GALLAG Strip: A Mobile, Programming With Demonstration Environment

for Sensor-Based Context-Aware Application Programming

by

Luis Garduno Massieu

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2012 by the
Graduate Supervisory Committee:

Winslow Burleson, Chair
Eric Hekler
Sandeep Gupta

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

The Game As Life – Life As Game (GALLAG) project investigates how people might change their lives if they think of and/or experience their life as a game. The GALLAG system aims to help people reach their personal goals through the use of context-aware computing, and tailored games and applications. To accomplish this, the GALLAG system uses a combination of sensing technologies, remote audio/video feedback, mobile devices and an application programming interface (API) to empower users to create their own context-aware applications. However, the API requires programming through source code, a task that is too complicated and abstract for many users. This thesis presents GALLAG Strip, a novel approach to programming sensor-based context-aware applications that combines the Programming With Demonstration technique and a mobile device to enable users to experience their applications as they program them. GALLAG Strip lets users create sensor-based context-aware applications in an intuitive and appealing way without the need of computer programming skills; instead, they program their applications by physically demonstrating their envisioned interactions within a space using the same interface that they will later use to interact with the system, that is, using GALLAG-compatible sensors and mobile devices. GALLAG Strip was evaluated through a study with end users in a real world setting, measuring their ability to program simple and complex applications accurately and in a timely manner. The evaluation also comprises a benchmark with expert GALLAG system programmers in creating the same applications. Data and feedback collected from the study show that GALLAG Strip successfully allows users to create sensor-based context-aware applications easily and accurately without the need of prior programming skills currently required by the GALLAG system and enables them to create almost all of their envisioned applications.

DEDICATION

To my parents and my sister, for their endless love, support and understanding.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Winslow Burleson, for his guidance and for giving me the opportunity to be part of his research group. And to the rest of my committee: Dr. Eric Hekler for his valuable input and to Dr. Sandeep Gupta for his support.

I also want to thank the members of the Motivational Environments research group; everyone helped me in some way or another. Special thanks to Jisoo Lee for all her help during the user study and to Erin Walker for her mentoring. Also thanks to Cecil Lozano, Ryan Brotman, Helen Chavez and Javier Gonzalez for their input.

Thanks to Ana Enciso and all my friends who either helped me with their participation during the study or just by being there when I needed them.

Finally, thanks to Consejo Nacional de Ciencia y Tecnologia (CONACYT) for their financial support throughout my studies.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

INTRODUCTION: GAME AS LIFE – LIFE AS GAME

Game As Life – Life As Game (GALLAG), a project from the Motivational Environments research group at Arizona State University, investigates how people might change their lives if they think of and/or experience their life as a game. GALLAG aims to help people reach their personal goals through the use of context-aware computing, and tailored games and applications [6]. Finally, GALLAG also tries to enhance home life and expand what people think computing can do for them.

To accomplish this, the GALLAG system uses a combination hardware and software to empower users to create their own context-aware applications, which we refer to as GALLAG applications (see figure 1).

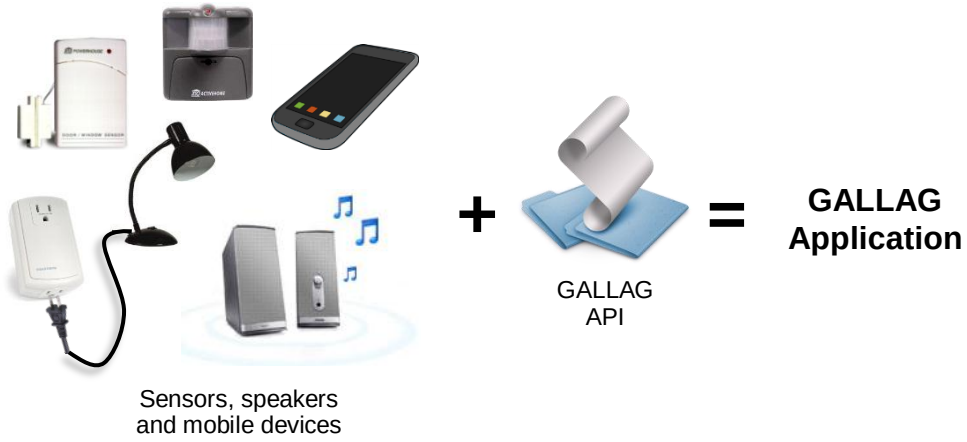


Figure 1. Components needed to create a GALLAG application

The hardware is composed of wireless and wired sensing technologies, local and remote audio/video feedback and mobile devices. The software consists of an application programming interface (API) based on the AppleScript [7] programming language.

One of the most important features of GALLAG is that it allows users to customize an already existing GALLAG application (e.g., a sample application or a template) or create a new one by using GALLAG's API.

GALLAG API

The GALLAG team noticed that the average novice GALLAG user was not able to create GALLAG applications from scratch using AppleScript code. This motivated me to lead the team in the development of a programming library in order to address this problem, and this is how the GALLAG API started.

The GALLAG API consists of a collection of AppleScript functions that abstracts complicated configuration and control actions so that users can more easily understand, create, or modify GALLAG applications by reading and writing AppleScript code.

The main advantages of GALLAG's API are that it significantly reduces the amount of code that a user needs to write when creating a GALLAG application, and that it provides a consistent way to configure and communicate with devices (e.g., sensors, actuators and mobile devices), as well as with applications (e.g., audio/video playing software, word processors, file and web browsers) (see figure 2).

```
setDeviceOn("MyLamp")
```

```
iTunesPlayFile("/Users/GALLAG/Welcome.mp3")
```

Figure 2. Example use of GALLAG's API

In the current implementation of the GALLAG API [29], we have provided support for the following: remote audio playing with one or more wireless speakers, database logging, sensing devices setup, communication with iOS mobile devices, audio/video media

playing, communication with iRobot, opening a web page or a file in the server machine or a remote computer, write to Microsoft Word, text to speech and input for voice commands, among others.

Sample GALLAG applications

The GALLAG team has also worked on creating sample GALLAG applications as a way to introduce novice users to the GALLAG system. These ready-to-run sample applications have two main objectives: first, to help users understand how the system works and the type of applications they can create, and second, to serve as templates that users can quickly test and modify to better suit their needs.

As part of the GALLAG team, we have created sample GALLAG applications and posted them on the GALLAG wiki site [30] for easy user access (see figure 3). One of these sample applications created by Jisoo Lee, shows how to use open-closed magnetic sensors and play music if a sensor is opened [31]. Another example by Jisoo Lee is a matching game using audio and visual feedback [32]. An example that I created allows users to play a treasure hunt game with a GPS-enabled Apple iOS [33] device (e.g. iPhone, iPad) [34]. Lastly, an additional application that I created with the help of Byron Lahey, reminds users to keep in contact with family and friends by using a Pendaphone [35] and calling them through Skype [36].

We have also provided users with information on the wiki site about how to setup a GALLAG system (hardware and software) and how to get started with basic AppleScript programming.

Problem statement

As mentioned earlier, a major goal of GALLAG is to enable users to create their own GALLAG applications or to modify an existing one.

Although we have tried to make the GALLAG API as easy to use as possible, it still requires a fair amount of programming knowledge and poses a substantial cognitive load to the user, which becomes a significant barrier for users that do not know how to program. Additionally, the sample GALLAG applications that we have created are not quite at a level where users without programming experience can quickly modify to create customized versions of them, as a small amount of programming is necessary.

This triggered the question of how could this barrier be lowered and make GALLAG programming an easier and engaging activity for novice users.

Proposed solution

GALLAG Strip, the system discussed in this thesis, attempts to address these issues by providing a Programming With Demonstration environment which enables users to create GALLAG applications in an intuitive and appealing way without the need of computer programming knowledge; instead, they program applications by physically demonstrating their envisioned interactions within a space. This demonstration is done using the same interface that will later be used to interact with the GALLAG system, that is, using GALLAG-compatible sensors and mobile devices to realize personalized, empowering, and life-changing games and applications.

GALLAG Strip provides a novel approach to programming sensor-based applications by combining the technique of Programming With Demonstration with a mobile application to enable users to more easily experience their applications while they program (i.e., demonstrate) them, thus lowering the level of programming abstraction. GALLAG Strip's main objective is to allow novice GALLAG users to create most of their envisioned applications easily and accurately.

The proposed solution consists of a sensing and application running commercial software, a server application for web service communication and application code generation, and

a mobile application with a simple and appealing user interface based on the comic strip metaphor (i.e., having a strip or sequence of frames) that will help users to program their GALLAG applications.

The Programming With Demonstration technique, explained in the next chapter, was selected because of its effectiveness when programming context-aware applications shown in previous research.

Example-Based Programming (EBP) is a technique that simplifies the programming process by avoiding the need of textual notation (i.e., writing source code). With EBP, users are not forced to learn an unnatural, abstract programming language or standard computer science concepts like conditionals, loops and variables; instead, users provide examples of their intended actions with a specific application [8]. The main advantage of EBP is that the examples are demonstrated in the same environment in which users perform their actions, or as Dan Halbert defines it, it is like "programming in the user interface" [12].

The earliest documented reference of EBP dates back to 1969 with Teitelman's PILOT system, which acted as an intermediary between users and their intended program code [28]. The PILOT system allowed users to provide high-level descriptions of their intended tasks and produce programs to accomplish them. Teitelman's vision was to create a cooperative and helpful programming environment that would free users from routine aspects of programming so that they could be more ambitious, productive and creative.

Types of EBP

There are two types of EBP: Programming By Demonstration (PBD) and Programming With Demonstration (PWD). PBD denotes systems that infer the user's intended program through a series of example inputs, actions and outputs, and creates a generalized program.

	PWD	PBD
Advantages	Simple for users to understand and edit.	Provides flexibility with generalized programs.
Disadvantages	Not very flexible, what is demonstrated is what is programmed	Hard to infer user intent. Difficult for users to understand and edit.

Table 1. Comparison between PWD and PBD

With PWD systems, users can also program their applications with an example; however, they need to specify all the details, as there is no inference involved. As exemplified by Halbert, PWD can be described as “Do What I Did”, and PBD as “Do What I Mean” [23]. Table 1 shows a comparison of the advantages and disadvantages between PWD and PBD.

The advantage of a PWD system is that it makes it easy for users to understand how an application gets created and how to modify it, as there is a direct mapping between the demonstration and the programmed application. With PWD, what users demonstrate is exactly what gets programmed; if a user demonstrates a sequence of actions, this exact sequence is what the application will perform. However, this makes a PWD system not very flexible if the user does not follow the exact sequence of actions all the time, in which case he/she would need to demonstrate a variation of the actions and create a different application for them.

In contrast, the advantage of a PBD system is that it has the ability to infer what the user wants to program through a series of demonstrations; if a user demonstrates a sequence of actions in his first demonstration and then demonstrates a variation of these actions for the same application, the system will create a generalized application that reacts to these two sequence of actions. The disadvantage of this inference capability of PBD systems is that sometimes is hard for the system to create a generalized application from demonstrations that are very different, and it also makes it hard for users to understand what exactly the generalized application will react to and how to modify it, as these applications are created based on several demonstrations.

EBP for sensor-based context-aware applications

Although EBP has been used for general computer programming in some projects like Teitelman’s [28] and others like APE [27], most have focused on automating repetitive tasks in a domain-constrained graphical user interface environment like CAD automation

[11], graphical editing in Chimera [18] and Mondrian [19], and user interface creation in Peridot [24]. Another example is Modugno's Pursuit [21], which provided a graphical representation of programs in a demonstrational visual shell. An interesting feature in Pursuit is the comic strip metaphor (i.e., showing the before and after states in a sequence of frames) used on desktop icons to represent users' actions on real data files. I was inspired by this representation of user actions as it makes it easier to visually understand an application based on actions and events along time.

Recently, PBD has been used for context-aware, ubiquitous and physical applications. Anind Dey et al. developed "a CAPpella", a PBD system to help users develop context-aware applications [10]. Their approach consisted on letting users demonstrate a specific behavior using several sensors and then review the gathered data by marking the relevant parts of the demonstration as well as the desired actions. A CAPpella's user interface was a PC desktop application showing continuous streams of sensor data and event information in a graph that the user later had to analyze and determine which part to keep and which to ignore. Although their evaluations were successful, they mentioned that users had difficulty when selecting relevant parts of the data streams and marking events. In contrast to a CAPpella, my vision is to provide users with an interface that does not require them to analyze streams of raw data but rather concrete events and actions that they can easily understand.

An example of a context-aware PWD programming system for end users is iCap by Anind Dey *et al.* [9]. They developed a visual, rule-based, environment in which users could prototype context-aware applications without writing code. In their system, users first define the objects, the activities, the locations, the people and the time they want to use in their programs and are then able to drag and drop them into panels to create rules with associated actions. They performed a user study to identify the kind of context-aware programs that users were interested in and found that most of their programs were

specified using simple if- then rules (79%). They also noticed that the rules were fairly simple, with an average of 2.5 elements used in each rule, where an element could be an activity, object, location, time, person or state. Their findings support my belief that most sensor-based user applications can be defined using simple if-then rules and that they employ a small number of rule elements.

Papier-Mâché by Klemmer and Landay [16] introduced a modular, event-based architecture to support tangible user interface programming. By employing high-level input abstractions, their developer-focused API lowers the difficulty threshold to work with physical (i.e., sensor) input and makes it easier for applications to switch to different input technology. Although GALLAG Strip differs in that it is intended for end users and not developers, I liked their mixed-methods approach for evaluating the usability of their system, as there are no best practices for emerging areas like ubiquitous and physical computing.

A successful PWD project that used the Papier-Mâché architecture was SiteView [1,16], a rule-based physical programming interface for a home automation system. SiteView employed a ceiling-mounted camera that locates physical icons over a floor plan representing the environment, RFID (Radio Frequency ID) tags and readers for rule creation and editing, a rule display showing the rules in text, and an environment display to preview the rules in action. They argue that lowering the programming difficulty threshold for novice users does not necessarily mean that the system will require machine learning techniques or complex inference methods [1]. I agree with their argument and I believe that the programming threshold can be lowered with a good user interaction experience, which has proven true for several other PBD systems [8].

Finally, just last year, a Kickstarter project called Twine [37] is working on bringing to market user-programmable wireless modules with embedded sensors for temperature and vibration, and an expansion connector for additional sensors. Their PWD approach is

a visual, rule-based web editor that lets users create if-then rules to program their sensor modules. Having raised more than half a million dollars just over a month to fund their project is a good indicator of the current demand for sensor-based context-aware end-user programming and it provides additional evidence that an if-then rule based programming model is (or at least appears to be) effective for end-users.

Motivation

Today, most of the applications are programmed not by professional software developers, but by people with diverse backgrounds working towards goals supported by computation [17]. These people, which are also end users, know better how to describe the tasks they are interested in programming, however, programming can be a daunting task for them; traditional development tools to create interactive applications require extensive programming knowledge and can be difficult even for experienced programmers [20].

