MultiModal Interfaces (MMI):
The Service Counter System Toolkit

Document, Image and Voice Analysis Research Group (DIVA)
Department of Informatics (DIUF), Faculty of Science
University of Fribourg, Switzerland

Authors: Pedro de Almeida, Dominique Guinard, Martin Eric Ritz

Report Type: Final Project Report
Date: 29.06.2006

Table of contents

Abstract:

We have designed and implemented a toolkit for building states centric user interfaces providing a seamless integration of concurrent input and output modalities. The system is extensible through other toolkits, so new actions can be quickly added. The modular design of the code allows to easily replace parts of the system.

1 Introduction

This project was developed in the context of the lecture “multimodal interfaces” and is considered as contribution to the Memodules Award. The task consisted in the conception and realization of a prototype integrating several modalities such as voice, vision, etc. on the one hand and Phidgets[1] on the other.

We decided not only to develop a prototype but a toolkit, since this can be used many more generally and so it is more powerful. With the help of the Service Counter System Toolkit it’s possible for the user to create states centric user interfaces, which can contain a multiplicity of in- and output modalities, without large expenditure of time. In chapter two we will describe our toolkit in some details.

Nevertheless in chapter three we also would like to provide a concrete use case of our system. The multimodal Librarian Service Counter is a hard- and software-based user interface, which implements some task of the administrative work in a library and automates the basic customer service.

In chapter four we then conclude the project and this report by summarizing some important points.

[1] **Phidgets** are a set of building blocks for low cost sensing and control from a personal computer. Using the Universal Serial Bus (USB) as the basis for all Phidgets, the complexity is managed behind an Application Programming Interface (API). Applications can be developed in Mac OSX, Linux and Windows operating systems.

2 Toolkit description

First we would like to have a look at the toolkit, focusing on the architecture and operational principles. This is to give an overview and create a common basis which then permits the reader to better understand the use case of the following chapter.

2.1 Architecture

Figure 2-2-1 illustrates the major components of the Service Counter System toolkit's architecture. The design of the toolkit is based on our personal experience in developing and deploying user interfaces. It heads towards a seamless integration of modalities and multimodal libraries aggregated in a single final user interface.

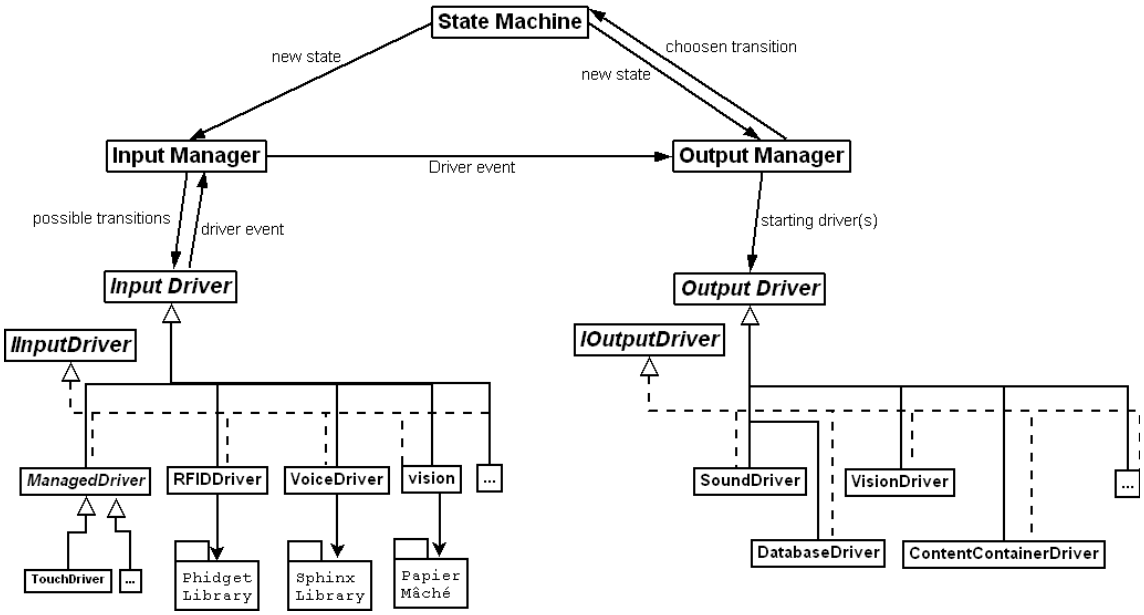

Fig. 2-2-1: the Service Counter System Toolkit's architecture

2.1.1 The State Machine

In the conception of the toolkit, we made use of the Automata Theory. Using the toolkit the programmer has to define a deterministic finite automaton[2] with several states. At the be-

