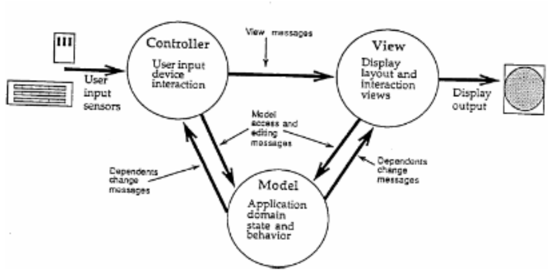
Controller
User input device interaction
View messages
View
Display layout and interaction views
User input sensors
Display output
Model access and editing messages
Dependents change messages
Dependents change messages
Model
Application domain state and behavior

Figure 1.1: The Model-View-Controller paradigm

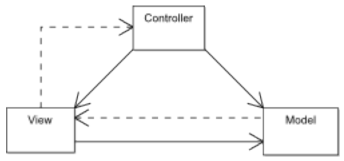

Figure 1.2: The Model-View-Controller paradigm AGAIN!

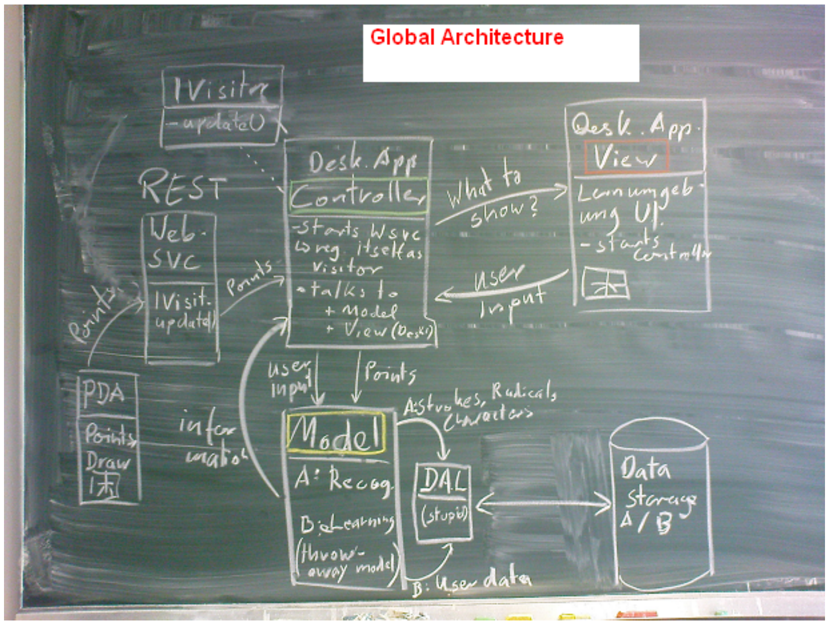

Figure 1.3: The global architecture of the software system

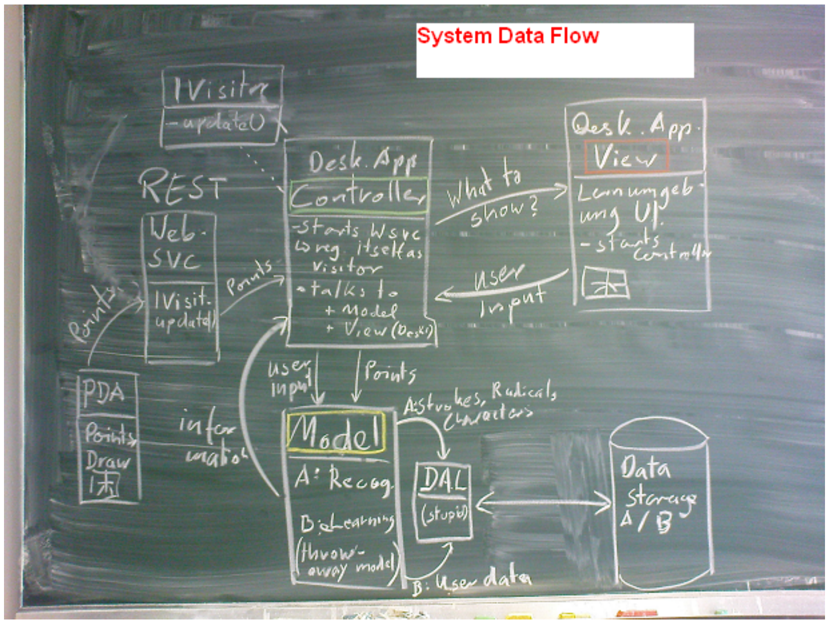

Figure 1.4: The data flow within the software system

1.1.2.1 Communication

The communication between the different parts of the application is realised in two independet and distinct ways. The communication between the modules running under the same process is realised via a messaging system. The communication between the modules that run on different devices or at least as a separate process is realised via a web service.

Loose coupling is used here as a term to emphasize that different modules in a larger system are only loosely connected and do not depend largely on each other. *Coupling* can be understood as the degree of knowledge a module or class have of each another. The lesser the knowledge of each other the software modules can manage with, the more loose the coupling.

In the course of the design and development cycles of the Kanji Coach, it became apparent that it has to be possible to attach different views, input devices and data storage systems to the controller. This is due to the distributed nature of the application. In order to ensure that the handwriting data input view, which currently runs on a mobile device, can run on a different device, it had to be loosly coupled to the main controller. Therefore, the communication between the mobile view and the main controller is realised via a web service. Because of this communication structure, it is possible to exchange the handwriting data input view with a different one, for instance when running the application on a device like a tablet PC. The implementation of the communication structure within the web service is described in greater detail in section 1.1.3.3.

The messaging system that forms the communication structure between the software modules running as the same process is realised with a message class that can be manipulated by the modules using these messages. Technically, the web service and the controller both run as a subprocess of the main desktop view. When the web service receives a request and accompanying data from the mobile view, both request and data are bundled into an encapsulated message and passed to the controller. A similar type of message is used for requests from the controller to the model, for instance a recognition task that needs be performed.

1.1.2.2 Recognition Data Flow

1.1.2.3 Learning Data Flow

1.1.3 Software Modules

1.1.3.1 Mobile GUI

1.1.3.2 Desktop GUI

1.1.3.3 Web Service

http://de.wikipedia.org/wiki/Windows$_C$ommunication$_F$oundation http://en.wikipedia.org/wiki/Web$_s$ervice http://kleineurl.de/1kfd1k.htm

1.1.3.4 Recognition Module

1.1.3.5 Learning Module

1.2 Framework and Devices

1.2.1 Operating System

1.2.2 Framework

http://de.wikipedia.org/wiki/Windows$_C$ommunication$_F$oundation

.NET vs. Java etc.

1.2.3 Desktop Computer

1.2.4 Pen Input Device

warum mobil? - damit man sehen kann, was man macht!

1.2.4.1 Stylus Input

List of Figures

List of Tables

References

Krasner, G. E. and S. T. Pope (1988). A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program. 1*(3), 26--49.