The API for GALLAG (described in the preceding chapter) is not at the level of users without programming experience, as it requires them to write scripts, which is still programming and the hurdle to learn a scripting language is simply too high for the average computer user [8].

The research projects described in the previous section have shown the potential of PBD and PWD systems to enable end users to program sensor-based context-aware applications. However, although they effectively lowered the programming difficulty level for users, they do not address the issue of detachment from real world actions. This issue causes two main problems: one is that users need to move away from the environment where they are acting on in order to use the programming system (e.g., a desktop PC-based system), and the second is that they need to spend time understanding the abstract representation of their actions after they performed them.

GALLAG Strip differs from previous research efforts in that it provides a simple, mobile rule-based programming interface that allows users to see their actions in real time, thus reducing their detachment from the world.

GALLAG Strip was built as a visual programming system, as it has been proved that a visual representation of a program's code is easier to understand than in text form, especially if they are short [21,23]. Additionally, most end user-created programs are small and do not contain nested loops or nested conditionals [22], therefore being relatively easy to represent visually. GALLAG Strip's if-then rule visual programming interface was based on the comic strip metaphor (see figure 4), similar to the one in Pursuit [21].

The decision to build GALLAG Strip as a PWD system (instead of PBD) was based on previous research showing that it is easier for users to understand, use and modify applications using a PWD system than a PBD one [23]. Also, because GALLAG Strip was designed to be used by a single user, the assumption is that the demonstrated user actions will not vary much and thus a PWD system will suffice.

Although I recognize that being able to create generalized applications with a PBD system could provide some flexibility while creating applications, such an environment would have demanded considerably more time to develop and to test with users because of the required inference engine. However, the current implementation of GALLAG Strip can be modified later to include an inference engine, converting it into a PWD system with more flexibility.

There were two major challenges while developing GALLAG Strip. First, there was the challenge of creating an appealing user experience that would make it easy for users to create GALLAG applications through demonstration. The second challenge was to implement it using a medley of technologies that were not used together before on the GALLAG project. The following sections describe how these challenges were tackled.

User experience

The design of the user experience was an iterative approach and had elements of participatory design. It started with a series of low-fidelity prototypes on paper which, after a round of iterations, were transformed into medium-fidelity, interactive prototypes using Microsoft Expression Blend and SketchFlow [38] (see figure 3 and 4).

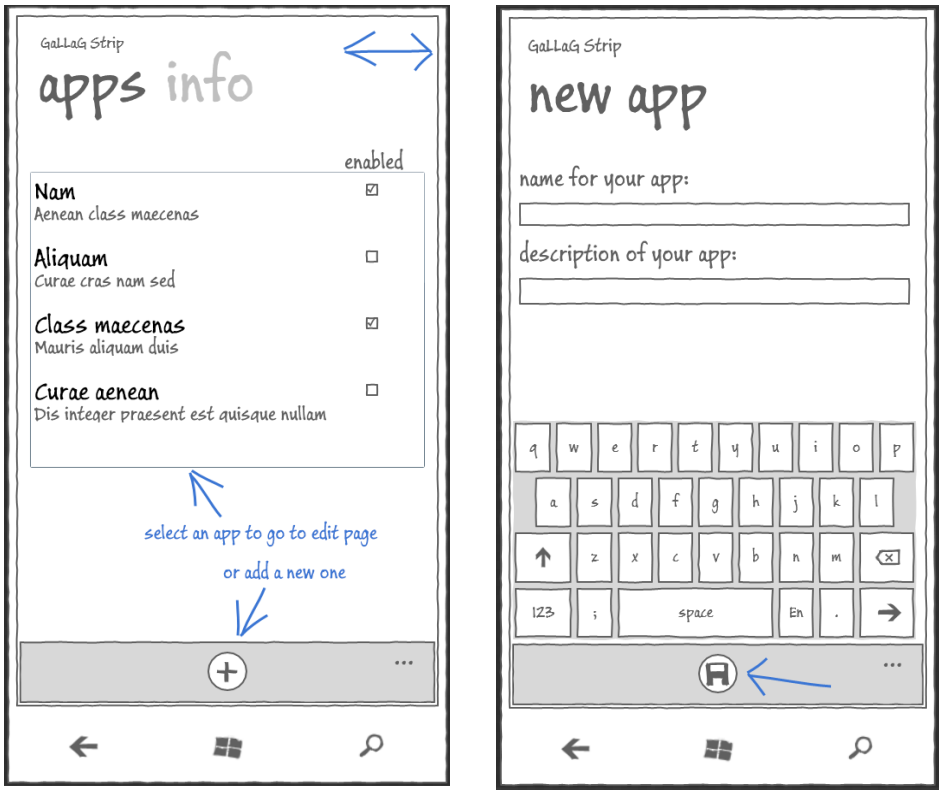


Figure 3. Initial mobile application sketches

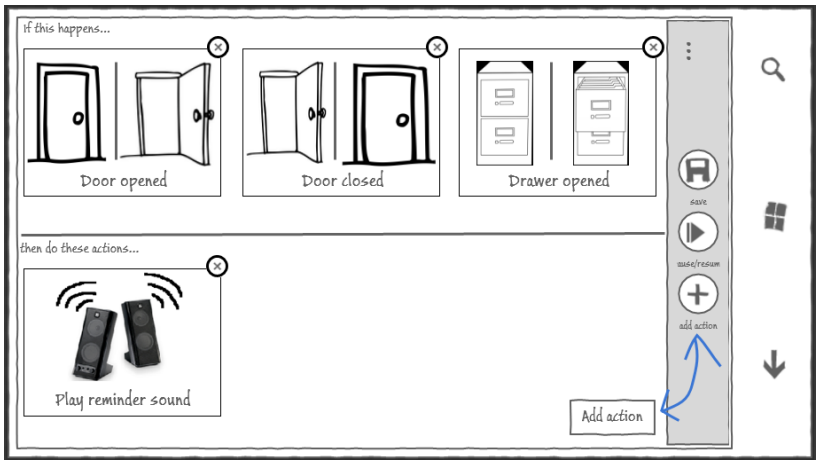


Figure 4. Sketch of the comic strip-based demonstration screen

These interactive prototypes were shown to some members of the Motivational Environments research group for feedback and underwent a couple of refining iterations.

The final user experience design for GALLAG Strip consists of three main components: physical interaction sensing, a mobile PWD user interface and user customization.

PHYSICAL INTERACTION SENSING

An important part of GALLAG Strip is its ability to sense user interactions with objects and spaces. Users are expected to interact with sensed objects and spaces both when programming their GALLAG application, as well as when “running” or testing the application.

Currently, GALLAG Strip supports four types of sensors: X10 [39] open/closed magnetic sensors, X10 motion sensors, Insteon [40] LampLinc modules and Insteon SynchroLinc modules. The X10 open/closed sensor (also called Door/Window sensor) is probably the most versatile, as it communicates through radio-frequency (RF) and can be installed on a wide variety of objects, like drawers, doors, or any other object for which we can change its position or location (see figure 5).

Figure 5. X10 magnetic sensors attached to drawer and remote

X10 motion sensors (MS16A and MS14A) also communicate through RF and they detect motion through changes in infrared heat; therefore, they are used to detect presence (or lack of) in a specific space. They are also very useful in situations where you do not want to attach an open/closed sensor to an object, like detecting when you reach to grab a book or an object beneath a table (see figure 6).

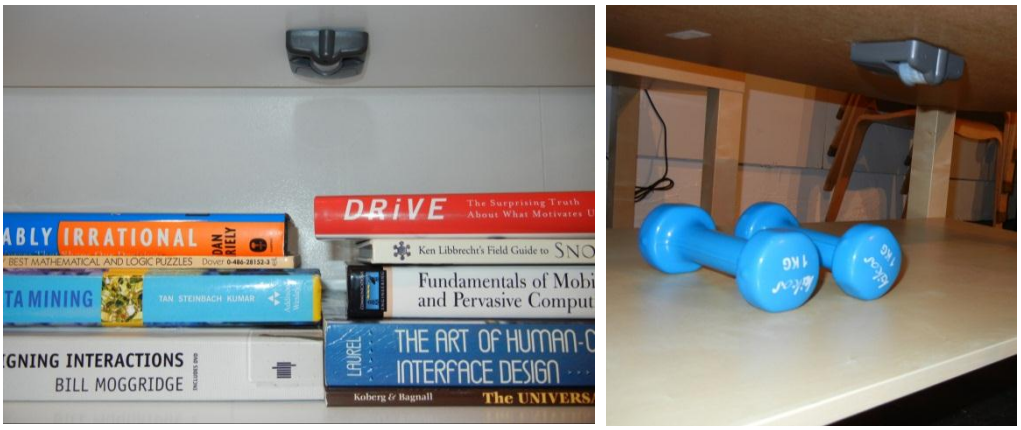


Figure 6. X10 motion sensors on bookshelf and table

Insteon LampLinc modules (see figure 7) allow both detecting when a lamp is manually turned on/off, and turning it on/off via a command sent through the AC power lines. Insteon SynchroLinc modules are designed to be used with bigger appliances than lamps (e.g., TV, washing machine) and are similar to LampLinc modules in the sense that they

can detect an appliance being manually turned on/off, however they cannot receive commands to turn the appliance on/off.

Figure 7. LampLinc and SynchroLinc modules

MOBILE PWD USER INTERFACE

The most important feature of GALLAG Strip is its mobile PWD user interface, which enables users to create, review and edit their GALLAG applications. By being a mobile programming interface, it allows users to move freely through the sensed space and to demonstrate their actions in a natural way.

GALLAG Strip's mobile PWD interface has three core elements, the main screen, the demonstration screen and the application frames.

The main screen is what the user is shown when the mobile application starts. If there are no previously saved GALLAG applications, the main screen will show the information page, where the user can learn how to create a new application, learn about the different application frames that he/she will be able to use, go to GALLAG's website for more information and get general details about the GALLAG Strip mobile application. If GALLAG applications have been previously saved, the system application list page is shown where the user can see which applications are currently enabled and disabled (see figure 8).

Figure 8. Main and application list screens

The demonstration screen is where users demonstrate what they want to program. It has two modes: a recording mode and an edit mode. In recording mode, the system listens for sensor events triggered by user actions and it appends them at the end of the application (see figure 9).

When the user touches the pause button while in recording mode, the demonstration screen goes into edit mode. In this mode, the user can review the application being created by scrolling up and down, and can also edit it by modifying or deleting existing frames and by adding new ones (see figures 10 and 11).

Figure 9. Adding action frame in recording mode

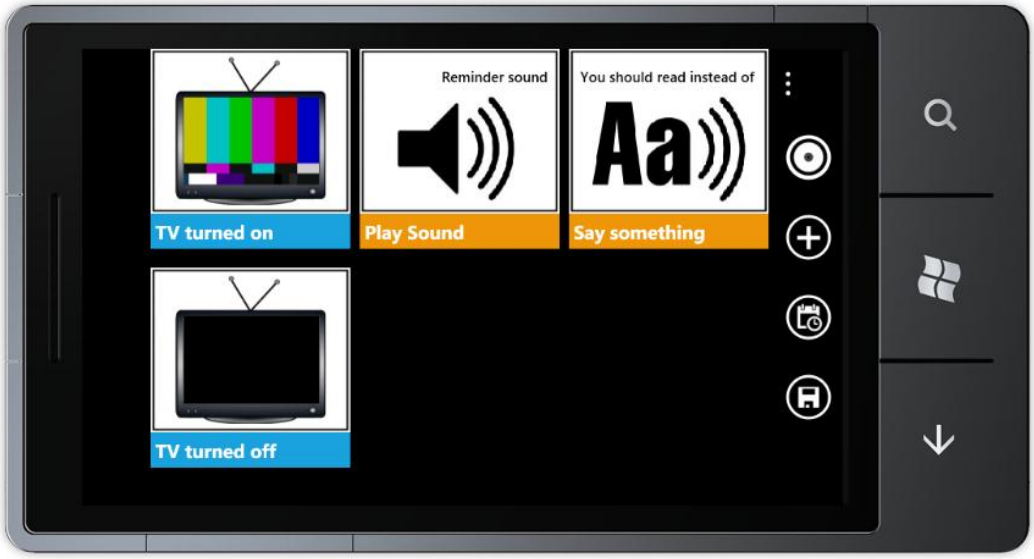


Figure 10. Demonstration screen while in edit mode

Figure 11. Adding a response frame to the application strip

When the user finishes creating his/her GALLAG application, he/she then touches the save button, the application information is sent to the server and the system configures itself to do what the user just programmed. After the application has been saved, the application is ready to be run and can be tested simply by performing the actions previously defined in the application.

A GALLAG application is represented in the demonstration screen through a sequence of application frames. This sequence of frames is called the application strip and can have three types of frames: action, response and time-date.

Action frames represent the user's actions within the sensed space and are shown as blue frames in the application strip. These frames have a default text label and image depending on the type of sensor. Actions involving open/closed and motion sensors have a description based on the sensor ID and type of action, as well a generic image depending on the type of sensor. Actions related with the LampLinc module show a text label referring to the lamp being turned on/off and have a generic image of a light bulb.

Finally, actions associated with the SynchroLinc module show a text label referring to the TV being turned on/off and have a generic image of a TV (see figure 12).

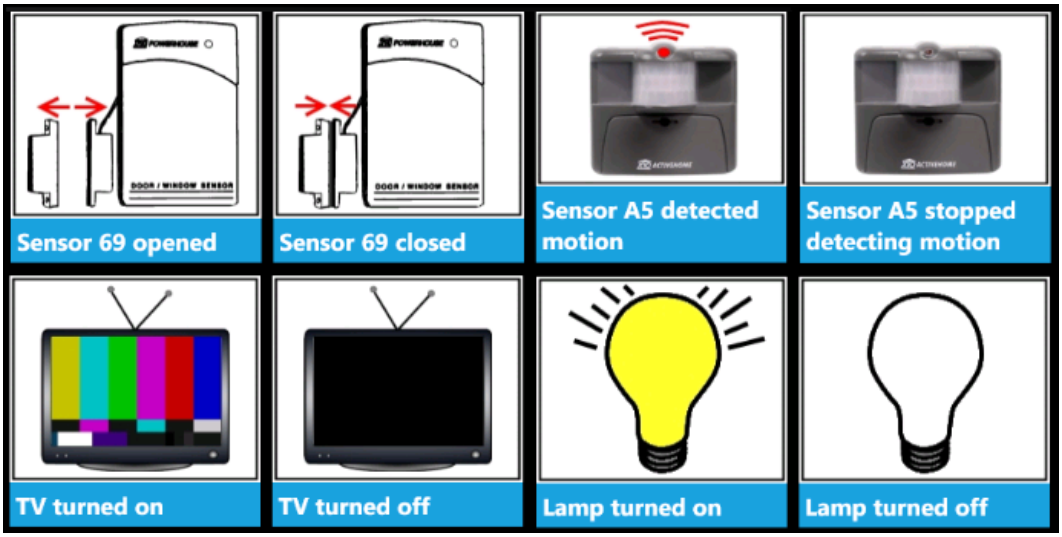


Figure 12. Action frames with default text label and image

Response frames represent actions that the system will perform and are set by the user. This type of frames is shown in orange and have a text label and image related to the type of response selected. Response frames can also have an additional parameter that is displayed in text above the frame's image (see figure 13).