[2] A deterministic finite automaton (DFA) is a deterministic finite state machine which is a finite state machine where for each pair of state and input symbol there is one and only one transition to a next state.

ginning we are in an initial state. The so-called `State Machine` communicates the actual state and all possible transitions for this new state to the `Input` and `Output Manager` after each state transition. In term, these are communicating the transitions to both the input and output drivers. The way this communication is implemented is based on the Observer Pattern[3] well-known from E. Gamma et Al.'s book about Design Patterns.

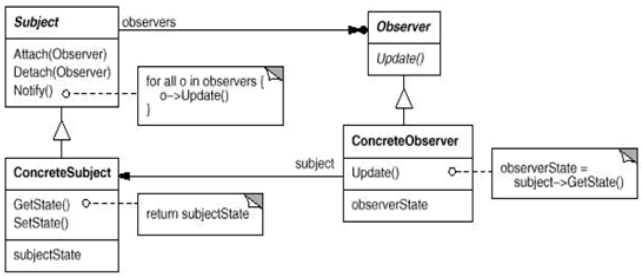

Fig. 2-2-1-1: Observer Pattern Structure

2.1.2 The Input and Output Managers

Getting the Information about the new state, the `Input` and also the `Output Manager` communicate all possible transitions for this state to all the Input Drivers respectively to all the Output Drivers. In each state one declare which in- and output devices are enabled. There is no limitation of the number of these devices. In the case where we have more than one event at the same time responding to a transition, the first arriving will be treated. If there is an event of a currently disabled input device driver, the concrete Input Driver (e.g. RFIDDriver) sends `NIL` to the `Output Manager` and there is no state transition. If the event originated from an enabled driver, the concrete Input Driver sends the driver event to the `Output Manager` and this one will decide what will happen with the event and then communicate the chosen transition to the `State Machine`. It also starts one or more output drivers intended for the appeared event. To avoid several driver event messages for the same event, the Input Driver can apply a filter on the accordant device driver.

2.1.3 The Input and Output Drivers

In principle in- and output devices are separated from each other and implemented to differentiate them completely. For each possible in- and output device a suitable driver is available (see also the list of drivers in the annexe). For this purpose the toolkit integrates and ab-

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. p. 293ff

stracts various open source solutions such as SPHINX, the Papier Mâché and the Phidget toolkit. For the phidgets a special Driver, has been implemented to build a common base for them. This was necessary because of the fact, that the phidget library was built in Visual Basic and then automatically transformed into Java. As a consequence it's not really Object-oriented and thus, not easily integrable "as-is" in a toolkit. To keep the independence of device-types for the entire toolkit, the so called `Input Driver class hierarchy` was created. All drivers extend a general abstract Driver, called `InputDriver` respectively `OutputDriver` which contains functionalities common to all the drivers.

2.2 Installation Guide

This short guide will help you to install the Service Counter System Toolkit. Before you install make sure that you have the following things ready:

- A database (MySQL database version 5.0) named "`scs`" with username and password "`root`".

To use the toolkit you have to process to the following two steps:

1. execute the sql file `scs_database.sql` of the folder `/data/sql`
2. Then, all you need is the `ServiceCounterSystem.jar`, which you find on the attached CD. The JAR file contains all the files and other resources from the toolkit. You simply can include the jar file directly in your project e.g. in Eclipse. (see "Import third party JARs" from the Eclipse Tutorial[4])

From here you will be able to use the Service Counter System Toolkit.

[4] https://eclipse-tutorial.dev.java.net/eclipse-tutorial/part1.html

3 Use case: A multimodal Librarian Service Counter

3.1 Main idea

In a library there are some administrative works within the range of the customer service which are repeated each day for many times. On the one hand these are the lending and taking back of existing books and on the other hand also recording new books into the book catalog. The multimodal Librarian Service Counter is a hard- and software-based user interface, which implements these tasks.