Figure 13. Response frames

Time-date frames are conditions set by the user and they constrain the application's execution to a particular time and/or date. These frames are shown in green and show the selected date or time as their text label. Time frames additionally have a parameter to show the selected days of the week (see figure 14). Time and date frames can be combined to create conditions based both on a date and a time; that is, an application can have up to two, date and time frames.

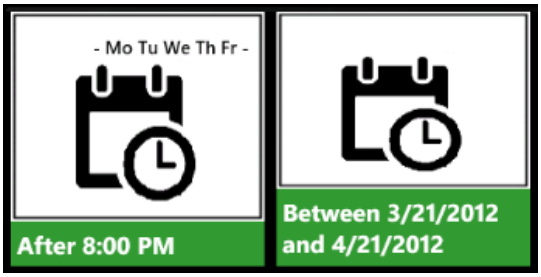


Figure 14. Time and date frames

The programming model for GALLAG Strip is a linear, if-then, rule-based one, and it is read from left to right and from top to bottom. This means that preceding actions or responses need to occur in the same sequence as they appear before the current one

can execute. Similarly, the time and/or date condition frames need to evaluate to true in order for the rest of the application to execute.

USER CUSTOMIZATION

User customization is an important feature that allows users to define the relationships between their actions and the sensors. As mentioned above, action frames are displayed with a default text label and image depending on the sensor being activated. Users can customize them by changing the text and by taking a picture with the phone's built-in camera (see figure 15, 16 and 17).

The ability to customize action frames makes the GALLAG application being programmed much easier to read and to debug, and makes the programming process more enjoyable.

Figure 15. Customizing text and image of action frame

Figure 16. Taking a picture with the phone's camera

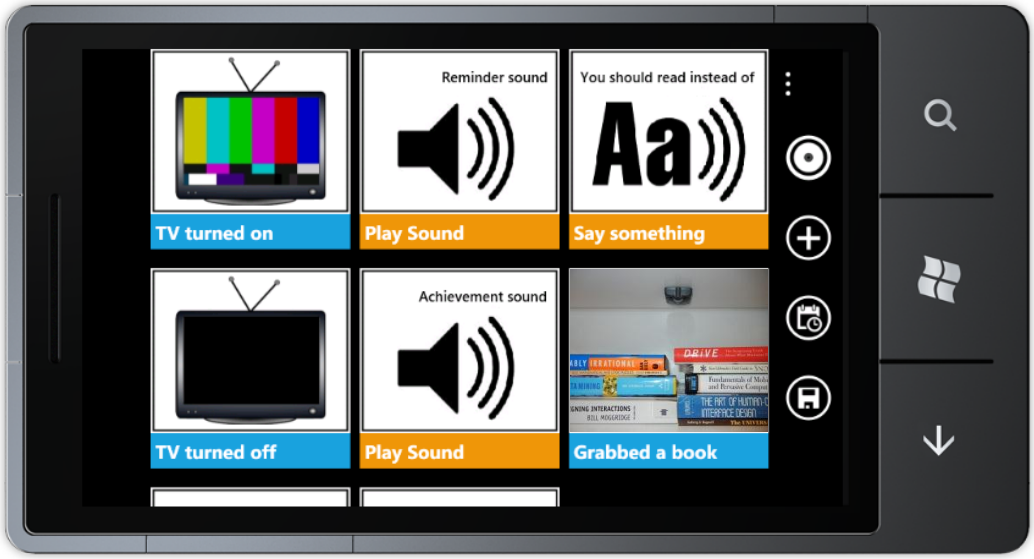


Figure 17. Action frame after being customized

ENABLING, DISABLING, AND RESETTING APPLICATIONS

GALLAG Strip also lets users disable, enable and reset a GALLAG application. Because the system allows users to have one or more active applications at the same time, the ability to enable and disable applications was implemented to avoid the need of deleting

a certain application that the user do not wants to be active at the moment. Additionally, because the current implementation of GALLAG Strip does not provide feedback as to the current state of an application, users can reset a particular application to its initial state (see figure 18).

Figure 18. Editing an application: enabling, disabling and resetting

This combination of physical sensing and a user-customizable mobile PWD interface is what comprises the user experience for GALLAG Strip. The following section describes the implementation details.

Technical implementation

Several technical challenges were faced during the implementation of the aforementioned user experience. Some of these challenges and some of the implementation decisions are discussed in this section.

The system has three main components: the Indigo server, the GALLAG Strip server application and the GALLAG Strip mobile application, the last two being the contributions of this work. Figure 19 shows these components and how they communicate with each other.

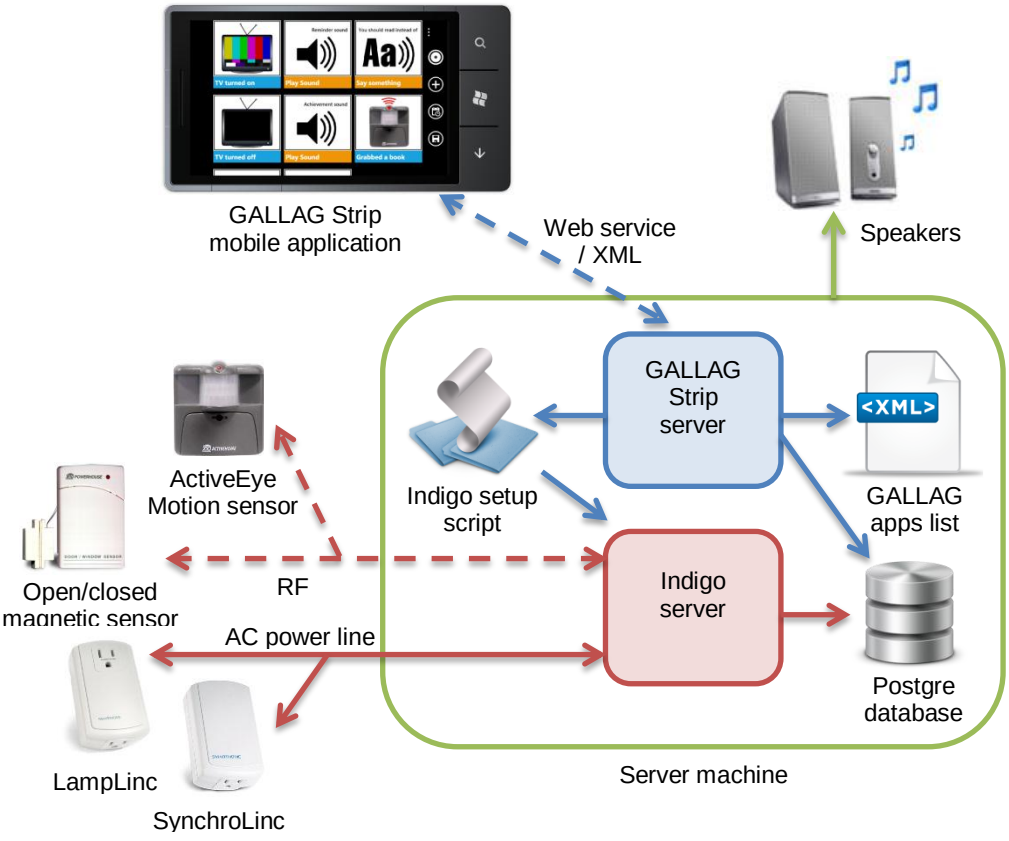


Figure 19. GALLAG Strip system components and connections

INDIGO SERVER

Perceptive Automation's Indigo [41] is a commercial home automation software that runs on Apple's OSX and is used both for communication with X10 and Insteon hardware, as well as a running platform for GALLAG applications.

Indigo receives information from X10 and Insteon devices (e.g., open/closed magnetic sensors, motion sensors, LampLinc and SynchroLinc modules) either through RF or through AC power lines, and can send commands to these devices using these same channels.

In addition, Indigo is capable of running AppleScript code based on configurable events, making it the backbone for GALLAG application execution, as it can continuously monitor X10 and Insteon devices and execute what the user defined in their application using GALLAG Strip.

Indigo also has the ability to log event information from devices into a SQLite or PostgreSQL database, a feature used by the GALLAG Strip server to provide sensor event information to the mobile application, which is explained in further detail below.

GALLAG STRIP SERVER

The GALLAG Strip server was developed as a console C# application and has three main features: serve as a sensor event provider to the mobile application, provide the ability to read, update and delete the list of saved GALLAG applications, and to create the Indigo setup script for a particular GALLAG application. The class diagrams for the server application, exposed web service methods, event class, event and device type enumerations are shown in figure 20.

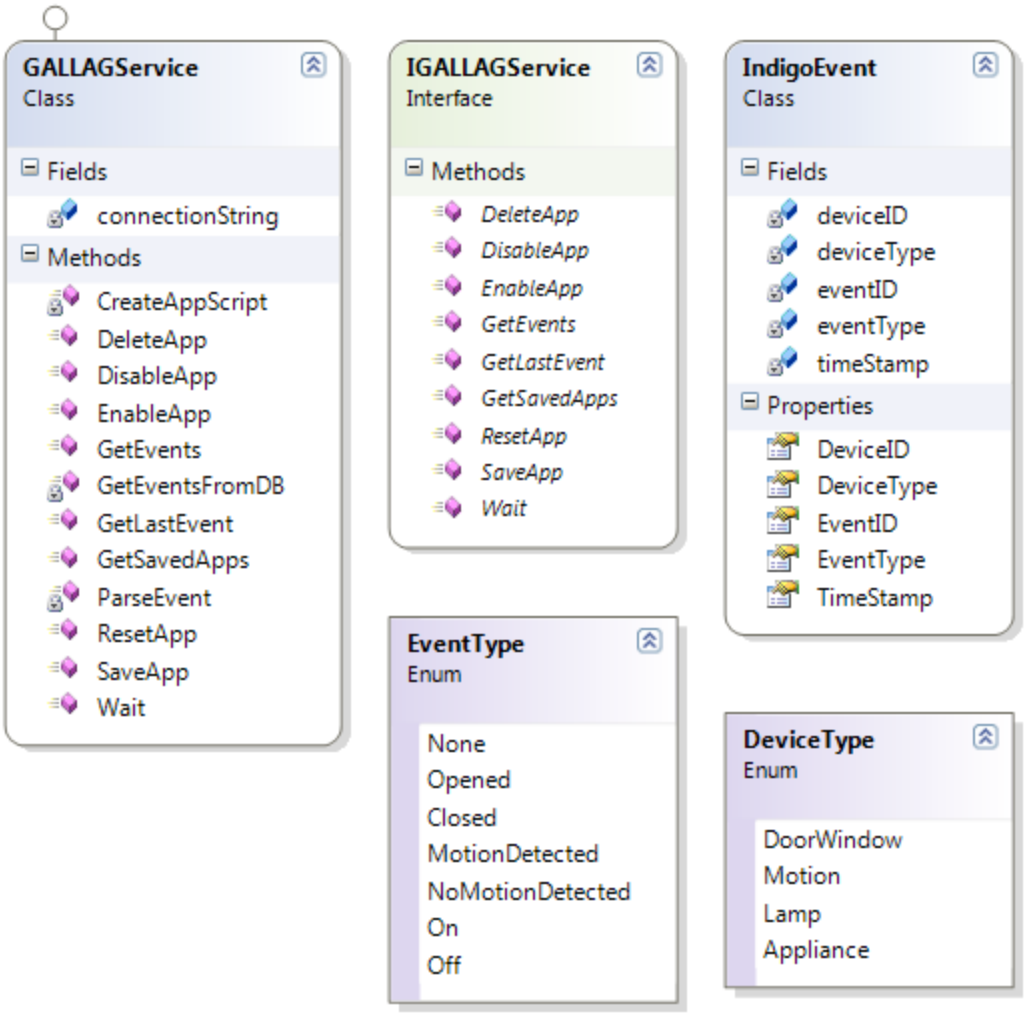


Figure 20. Class diagrams and enumerations for the GALLAG Strip server application.

The GALLAG Strip server provides the mobile application with information about the events detected by Indigo through the exposed web service methods (see IGALLAGService interface in figure 20). When the user touches the record button on the mobile application, the phone sends a request to the GALLAG Strip server to get the latest X10, RF and Insteon device events using the GetEvents web service method. The server application then connects to Indigo's PostgreSQL database, gets the latest logged

device event information and responds the web service call with a list of device events in XML format.

Another feature of the GALLAG Strip server is to facilitate basic persistent storage functions for the list of GALLAG applications, that is, it provides a way to read, update and delete a GALLAG application from the list of saved applications. This is also done through web service methods and the list of saved GALLAG applications is stored as an XML file on the server machine (see appendix A for an example of a saved GALLAG application XML structure).

When the GALLAG Strip server receives a web service request to save a GALLAG application from the mobile device, it creates the AppleScript setup file that will be run to configure Indigo. This setup script is created with the help of the GALLAG API and it is then saved and run on the server machine using Apple's OSX Folder Actions. The setup script file is then run and creates the necessary variables, triggers and action groups to do what was defined in the GALLAG application. After Indigo is configured, the GALLAG application can be tested.

The previous section explained how GALLAG Strip server lets users disable and enable a GALLAG application. When a user disables an application, the server creates and runs an AppleScript file to disable all triggers in Indigo for that particular application. Similarly, the server enables all the related triggers when an application is enabled.

The execution state of a previously saved GALLAG application is managed through state variables in Indigo. When a user wants to reset a certain application, the GALLAG Strip server creates and runs an AppleScript file to reset all the state variables in Indigo for that application.

The choice of using C# to develop the GALLAG Strip server application was based on my experience with the programming language. The decision of using XML-based web

service methods for communication was based on the advantage of having a platform and language-free that can enable the implementation of the mobile application on a different mobile operating system (e.g., Android [42], iOS).

GALLAG STRIP MOBILE APPLICATION

As mentioned in the previous chapter, it was decided to develop the user interface on a mobile platform; this makes it easier for users to experience the demonstration process, as they need to physically move around and interact with the sensors to demonstrate their application. The mobile application was developed to run on the Windows Phone [43] mobile operating system (version 7.5 Mango) and was programmed using the C# language, this again because of my experience with C# and the .Net framework. Figure 21 shows a class diagram and the enumerations used for the Windows Phone application.

All of the communication between the mobile application and the rest of the system is done through web service calls. Additionally, the only information stored on the mobile application related to GALLAG applications are the customized text labels and images for the action frames, the rest of the information about the GALLAG applications is stored in an XML file on the server machine, as explained in the previous section.

The mobile application was designed with future improvements in mind. Separate classes were created for the application frames, the supported sensors and devices, the predefined sounds and music streams, and the available phone carriers among others. Figure 21 shows the classes and enumerations used on the GALLAG Strip mobile application (see appendix B for more details on the main classes and enumerations).

Figure 21. Class diagrams and enumerations for the GALLAG Strip mobile application

CONCURRENCY AND RESOURCE CONFILCTS

Currently, GALLAG Strip only supports one user, meaning that conflicts could occur when having multiple concurrent users. Design for multiple users was out of the scope of the initial implementation of GALLAG Strip, since a method for user identification (e.g., facial recognition, RFID, location tracking) is necessary to create multiple-user applications.

Additionally, there could be resource (e.g., sensors, devices, sound, and music) conflicts when having multiple active GALLAG applications. To illustrate a conflict with using the same sensor, imagine that a user creates an application that will detect if the front door of the house is opened when arriving home and play a welcome sound afterwards. The user then creates another application that will detect if he/she turns the AC on and then opens the front door to leave the house; if this happens, he/she wants the application to play a reminder sound. If both applications are saved and enabled, when the user turns on the AC and then opens the door, both applications will be activated and both sounds will be played consecutively, with one of them only being played a fraction of a second. One way to avoid this particular conflict could be by using a motion sensor outside of the house to detect the user entering his/her home. Other ways to avoid these resource conflicts is for users to restrict the execution of their applications to a certain time or date using time-date conditions, and also to enable or disable applications according to their current needs. I acknowledge that this does not solve the problem completely, but it helps to avoid many of these conflicts.

Before testing the system with users, an examination of GALLAG Strip by applying the Cognitive Dimensions of Notations (CDN) framework [2] was performed to analyze its usability and design tradeoffs as a PWD environment.

Cognitive dimensions usability analysis

The purpose of using the CDN framework is to perform a quick, low cost examination by experts of potential usability problems before running a more expensive user study.

The CDN framework is one of several techniques used in the Human Computer Interaction (HCI) community, which provides a broad-brush evaluation of a system's form and structure [21]. It was conceived as a framework to describe notational systems and information artifacts, considering in particular the novel graphical interfaces and visual grammars of complex products like visual programming languages and ubiquitous computing platforms [3], making it a suitable evaluation method for sensor-based and visual programming projects [13].

The CDN framework is similar to heuristic evaluation methods like Jakob Nielsen's list of user interface heuristics [25,26] in the sense that it provides a list of usability issues. However, CDN also provides a vocabulary that designers can use to discuss the effects of their design. These effects are normally expressed as trade-offs between dimensions, as an adjustment in a dimension might affect one or more dimensions of a notation or artifact [3,15].

Procedure

To perform the design examination of GALLAG Strip, the CDN questionnaire by Green and Blackwell [2] was followed. The activities allowed by GALLAG Strip's mobile application were taken into account while answering the CDN questionnaire:

- Addition. Adding further information without altering the structure in any way.

- Modification. Changing the existing structure, possibly without adding new content.
- Exploratory design. Combining addition and modification, with the distinction that the desired end state is not known in advance.

These activities were then analyzed across the 13 cognitive dimensions (see appendix C for details about the CDN questionnaire that was used).

Results

The examination starts by defining the main notation and estimating the time spent while performing the task or activity being examined, which in this case is creating a GALLAG application using GALLAG Strip.

PARTS OF THE SYSTEM

GALLAG Strip's main notation is a visual representation of sensor events with user-defined responses and conditions, that is, the application strip. After a few trials with the system, the following are the time estimates while creating GALLAG applications:

5% Searching for information within the notation (visually examining the application strip, reading text labels on application frames)

20% Translating substantial amounts of information from some other source into the system (demonstration, translating physical interactions into action frames)

25% Adding small bits of information to a description that you previously created (adding response and time-date frames, changing text label and image to frames)

20% Reorganizing and restructuring descriptions that you have previously created (deleting application frames to add new ones)

30% Playing around with new ideas in the notation, without being sure what will result (exploration, adding application frames and possibly testing them)

These time estimates can be compared to the ones reported in the evaluation of Exemplar, a PBD tool for rapid sensor-based interaction prototyping [13]. Because Exemplar requires users to read live sensor data (i.e., sensor data graphs), they report that users spend 30% of the time searching for information within the main notation (analyzing sensor signals); in comparison, users in GALLAG Strip spend only around 5% of the time searching for information when analyzing the main notation (examining the application strip). They also show that users spend 10% of the time while demonstrating their application in their 2D desktop user interface; in GALLAG Strip, users spend around 20% of the time in the demonstration phase, as users need to physically move around to demonstrate their applications. The last major difference is the time needed to reorganize a previous demonstration, where they report that it takes users around 10% of the time to change analysis types and redefine events; in comparison, users in GALLAG Strip take around 20% of the time to reorganize their demonstrated application, this is likely because application frame reordering is not currently available. Finally, one thing in common between Exemplar and GALLAG Strip is that users spend a fair amount of time (30%) trying out new ideas without knowing what will be the end result, which highlights the importance of exploration for users in both systems.

SUB-DEVICES

Helper and redefinition devices are system sub-devices that help with a specific part of the activity. In GALLAG Strip, the X10 and Insteon sensors were considered helper devices, and the phone's camera was considered a redefinition device.

DIMENSIONS OF THE MAIN NOTATION

1. Viscosity (ease or difficulty of editing previous work). It was easy to change parameters or details to previously added response frames, as well as deleting unwanted frames.
2. Visibility (ability to view components easily). The elements of the notation (i.e., application strip) were easy to view, with the only limitation being that if a GALLAG application requires more than six frames, scrolling is needed to view the entire application strip.
3. Premature commitment (constraints in the order of doing things). Due to the sequential nature of the current version of GALLAG Strip, the level of premature commitment is quite high; if you decide to insert a frame in the middle or beginning of the application, it requires you to first delete all subsequent frames, as new action and response frames are always added at the end of the application strip. On the other hand, the ability to delete frames decreases the cost of commitment in some way. Nonetheless, adding a feature to reorder frames would eliminate this problem.
4. Hidden dependencies (important links between entities are not visible). The only dependency difficult to see is that the X10/Insteon sensors and devices used were tightly coupled to the system, meaning that they were previously configured and that adding a new one requires updating the system.
5. Role-expressiveness (purpose of a component is easily inferred). This was not a particular concern with GALLAG Strip, as all components are easy to understand.
6. Error-proneness (the notation invites mistakes and the system give little protection). The inability to reorder application frames was the only type of error that occurred repeatedly, which was a significant problem with long applications. Again, adding the feature to reorder frames should alleviate this issue. Other types of errors were due to the lack of enough user input validations on text fields.

7. Abstraction. The level of abstraction can be considerable if you do not customize the action frames, as you are translating physical actions into a sequential strip of frames; however, the ability to customize the text labels and images on the application frames lowers the level of abstraction substantially and makes it much more understandable.
8. Closeness of mapping (closeness of representation to domain). The notation (i.e., the application strip) is directly mapped to the created application, as the application will be executed in the same sequence as it was demonstrated.
9. Consistency (similar semantics are expressed in similar syntactic forms). The only consistency issue found was that the time-date frames are always at the beginning of the application, in contrast with the action and response frames which are always added at the end. However, because timers are not currently supported, having time-date frames in the middle would not make more sense than at the beginning of the application, so this should not be a significant issue.
10. Diffuseness (verbosity of language). Applications created with GALLAG Strip are fairly succinct; the notation directly represents the actions and responses that you want to program.
11. Hard mental operations. The biggest mental effort needed while creating an application occurs when adding time-date frames, as you need to make decisions on whether to add a time or date frame, a combination of both, and what value or range of values to select for the date and/or time. However, this is not considered a hard mental operation.
12. Provisionality (degree of commitment to actions). Because you are able to edit your application before saving it and because you can test your application before finishing it, the level of commitment is low and it enables you to try out new ideas.

13. Progressive evaluation (work-to-date can be checked at any time). GALLAG Strip allows to easily review the current state of the application being programmed; if the application is long, it can be reviewed by scrolling the application strip up and down.

In summary, GALLAG Strip performs well regarding viscosity, abstraction, role-expressiveness, closeness of mapping, consistency, diffuseness, hard mental operations, provisionality and progressive evaluation. The issues found with visibility, premature commitment, hidden dependencies and error-proneness are related to the fact that GALLAG Strip currently does not support reordering of frames, that the amount of application frames visible at the same time is limited (i.e., up to six frames) to the size of the phone's screen, that not enough validations exist for user input and that the devices and sensors used are tightly coupled (i.e., preconfigured) with the system. Another issue found after performing the examination is that the fact that GALLAG Strip only allowed for one active application can be quite limiting; having multiple active applications can enable users to create a much more complex application based on two or more simpler applications.

Based on these results, changes to GALLAG Strip's design were made to improve it in the areas that were identified as most problematic and that were able to be implemented within time constraints. The changes made were two, more user input validations and being able to have more than one active GALLAG application at the same time.

Also relevant to note is that, in addition to being able to spot deficiencies in our initial design, this examination will facilitate the comparison of the current version of GALLAG Strip with future ones, and with other systems.

EXPLORATORY USER SESSIONS

After updating GALLAG Strip according to what was found on the CDN examination, a set of exploratory user sessions were performed to determine the system's intuitiveness, to get user feedback and to update the system accordingly before running the final user study.

The thinking aloud method was used to get a better sense of what users think while using the system. Think aloud sessions are a well-known usability-engineering method that helps to understand how users view a system and to identify any major end user misconception [14].

These sessions took place in the Motivational Environments laboratory at Arizona State University and had the participation of four users. None of them knew about the project, two had prior programming experience and the other two did not.

Procedure

At the beginning of the session, a very short description of the system was given. They were just told that the system enabled them to program sensor-based applications with the use of a mobile phone; they were then told about all the sensors that they could interact with.

After the introduction, participants were asked to program a simple, previously defined scenario and to think aloud while doing it. They were also explained that my role would be of active listener, making it clear that the objective was to evaluate GALLAG Strip, not them. While the session was in course, participants were reminded to continue thinking aloud with simple acknowledgement tokens throughout the programming task to avoid any additional cognitive load, as proposed by Boren and Ramey [4].

Finally, they were asked to try to complete the programming task on their own and figure out how to use the system themselves, although they were told that they would receive help if they could not continue at all.

The following is the text description of the task that we asked them to program:

if you enter the living room then

play alarm sound

end if

Results

Some issues with GALLAG Strip were raised by the participants and some of them were selected to improve upon before continuing with the final user study.

Table 2 shows the things that participants pointed out, their occurrence and the action taken.

Occurrence	Comment	Action taken
2	Want to have music playing as a response option.	Response added
2	Play button is not intuitive to add user actions.	Changed button
2	Improve help messages in demonstration screen.	Improved messages
2	Add sensor to digital portrait, TV remote control and AC thermostat.	Sensors added
2	Difference between events and actions confusing.	Changed naming
2	Want to reorder frames in application strip.	None
1	Add timer functionality.	None
1	Ability to duplicate a saved application.	None
1	Ability for system to turn on/off the AC and TV.	None
1	Have feedback when testing a saved application.	None

Table 2. Feedback gathered from exploratory user sessions

From the user comments and changes made on the above table, it is relevant to note that the version of GALLAG Strip used in these sessions referred to user actions as events (e.g., event frames) and system responses as actions (e.g., action frames), we changed the naming to be more user-friendly, therefore “event” frames where changed to “action” frames and “action” frames where changed to “response” frames. Also, the button to add user actions was changed from a play button to a record button, making it clearer that the system was in fact recording user’s actions.

The functionality to reorder frames, to have timers, to duplicate applications and to have feedback when testing a saved application was not implemented due to their complexity and time constraints. The ability to turn on/off the AC and TV was not possible at the time due to Insteon hardware limitations.

VALIDATION: USER STUDY

The preceding chapters described the motivations, design, implementation, usability examination and preliminary user testing. Although this iterative process yielded a refined system to some extent, it was still necessary to determine how effective it is as a PWD environment for novice GALLAG users. This chapter presents the procedure followed to validate GALLAG Strip by answering the following questions:

- Can users with none or little programming experience use GALLAG Strip to create sensor-based context-aware applications easily and accurately?
- Does GALLAG Strip support the creation of the majority of the applications envisioned by users?

A user study was conducted in a setting that closely resembles a living room and a post-session questionnaire was administered to get the users' overall feedback on their experience. The study took place at the Motivational Environments laboratory at Arizona State University over a period of 6 days and the sessions were video-recorded using a head-mounted camera wore by the participants.

Subjects

A total of 13 subjects volunteered to participate in this study. They were recruited through flyers posted around campus through specific email distribution lists at Arizona State University. Ages ranged from 21 to 49, six male and seven female, four undergraduates and nine graduates. Participants had a variety of backgrounds: educational technology, distance education, graphic design, architecture, civil engineering, material science, computer science, electrical engineering and chemical engineering.

Participants were required to have prior exposure to smartphones and none or little programming experience. However, four of them actually had some, considerable or extensive programming experience (see figure 19); although they did not meet the initial

requirement, it was decided to include them in the study due to our difficulty in getting enough participants and because their performance was almost the same as the participants with none or little programming experience.

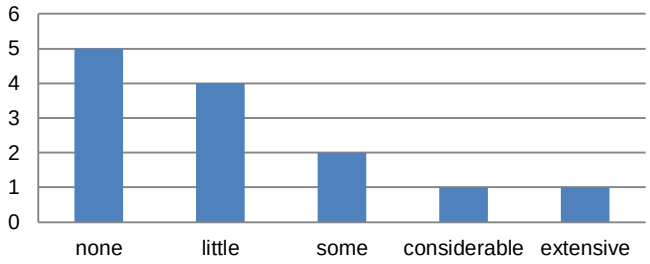


Figure 22. Programming experience level for user study participants

In addition to the 13 study participants, two members of the Motivational Environments research group volunteered to participate as GALLAG experts. They helped with the gathering of benchmark data to compare the performance between novice GALLAG users creating applications with GALLAG Strip and expert GALLAG users programming the same applications through the use of AppleScript code and Indigo elements.

Setting

As mentioned earlier, the sessions were held in a laboratory that simulates a common scenario of a living room, with sensors placed around the space. In particular, X10 magnetic sensors were placed on: the laboratory's front door, the TV remote, a plant's vase, two drawers, a digital portrait and the AC's thermostat. X10 motion sensors were placed: on top of the TV to detected presence in the living room, below the table to detect if the user reached for the dumbbells and on the bookshelf to detect if the user reached for a book. The lamp was connected to an Insteon LampLinc module to be able to sense and control it turning on/off. The TV was connected to an Insteon SynchroLinc module to sense when it was turned on/off (see figures 20 and 21).