Fig. 3-1-1: The multimodal Librarian Service Counter

3.2 Components / modalities

In the Librarian Counter System the following modalities are used as paths of communication between the human and the computer:

- Vision:
  - Input:
    - Take a photo of a book via video camera
  - Output:
    - Ten LEDs to indicate the user which input possibilities are actually enabled

- Touch:
    - Input:
        - Touching Sensors for OK and Chancel operations
        - Panic Button for stopping all activities
        - A Rotation Sensor, two slider sensors and a mini-joystick for choosing different options
        - A switch for turn on/off the hardware

- Auditory:
    - Input:
        - Speech recognition allowing the user to control the system via voice
    - Output:
        - Speech output to inform the user about current state

- Kinaesthetic:
    - Input:
        - Replacement of a book in one of two predefined RFID areas
        - Trace user movements using the video camera and Papier Mâché
    - Output:
        - Two Servo Motors to calibrate the camera

3.3 State automaton

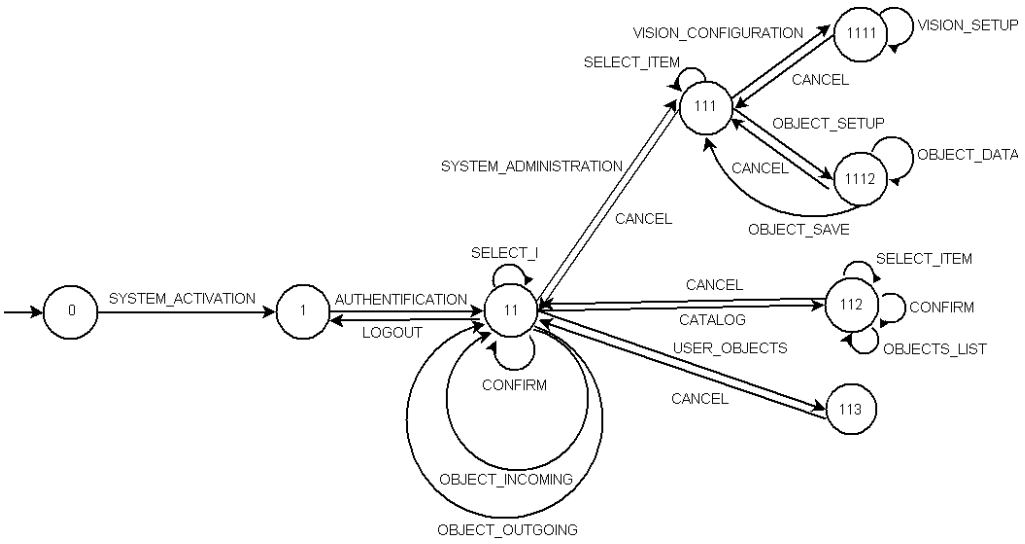

Fig. 3-3-1: State automaton for the multimodal Librarian Service Counter

4 Conclusion

Our idea was to build a re-usable toolkit for building user interfaces that accept multiple concurrent modalities from various toolkits. This task is far from being easy and the library we presented here is certainly not complete. However it heads towards the re-usability of multimodal interfaces which is, we believe, a key factor for both the user and inducting acceptance of such systems.

5 Acknowledgements

The editors would like to thank to Irène Ruffieux for her valuable assistance. Her pleasant voice gives a special character to the multimodal Librarian Service Counter by different voice outputs.

6 References

E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison-Wesley, 1995.

7 Annexe

List of Input Drivers:

RFIDDriver:	capture of RFID events
TouchDriver:	touch buttons (confirm, cancel)
VoiceDriver:	voice recognition
VisionDriver:	objects recognition via camera
Joystick:	menu navigation

List of Output Drivers:

CameraMotorDriver:	enables camera movement
ConsoleContainerDriver:	handles confirm and warning messages
ContentContainerDriver:	builds menu items and other special components
ObjectContainerDriver:	displays book information and handles events related to it
DatabaseDriver:	makes database queries