Figure 23. Living room setting and placement of sensors (yellow circles)

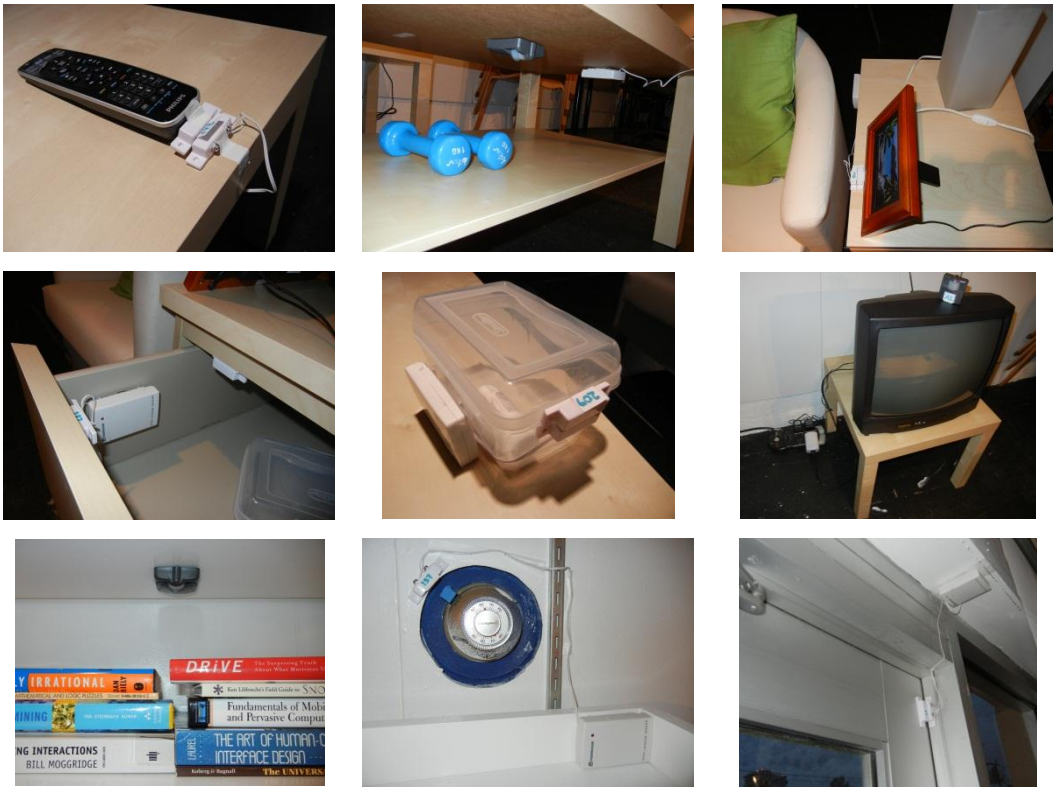


Figure 24. Placement of sensors for user study

Application categories

For this study's purpose, applications were divided into two categories: simple and complex. A simple application consists of a single if-then rule with two conditions and two actions that will execute if the condition is met. A complex application consists of a date-time condition and two, nested if-then conditions with actions inside both of them (see table 3).

Category	Application

1 Simple	if you enter the living room and you turn the TV on then
	play reminder sound
	make system say "Remember to take pills"
	end if

2 Simple	if you open the top drawer and you open the box with pills then
	play achievement sound
	send an email to let someone know that you just took your pills
	end if

3 Complex	if time is after 9am and you turn the AC on and you open the front door then
	play alarm sound
	make system say "Turn off the AC"
	if you close the front door then
	send an SMS to your phone to remind you to turn the AC off
	end if
	end if

Table 3. Application category examples

The rationale behind dividing applications in these two categories was to determine if GALLAG Strip is an effective PWD environment for both simple and complex applications.

Procedure

The individual, one-hour user sessions consisted of an initial tutorial and walk-through (approximately 20 minutes) about GALLAG Strip, followed by the application programming tasks. After completing the programming tasks, participants were then asked to fill an online questionnaire to obtain feedback about their overall experience with the system.

During the study, the measures evaluated were the following:

- Completeness. The number of implemented application requirements.
- Accuracy. The number of errors in the implemented application (deviations from what was expected, missing requirements are not counted here).
- Duration. The time in minutes required to program the application.

TUTORIAL

The first part of the session was a tutorial to introduce the participants to GALLAG Strip's capabilities, components and user interface.

The tutorial was provided verbally and a script was followed to be consistent with all participants. It started with an introduction to what GALLAG Strip is about and continued with a demonstration of how to program a GALLAG application. The demonstration showed how to program an application that would do the following:

```
if you enter the laboratory then
        play music
        if you enter the living room then
                stop audio
                turn on the lamp
        end if
end if
```

GALLAG Strip's user interface (i.e., the mobile application UI) was explained while demonstrating how to program the application. After finishing with the application demo, participants were told about all the available sensors in the living room that they were able to use and how they worked.

Finally, participants were reminded that they were not the subject being tested, but the system. Also, they were asked to talk aloud while programming their applications and that they might be reminded to do so if they went silent for a while.

PROGRAM GENERATION

After the tutorial, participants were asked to complete four programming tasks. The first three programming tasks are the same as the ones described in Table 3 above, two simple and one complex, and the last one was a free-form application with no restrictions other than a 15-minute time limit. After finishing each programming task, participants were encouraged to test them.

For the three first applications, a text description and a graphical representation were shown to the participants on a computer monitor, both representing the same application requirements. The graphical representation used a similar notation as the one they would see while programming it with GALLAG Strip (see figures 22, 23 and 24).

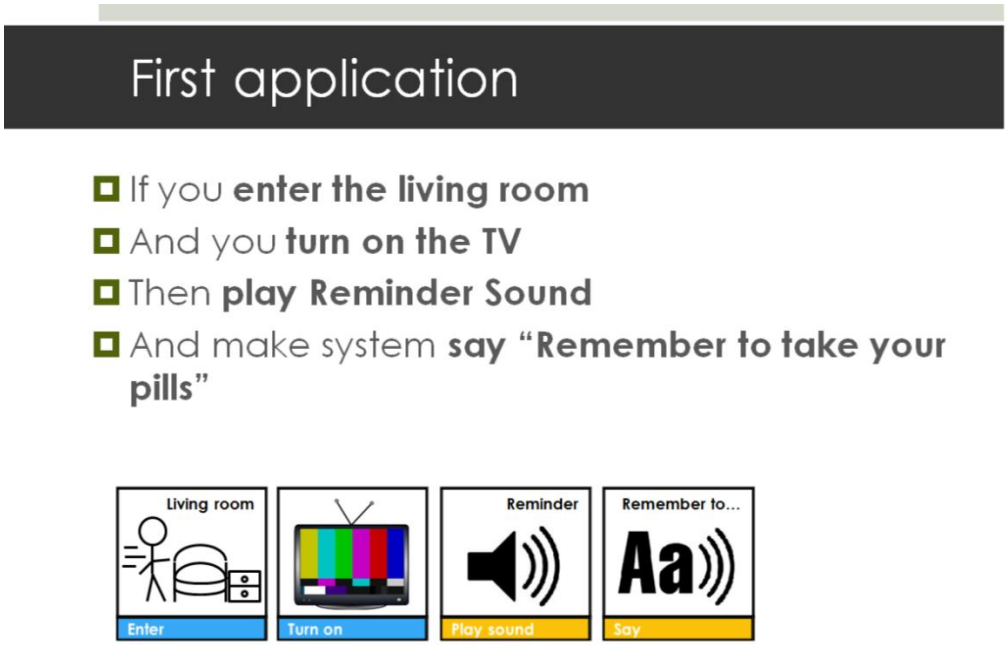


Figure 25. Application requirements for first task

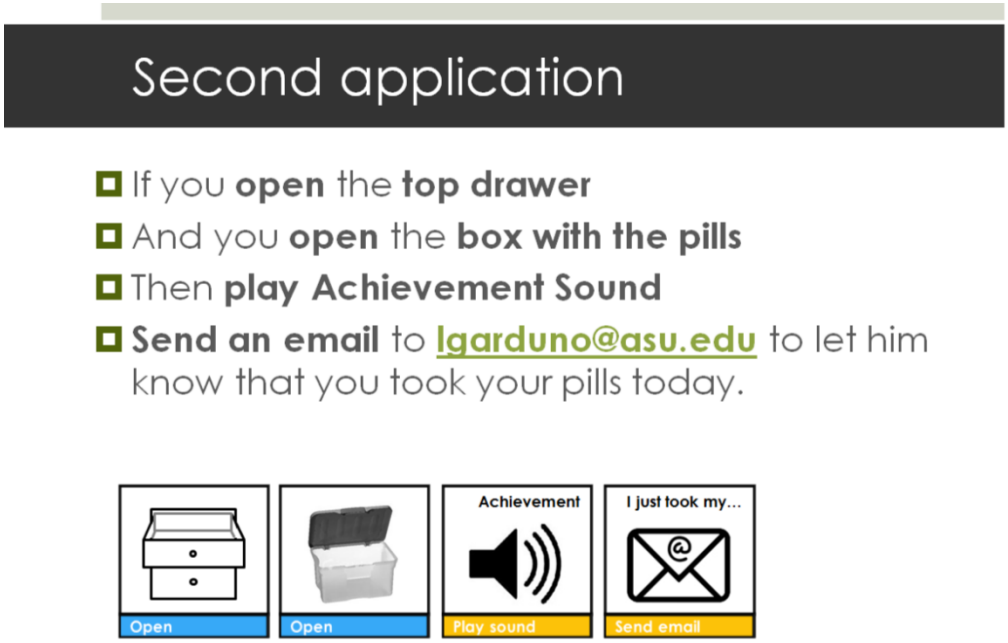


Figure 26. Application requirements for second task

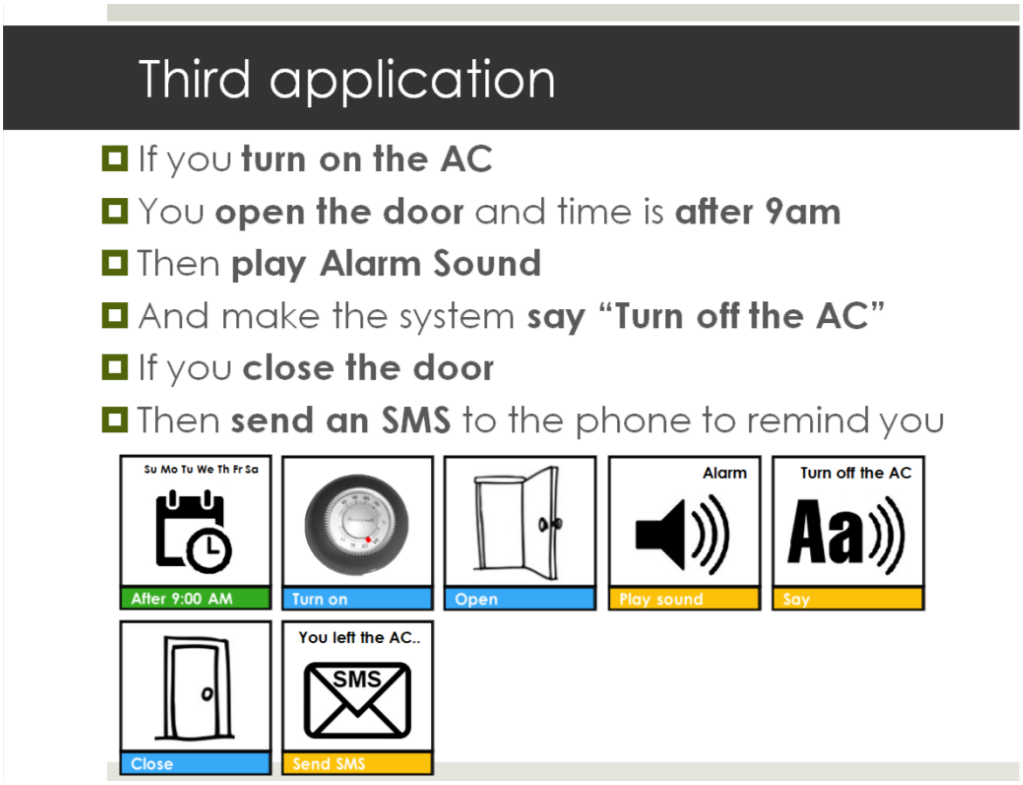


Figure 27. Application requirements for third task

The idea behind the first and second application is to test how much do participants improve when asked to program an application of the same complexity as the one they just programmed. Because of this, the first and second applications were intentionally alternated between participants.

For the last application, the time limited free-form, participants were first asked to think about an application they would like to have at home or that they were interested in programming. They were also asked to describe it verbally before starting to program it, so that it could be determined whether they were able to program all of their initial requirements or not.

Finally, the expert GALLAG users were shown where all the sensors were placed and were then asked to program the first three applications using AppleScript code and Indigo

elements, showing them the same textual and graphical application requirements as the rest of the study participants.

SURVEY

After the programming tasks, participants were asked to answer a web-based questionnaire about their experience with GALLAG Strip. The objective of the questionnaire was to get their subjective satisfaction and/or possible anxieties while using the system.

The questionnaire was based on the System Usability Scale (SUS) developed by John Brooke [5], which has been an industry standard for usability and has also been used in several research projects. SUS is a simple, ten-item questionnaire measuring usability in terms of effectiveness, efficiency and satisfaction. The possible answers are presented using a Likert scale with 5 possible answers ranging from “strongly disagree” to “strongly agree”.

In addition to the SUS standard questions, participants were asked if they felt limited by the kind of applications that they could develop with GALLAG Strip and if they felt that they needed to learn a considerable amount of things before being able to use it. They were also asked to describe things that they would like to see in a future version, and finally, questions to ascertain specific characteristics about them, like age, gender, educational background and programming experience. The following are the questions that were asked:

> Please answer the following questions based on the experience you have just had with GALLAG Strip:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.

4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I felt limited by the kind of applications that could be created.
11. I needed to learn a lot of things before I could get going with this system.
12. I think that I would be able to program the majority of the applications that I would like to build.
13. What did you like the most about the system?
14. What did you like the least about the system?
15. What would you like to see in a future version of the system?
16. Are you male or female?
17. What is your age?
18. What is the highest level of school you have completed or the highest degree you have received?
19. What is your major? (If applicable)
20. How much programming experience do you have?

Results

First, a within-subject comparison with the 13 study participants is performed. Not surprisingly, almost all were able to program all the requirements for the first three applications (see figure 25), with the exception of three participants that missed one requirement. However, these three participants mentioned that the reason was that they were not careful in reading the requirements in detail. For the free-form application, three

participants failed to implement one requirement each; one of them forgot a requirement that he described before implementing his application, another realized that he would have had to delete several frames in order to implement the requirement and decided not to do it, and the last one wanted to add a timer to his application, which is not currently supported.

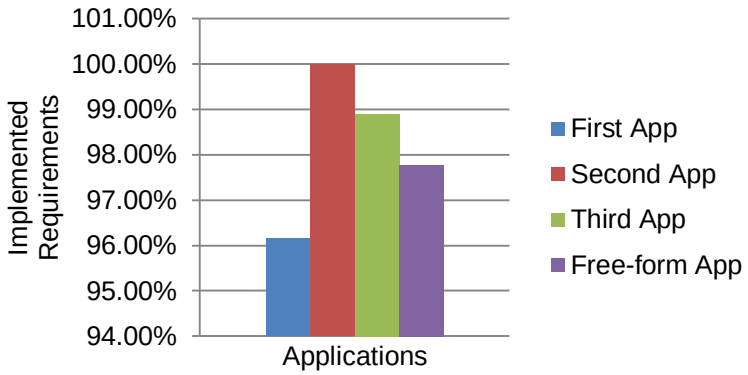


Figure 28. Percentage of implemented requirements for participants

Although it was expected to see a high value for application completeness, it is good to see that participants were also able to implement almost all of the requirements for their free-form applications.

Figure 26 shows the average number of requirements for each application, with the free-form application varying considerably as anticipated. It is important to mention that the complexity level for 12 of the 13 free-form applications was equal or higher than the third application.

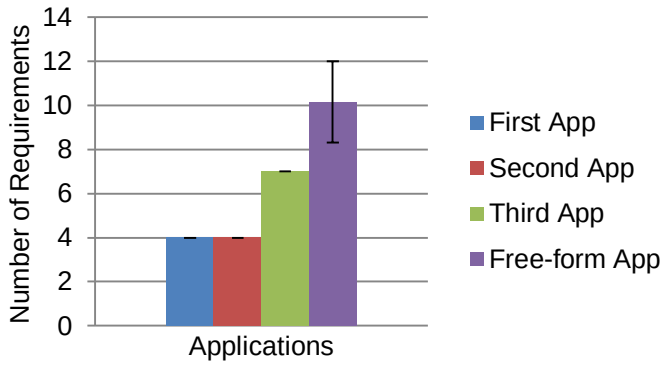


Figure 29. Average and std. dev. of requirements for participants

The average number of errors was low for all applications and went down after the first application (see figure 27). Similarly to the missing application requirements, most of the errors were due to lack of user attention to the requirements, as many of them selected the wrong sound to play or left additional action frames that were not needed. Two errors were caused by confusion with the action of a user turning the lamp on/off and a system response to turn it on/off. Lastly, one error was due to an unknown bug in the mobile application that caused a response frame not to be saved correctly.

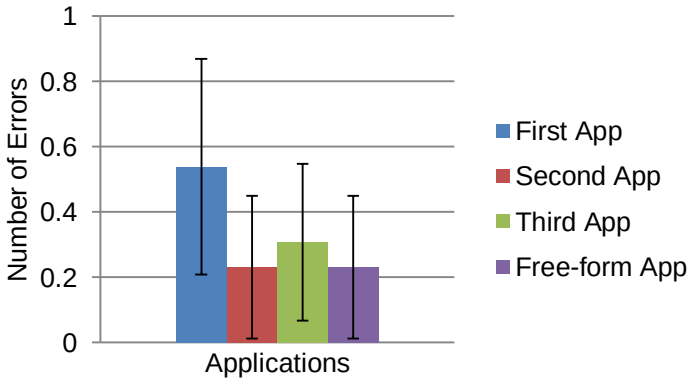


Figure 30. Average and std. dev. of errors for participants

All participants completed the applications in a reasonable amount of time and an improvement was evident for the second application (see figure 28). Another interesting finding is that, even though the average number of requirements (and complexity) for the free-form application was higher than the third one, the average time to program them was lower; this shows that GALLAG Strip has a low learning curve, as participants were able to program applications of increasing complexity in a shorter amount of time.

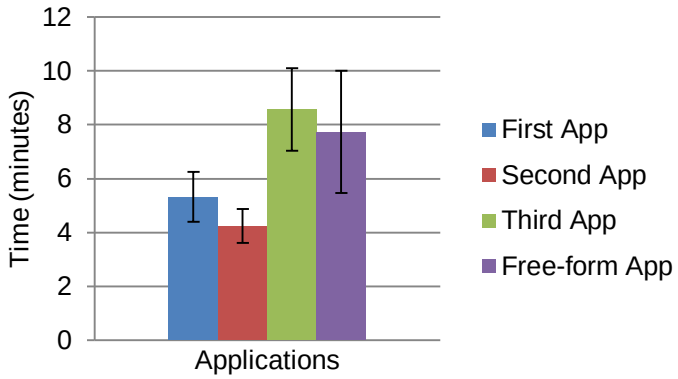


Figure 31. Average and std. dev. of time required for participants

Regarding the two expert GALLAG participants, both were able to program all the application requirements, except for the second application in which one expert was not able to use the TV because its setup was modified by an external individual without previous notice, thus only the data from one expert is available for the second application.

Although both had to debug their applications to make sure they worked, none of them had errors in their final applications.

About the time spent, also as expected, the second application took less time than the first (see figure 29); however, it is interesting to note that the average time they required for the third and more complex application was almost double (16.3 minutes) than the average time that the study participants needed (8.5 minutes). This is a positive finding, because it demonstrates that users without knowledge about how to program a GALLAG

application are able to program them with GALLAG Strip faster than expert GALLAG programmers using traditional tools.

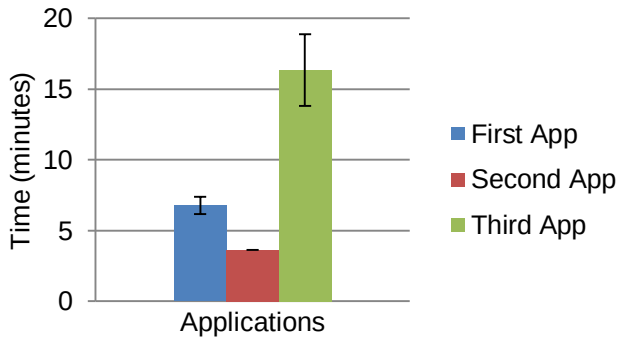


Figure 32. Average and std. dev. of requirements for experts

It is also important to highlight that the expert GALLAG participants had one or more years of exposure to the GALLAG system and were proficient in AppleScript programming.

SURVEY

In the post-session survey results (see figure 30), participants graded GALLAG Strip highly for its ease to use (average=4.45 on a 5-point Likert scale, σ=0.52), and for making them think that people would be able to learn how to use the system quickly (average=4.08, σ=0.51). Results were not as decisive (σ>1) on making them feel confident in using the system (average=4.0, σ=1.1); on making them feel that they would be able to program the majority of the applications that they would envision (average=4.0, σ=1.41) and on feeling limited by the kind of applications that could be created (average=3.1, σ=1.1).

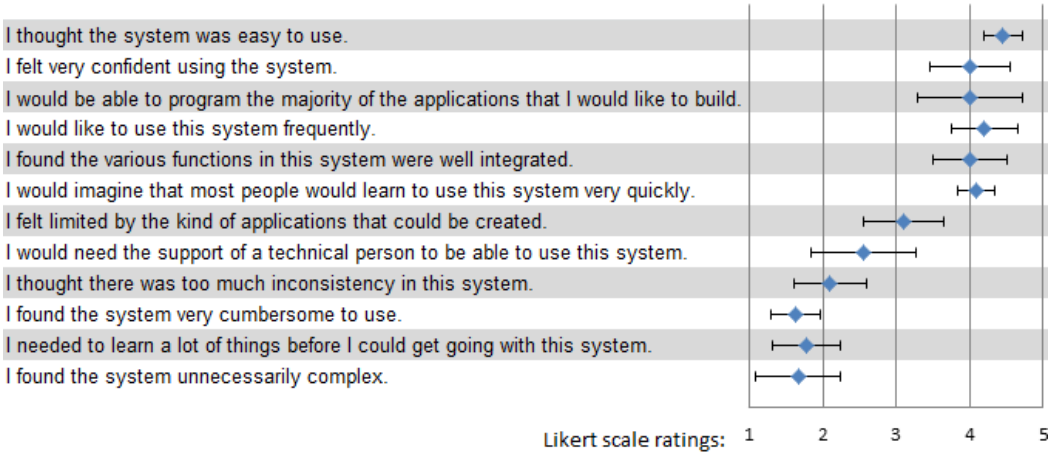


Figure 33. Post-session questionnaire showing average and std. dev.

Regarding the open questions, participants said the following about what they liked the most about GALLAG Strip:

> "Fun, intuitive, easy, connection to mail and SMS"

> "The fact that you can create your strip with out know nothing about how to program the sensors."

> "I liked that applications can be created easily."

The following are some responses about what participants liked the least about GALLAG Strip:

> "Hang in middle... not ease to swipe the activities order"

> "The fact that you can not re-order the frames on the strip and the fact that you have to remember the exact order of the frames to have the system reacts as expected."

> "I didn't like that frames cannot be reordered in a different way. Also, time and date frames cannot be placed either in the middle or at the end."

APPLICATION THEMES

Although it was not part of the initial research agenda, I was interested in determining what type of applications were users interested in programming with GALLAG Strip, as well as the system elements that were used the most. This was done after the user study by reviewing the video recorded from the sessions.

The verbal descriptions of the free-form applications given by the participants were analyzed and three major application themes were noticed: behavior change, home automation and social. Figure 31 shows the free-form applications classified under these themes, with some applications encompassing more than one theme.

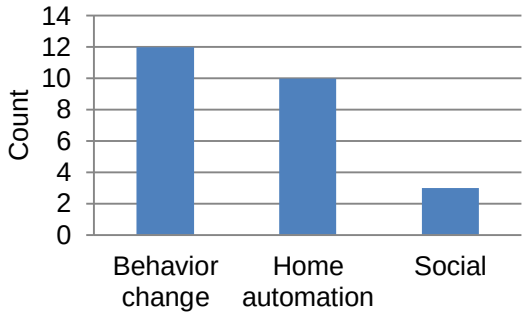


Figure 34. Major themes for free-form applications

From these three major themes, two were then subdivided into more specific ones. For the social theme, subthemes were observed for celebration, information sharing and play (see figure 32). The applications that fell under the social theme were probably the most creative: one participant created an application to cheer up her husband on his birthday by making the system say something, playing a sound, and making the lamp blink repeatedly; another participant programmed an application to make the teddy bear speak and invite her to watch TV together. Finally, a participant wanted to let a friend know that he was doing exercise, since they were competing to lose weight.

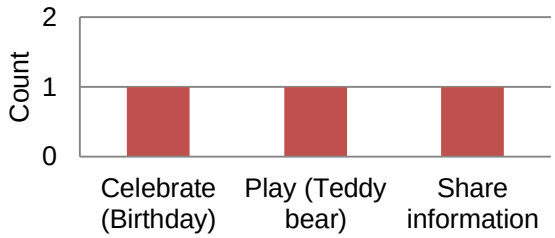


Figure 35. Social subthemes for free-form applications

The behavior change theme was subdivided into subthemes about exercise, energy consumption, diet, medication, TV watching and cleanliness. Figure 33 shows how many applications followed these subthemes.

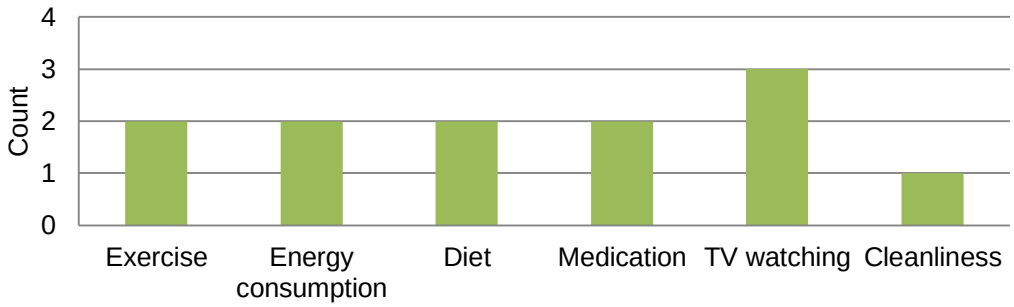


Figure 36. Behavior change subthemes for free-form applications

The home automation theme was not subdivided as all of the applications were fairly similar, participants either wanted to automate turning the lamp on/off or to play music according to a specific trigger.

Additionally, it was noticed that there were some recurring actions that participants referred to on several free-form aplications, which are shown in figure 34.

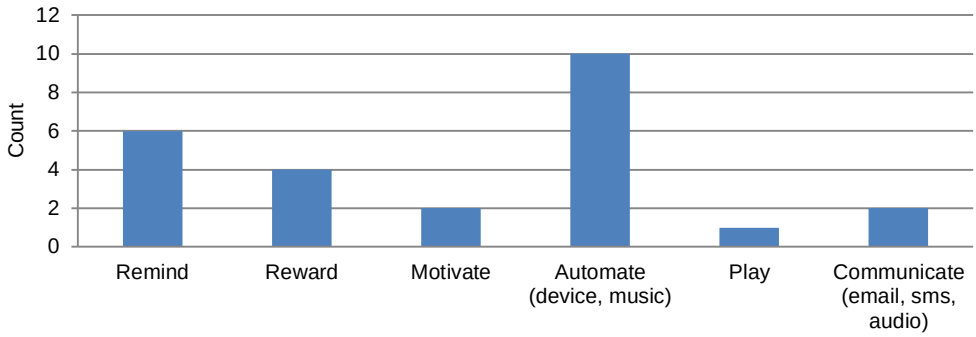


Figure 37. Typical actions used on free-form applications

Finally, in order to determine which GALLAG Strip elements were used the most, a simple break down of the type of application frames used during the free-form programming task is shown in figure 35.

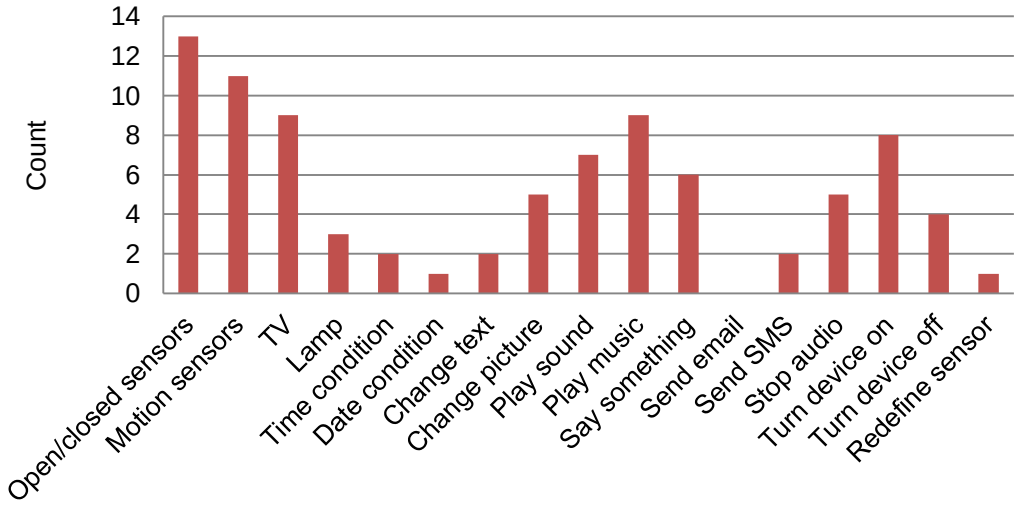


Figure 38. Types of GALLAG Strip elements used on free-form app

Not surprisingly, magnetic, motion sensors, TV and playing music were the most used (13, 11, 9 and 9 respectively). However, it is interesting to note that no one used the

email feature, and that the SMS, date and time features were seldom used. The reason might be that they were a little more complicated to test during the session than the rest. The feature to change the text on action frames was not used much either; although this was pointed out not to be necessary at that time, as they were able to understand what sensors they were using and did not feel the need to customize the text labels. Finally, only one user was interested in redefining the purpose of one of the motion sensors (the one below the table) to detect a teddy bear instead of the dumbbells.

DISCUSSION

I decided to follow a mixed-methods approach (i.e., CDN, think aloud sessions, user study, questionnaire) to evaluate GALLAG Strip because these are common practices in the Human-Computer Interaction field and there is really no established way to do this in emerging areas like physical and context-aware computing. I want, however, to mention some limitations in the methodology.

For the design examination using the CDN framework, the recommended procedure is that several design experts do it; because no one else is involved in the system's design or is knowledgeable on how to perform the CDN analysis, I decided to do it myself.

Regarding the think aloud and user study sessions, since I was who decided where to place the sensors, it might have limited or influenced participants' creativity when envisioning their applications for the free-form task in some way. Similarly, the fact that they were shown a demo application at the beginning of the session and that they were asked to program three previously defined applications, might have also influenced their decisions when thinking about their free-form application. The only way to answer this would be to run another study and ask them to define their envisioned application(s) at the beginning of the session.

For the user study sessions, the goal was to have at least 20 participants; unfortunately, this was not possible, as it was far more difficult to recruit people than I thought. I believe that my inability (because it was not approved by IRB) to pay participants limited me substantially in reaching this goal; if I were to do the study again, I would make sure I can pay them. Even though 13 participants may be a small sample to draw strong conclusions, I think it still provided solid feedback to answer the research questions.

CONCLUSION

This thesis presented the design, implementation and evaluation of a PWD environment to create sensor-based applications. By combining the PWD technique with a mobile application, a novel approach in the way users program sensor-based applications is provided, allowing them to experience what they are programming in real-time.

According to the data and feedback gathered from the user study, it can be concluded that GALLAG Strip effectively lowers the barrier for novice users and allows them to program GALLAG applications easily, accurately and without the need of prior programming experience. Even with its current limitations, GALLAG Strip enabled users to create almost all of their envisioned applications during the user study. It is necessary, however, to compare it with another authoring system to be able to tell if it is a better approach to programming sensor-based context-aware applications or not, something which I am planning on working on shortly.

I acknowledge that the current programming interface might not allow the creation of more complex applications; however, the initial goal was for users to be able to program simple, if-then rule-based GALLAG applications. This could be solved in two ways: one could be by adding more options to the current if-then rules, like the ability to have else statements and OR conditions; another option could be to enable GALLAG Strip to learn from a series of user demonstrations by implementing an inference engine, thus making GALLAG Strip a PBD environment that creates generalized applications.

Future improvements should allow users to reorder frames in the application strip, as it causes a major burden when the user needs to add a frame in the middle or beginning of the application. Another feature that should be added is the ability to have timers, not only initial date-time conditions. Also, there should be some kind of feedback when testing a saved application, as it is difficult to understand if an application is being activated or not.

It should also be considered that the current implementation of GALLAG Strip only supports a single user; a way to distinguish different users will need to be implemented for the system to allow multiple users with different preferences.

Finally, the current design for GALLAG Strip should allow the mobile application to be ported to other mobile platforms like a Windows 8 tablet, iOS or Android fairly easily, as the communication with the server application is done through platform-agnostic web services and XML.

REFERENCES

1. Beckmann, C. and Dey, A.K. *SiteView : Tangibly Programming Active Environments with Predictive Visualization*. Berkeley, CA, 2003.

2. Blackwell, A. and Green, T. A Cognitive Dimensions Questionnaire. http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDquestionnaire.pdf.

3. Blackwell, A. and Green, T. Notational Systems – the Cognitive Dimensions of Notations framework. *HCI Models, Theories and Frameworks: Toward*, December (2002), 1-21.

4. Boren, T. and Ramey, J. Thinking aloud: Reconciling theory and practice. *Professional Communication, IEEE Transactions on 43*, 3 (2000), 261–278.

5. Brooke, J. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, (1996), 189–194.

6. Burleson, W., Ruffenach, C., Jensen, C., Bandaru, U.K., and Muldner, K. Game as life --- life as game. *Proceedings of the 8th International Conference on Interaction Design and Children - IDC '09*, (2009), 272.

7. Cook, W.R. AppleScript. *SIGPLAN conference on History of programming languages (HOPL III)*, ACM (2007), 1-21.

8. Cypher, A. Bringing Programming to End Users. In A. Cypher, ed., *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993, 2-11.

9. Dey, A., Sohn, T., Streng, S., and Kodama, J. iCAP: Interactive prototyping of context-aware applications. *Pervasive Computing*, (2006), 254–271.

10. Dey, A.K., Hamid, R., Beckmann, C., Li, I., and Hsu, D. a CAPpella: programming by demonstration of context-aware applications. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2004), 33–40.

11. Girard, P. Bringing Programming by Demonstration to CAD Users. In *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, 2000, 135-162.

12. Halbert, D.C. SmallStar: Programming by Demonstration in the Desktop Metaphor. In *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993, 103-124.

13. Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S.R. Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (2007), 145–154.

14. Holzinger, A. Usability Engineering Methods for Software Developers. *Communications of the ACM 48*, 1, 71-74.

15. Kauhanen, M. and Biddle, R. Cognitive dimensions of a game scripting tool. *Proceedings of the 2007 conference on Future Play - Future Play '07*, (2007), 97.

16. Klemmer, S. and Landay, J. Toolkit Support for Integrating Physical and Digital Interactions. *Human-Computer Interaction 24*, 3 (2009), 315-366.

17. Ko, A.J., Myers, B., Rosson, M.B., et al. The state of the art in end-user software engineering. *ACM Computing Surveys 43*, 3 (2011), 1-44.

18. Kurlander, D. Chimera: Example-Based Graphical Editing. In *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993, 271-292.

19. Lieberman, H. Mondrian: A Teachable Graphical Editor. In *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993, 341-360.

20. McDaniel, R.G. and Myers, B.A. Getting more out of programming-by-demonstration. *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*, (1999), 442-449.

21. Modugno, F., Corbett, A.T., and Myers, B. a. Graphical representation of programs in a demonstrational visual shell---an empirical evaluation. *ACM Transactions on Computer-Human Interaction 4*, 3 (1997), 276-308.

22. Modugno, F. and Myers, B.A. Graphical Representation and Feedback in a PBD System. In *Watch What I Do: Programming by Demonstration*. MIT Press, Cambridge, MA, 1993, 415-422.

23. Myers, B.A. Taxonomies of Visual Programming and Program Visualization. *Journal of Visual Languages & Computing 1*, 1 (1990), 97–123.

24. Myers, B.A. Peridot: Creating User Interfaces by Demonstration. In *Watch What I Do: Programming by Demonstration*. MIT Press, 1993, 125-154.

25. Nielsen, J. and Molich, R. Heuristic Evaluation of User Interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, ACM (1990), 249–256.

26. Nielsen, J. Ten Usability Heuristics. http://www.useit.com/papers/heuristic/heuristic_list.html.

27. Ruvini, J.-D. and Dony, C. Learning Users' Habits to Automate Repetitive Tasks. In *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, Cambridge, MA, 2000, 271-296.

28. Teitelman, W. Toward a Programming Laboratory. *International Joint Conference on Artificial Intelligence*, Bolt Beranek and Newman Inc. (1969), 1–8.

29. GaLLaG Library. http://hci.asu.edu/GALLAG/Library/Documentation/files/AirFoil-applescript.html.

30. GaLLaG Wiki. http://gallag.wikispaces.asu.edu/.

31. GaLLaG - Play Music While Exercising. http://gallag.wikispaces.asu.edu/Play+Music+While+Exercising.

32. GaLLaG - Matching Game. http://gallag.wikispaces.asu.edu/Matching+Game.

33. Apple - iOS. http://www.apple.com/ios/.

34. GaLLaG - Treasure Hunt Game. http://gallag.wikispaces.asu.edu/Treasure+Hunt+Game.

35. Pendaphonics. http://www.pendaphonics.com/.

36. GaLLaG - Call Contacts. http://gallag.wikispaces.asu.edu/Call+Contacts.

37. Twine. http://www.kickstarter.com/projects/supermechanical/twine-listen-to-your-world-talk-to-the-internet.

38. SketchFlow. http://www.microsoft.com/expression/products/sketchflow_overview.aspx.

39. X10.com - Security Cameras, X10 Home Security, Wireless Camera, Home Automation, Electronics and More! http://www.x10.com/homepage.htm.

40. INSTEON - Wireless Home Control Solutions for Lighting, Security, HVAC, and A/V Systems. http://www.insteon.net/.

41. Indigo: Macintosh Home Automation and Control Server. http://www.perceptiveautomation.com/indigo/index.html.

42. Google Android. http://www.android.com/.

43. Microsoft Windows Phone. http://www.microsoft.com/windowsphone/en-us/default.aspx.

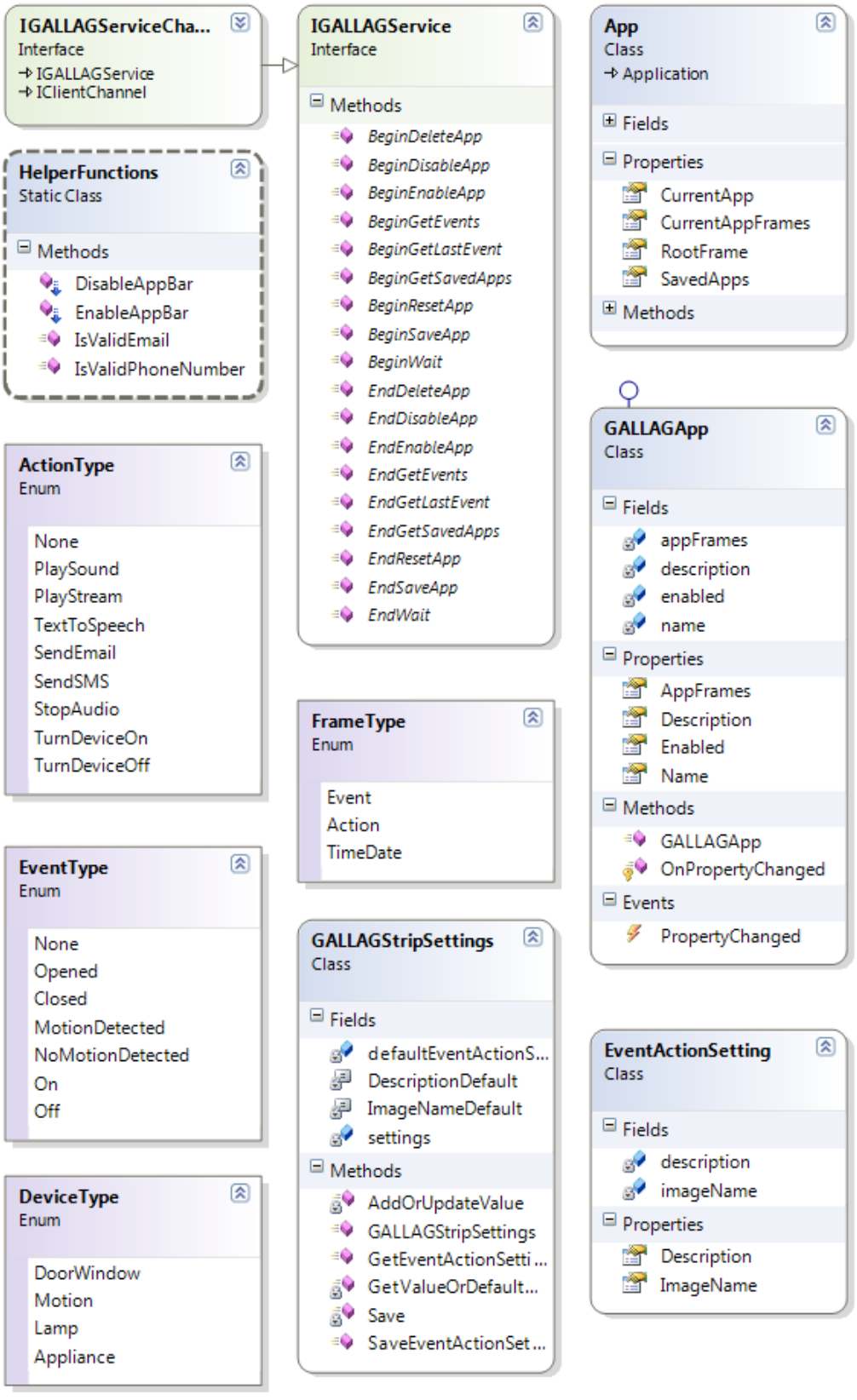
APPENDIX A

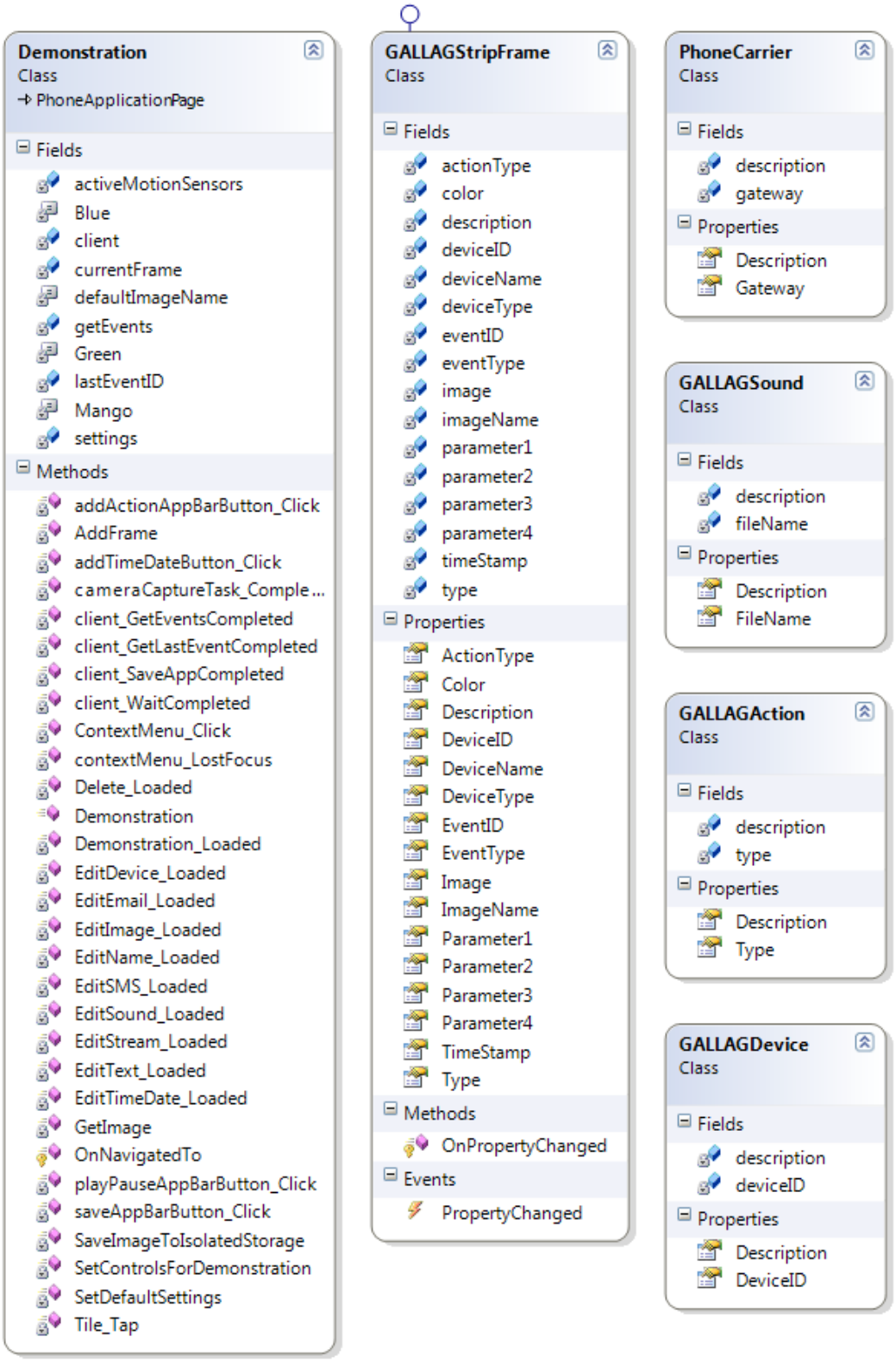
GALLAG APPLICATION XML FORMAT

```
<?xml version="1.0" encoding="utf-8"?>
<GALLAGApps>
  <GALLAGApp name="Read" desc="Read more" enabled="true">
    <Frame type="Event" deviceID="TV" deviceType="Lamp" event="On" action="None"
param1="" param2="" param3="" param4="" />
    <Frame type="Action" deviceID="" deviceType="DoorWindow" event="None"
action="PlaySound" param1="Reminder sound" param2="reminder.mp3" param3=""
param4="" />
    <Frame type="Action" deviceID="" deviceType="DoorWindow" event="None"
action="TextToSpeech" param1="You should read instead of watching TV!" param2=""
param3="" param4="" />
    <Frame type="Event" deviceID="TV" deviceType="Lamp" event="Off" action="None"
param1="" param2="" param3="" param4="" />
    <Frame type="Action" deviceID="" deviceType="DoorWindow" event="None"
action="PlaySound" param1="Achievement sound" param2="achievement.mp3"
param3="" param4="" />
    <Frame type="Event" deviceID="A5" deviceType="Motion" event="MotionDetected"
action="None" param1="" param2="" param3="" param4="" />
    <Frame type="Action" deviceID="" deviceType="DoorWindow" event="None"
action="TurnDeviceOn" param1="Lamp" param2="Lamp" param3="" param4="" />
    <Frame type="Action" deviceID="" deviceType="DoorWindow" event="None"
action="PlayStream" param1="Paris Cafe"
param2="http://u17.jazzradio.com:80/jr_pariscafe_aacplus" param3="" param4="" />
  </GALLAGApp>
</GALLAGApps>
```

APPENDIX B

MOBILE APPLICATION CLASSES AND ENUMERATIONS


IGALLAGServiceCha...
Interface
IGALLAGService
IClientChannel
IGALLAGService
Interface
Methods
BeginDeleteApp
BeginDisableApp
BeginEnableApp
BeginGetEvents
BeginGetLastEvent
BeginGetSavedApps
BeginResetApp
BeginSaveApp
BeginWait
EndDeleteApp
EndDisableApp
EndEnableApp
EndGetEvents
EndGetLastEvent
EndGetSavedApps
EndResetApp
EndSaveApp
EndWait
App
Class
Application
Fields
Properties
CurrentApp
CurrentAppFrames
RootFrame
SavedApps
Methods
HelperFunctions
Static Class
Methods
DisableAppBar
EnableAppBar
IsValidEmail
IsValidPhoneNumber
GALLAGApp
Class
Fields
appFrames
description
enabled
name
Properties
AppFrames
Description
Enabled
Name
Methods
GALLAGApp
OnPropertyChanged
Events
PropertyChanged
ActionType
Enum
None
PlaySound
PlayStream
TextToSpeech
SendEmail
SendSMS
StopAudio
TurnDeviceOn
TurnDeviceOff
FrameType
Enum
Event
Action
TimeDate
EventType
Enum
None
Opened
Closed
MotionDetected
NoMotionDetected
On
Off
GALLAGStripSettings
Class
Fields
defaultEventActionS...
DescriptionDefault
ImageNameDefault
settings
Methods
AddOrUpdateValue
GALLAGStripSettings
GetEventActionSetti ...
GetValueOrDefault...
Save
SaveEventActionSet ...
EventActionSetting
Class
Fields
description
imageName
Properties
Description
ImageName
DeviceType
Enum
DoorWindow
Motion
Lamp
Appliance


Demonstration
Class
PhoneApplicationPage
Fields
activeMotionSensors
Blue
client
currentFrame
defaultImageName
getEvents
Green
lastEventID
Mango
settings
Methods
addActionAppBarButton_Click
AddFrame
addTimeDateButton_Click
cameraCaptureTask_Comple...
client_GetEventsCompleted
client_GetLastEventCompleted
client_SaveAppCompleted
client_WaitCompleted
ContextMenu_Click
contextMenu_LostFocus
Delete_Loaded
Demonstration
Demonstration_Loaded
EditDevice_Loaded
EditEmail_Loaded
EditImage_Loaded
EditName_Loaded
EditSMS_Loaded
EditSound_Loaded
EditStream_Loaded
EditText_Loaded
EditTimeDate_Loaded
GetImage
OnNavigatedTo
playPauseAppBarButton_Click
saveAppBarButton_Click
SaveImageToIsolatedStorage
SetControlsForDemonstration
SetDefaultSettings
Tile_Tap
GALLAGStripFrame
Class
Fields
actionType
color
description
deviceID
deviceName
deviceType
eventID
eventType
image
imageName
parameter1
parameter2
parameter3
parameter4
timeStamp
type
Properties
ActionType
Color
Description
DeviceID
DeviceName
DeviceType
EventID
EventType
Image
ImageName
Parameter1
Parameter2
Parameter3
Parameter4
TimeStamp
Type
Methods
OnPropertyChanged
Events
PropertyChanged
PhoneCarrier
Class
Fields
description
gateway
Properties
Description
Gateway
GALLAGSound
Class
Fields
description
fileName
Properties
Description
FileName
GALLAGAction
Class
Fields
description
type
Properties
Description
Type
GALLAGDevice
Class
Fields
description
deviceID
Properties
Description
DeviceID

APPENDIX C

CDN QUESTIONNAIRE

CDN Questionnaire used for usability examination.

1. Viscosity: Resistance to change.

   How much effort is required to perform a single change?

2. Visibility: Ability to view components easily.

   Is every part of the code simultaneously visible or is it at least possible to juxtapose any two parts side-by-side at will? If the code is dispersed, is it at least possible to know in what order to read it?

3. Premature Commitment: Constraints in the order of doing things.

   Do users have to make decisions before they have the information they need?

4. Hidden Dependencies: Important links between entities are not visible.

   Is every dependency overtly indicated in both directions? Is the indication perceptual or only symbolic?

5. Role-Expressiveness: The purpose of an entity is readily inferred.

   Can the user see how each component of the application being programmed relates to a whole?

6. Error-Proneness: The notation invites mistakes and the system give little protection.

   Does the design of the notation induce “careless mistakes”?

7. Abstraction: Types and availability of abstraction mechanisms.

   What are the minimum and maximum levels of abstraction? Can fragments be encapsulated?

8. Closeness of Mapping: Closeness of representation to domain.

   What “programming nuances” need to be learned?

9. Consistency: Similar semantics are expressed in similar syntactic forms.

   When some of the language has been learned, how much of the rest can be inferred?

10. Diffuseness: Verbosity of language.

    How many symbols or graphic entities are required to express a meaning?

11. Hard Mental Operations: High demand on cognitive resources.

    Are there places where the user needs to resort to penciled annotation to keep track of what's happening?

12. Provisionality: Degree of commitment to actions or marks.

    Does the system allow for prototyping to try out new ideas or provisional actions?

13. Progressive Evaluation: Work-to-date can be checked at any time.

    Can a partially complete program be executed to obtain feedback on "How am I doing"?

APPENDIX D

USER STUDY CONSENT FORM

Consent Form

GaLLaG: Game as Life – Life as Game

INTRODUCTION

The purposes of this form are to provide you (as a prospective research study participant) information that may affect your decision as to whether or not to participate in this research and to record the consent of those who agree to be involved in the study.

RESEARCHERS

Winslow Burleson, assistant professor School of Computing, Informatics, and Decision Systems Engineering, along with other researchers (Ryan Brotman, Camilla Jensen, Byron Lahey, Jisoo Lee, Shawn ORourke, Luis Garduno, Naomi Newman) have invited your participation in a research study.

STUDY PURPOSE

The purpose of the research is to investigate the efficiency of interactive technology in guiding subjects toward achieving desired goals.

DESCRIPTION OF RESEARCH STUDY

If you decide to participate, then you will join a study funded by NFS grant, and we will ask you to initially select a goal to achieve from a panel of possibilities. Once selected, a script will be designed to stimulate and aid the subject in achieving the particular goal. The scriptwriter may be either the experimenter, another subject with whom you have already been paired with, or you may write the script yourself. Script merely refers to the actions and processes utilized by the environmental technology to aid you in accomplishing your goal.

The technology being utilized may include a Macintosh laptop, an iPhone, iTouch, and may also include speakers located around a house or a television capable of turning off after a certain length of time. In addition, we will record some physiological data such as body temperature, facial expression, and interaction patterns while you are working on the assigned project. The sensors that are used in the study are a form of 'wearable computing' just like your cellphone. They capture digital information about your location, interactive environment, and activity patterns. These patterns will help us in creating future scripts.

The progress of the game will be recorded and evaluated in order to assess the efficacy of the script. The actual effectiveness of these scripts greatly varies depending upon the scriptwriter and particular goal to be achieved.

RISKS

There are no known risks from taking part in this study, but in any research, there is some possibility that you may be subject to risks that have not yet been identified.

BENEFITS

The benefits of your participation in the research are to obtain a higher level of motivation for achieving desired goals. In addition, you will be helping to advance the research community's understanding of directing motivation by means of interactive technology.

CONFIDENTIALITY

All information obtained in this study is strictly confidential. The results of this research study may be used in reports, presentations, and publications, but the researchers will not identify you.

WITHDRAWAL PRIVILEGE

Participation in this study is completely voluntary. It is ok for you to say no. Even if you say yes now, you are free to say no later, and withdraw from the study at any time. Your decision will not affect your relationship with Arizona State University and any other institution or otherwise cause a loss of benefits to which you might otherwise be entitled. This includes your grades as a student or employment status, treatment, care etc.

COSTS AND PAYMENTS

The researchers want your decision about participating in the study to be absolutely voluntary. There is no payment for your participation in the study.

VOLUNTARY CONSENT

Any questions you have concerning the research study or your participation in the study, before or after your consent, will be answered by Dr Winslow Burleson, School of Computing, Informatics, and Decision Systems Engineering, at (480) 965-9253.
If you have questions about your rights as a subject/participant in this research, or if you feel you have been placed at risk; you can contact the Chair of the Human Subjects Institutional Review Board, through the ASU Office of Research Integrity and Assurance, at (480) 965-6788.

This form explains the nature, demands, benefits and any risk of the project. By signing this form you agree knowingly to assume any risks involved. Remember, your participation is voluntary. You may choose not to participate or to withdraw your consent and discontinue participation at any time without penalty or loss of benefit. In signing this consent form, you are not waiving any legal claims, rights, or remedies. A copy of this consent form will be offered to you.

VIDEO RECORDING CONSENT

This study involves the video recording of your performance. Parts of the recording of the session will be transcribed to written form, without identifying the speakers. The recording will be erased when all data from it have been reviewed and coded.

(Please check the appropriate blank below.)

_____ **I DO NOT** agree to be video recorded.

_____ **I DO** agree to be video recorded.

Your signature below indicates that you consent to participate in the above study.

_____ _____ _____

Subject's Signature Printed Name Date

INVESTIGATOR'S STATEMENT

"I certify that I have explained to the above individual the nature and purpose, the potential benefits and possible risks associated with participation in this research study, have answered any questions that have been raised, and have witnessed the above signature. These elements of Informed Consent conform to the Assurance given by Arizona State University to the Office for Human Research Protections to protect the rights of human subjects. I have provided (offered) the subject/participant a copy of this signed consent document."

_____ _____

Signature of Investigator Date

APPENDIX E

IRB APPROVAL

CITI Collaborative Institutional Training Initiative (CITI)

Responsible Conduct of Research Curriculum Completion Report
Printed on 9/30/2010

Learner: Luis Garduno (username: lgarduno)
Institution: Arizona State University
Contact Information Arizona State University
Department: CIDSE - Computer Science
Email: luisgarduno@asu.edu

Social and Behavioral Responsible Conduct of Research:

Stage 1. Basic Course Passed on 04/02/10 (Ref # 4275965)

Required Modules	Date Completed	Score
Introduction to the Responsible Conduct of Research	04/02/10	no quiz
Introduction to Research Misconduct	04/02/10	no quiz
Research Misconduct 2-1495	04/02/10	5/5 (100%)
Data Acquisition, Management, Sharing and Ownership 2-1523	04/02/10	4/5 (80%)
Publication Practices and Responsible Authorship 2-1518	04/02/10	5/5 (100%)
Peer Review 2-1521	04/02/10	5/5 (100%)
Responsible Mentoring 01-1625	04/02/10	6/6 (100%)
Conflicts of Interest and Commitment 2-1462	04/02/10	6/6 (100%)
Collaborative Research 2-1484	04/02/10	5/6 (83%)
The CITI RCR Course Completion Page.	04/02/10	no quiz
Arizona State University	04/02/10	no quiz

For this Completion Report to be valid, the learner listed above must be affiliated with a CITI participating institution. Falsified information and unauthorized use of the CITI course site is unethical, and may be considered scientific misconduct by your institution.

Paul Braunschweiger Ph.D.
Professor, University of Miami
Director Office of Research Education
CITI Course Coordinator