e-Chimera: Rapid In-Situ Mobile Experimentation through Integrated Design, Test, and Deployment

WENCAN LUO, University of Pittsburgh
ANDREW HEAD, University of Pittsburgh
WALTER SCHNEIDER, University of Pittsburgh
JINGTAO WANG, University of Pittsburgh

The advent of smartphones and Internet tablets provides unique opportunities for running large-scale, in-situ experiments in the wild. However, the steep learning curve of building mobile applications makes it difficult for researchers in most fields to design and deploy cognitive, behavioral, and social experiments on mobile platforms. Inspired by contextual inquiries with domain researchers and a comprehensive literature survey, we present e-Chimera, a visual end-user programming environment for designing and prototyping experiments on mobile devices. The e-Chimera IDE provides a statechart-based, multi-level visual language to create experiments at the workflow, trial, and screen interaction level. The e-Chimera runtime, driven by a Domain-Specific Language (DSL), enables high fidelity, cross-platform application deployment. Overall, e-Chimera integrates design, test, deployment, and analysis of mobile experiments. Through a 12-subject user study, we found e-Chimera effective and easy to use.

Categories and Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: prototyping; user-centered design.

General Terms: Design, Experimentation, Human Factors

Additional Key Words and Phrases: Mobile devices; End-User Programming, Visual Language, Rapid Prototyping Toolkits, Smartphone

1. INTRODUCTION

The popularity of mobile devices, such as smartphones and Internet tablets, presents both unique opportunities and challenges to a wide range of research fields. With their availability, portability, and connectivity, mobile devices are an ideal platform for crowdsourcing [Musthag and Ganesan 2013] and collecting environmental [Kim et al 2011] and behavioral data in situ [Froehlich et al 2007, Yeh et al 2006]. However, the form factor and unique input and output modalities raise new challenges related to human-computer interaction, from theoretically modeling new interactions [Bi et al 2013] to mobilizing existing non-mobile applications [Goel et al 2012].

Designing and deploying mobile experiments is expensive, challenging, and time consuming, especially for researchers in fields such as psychology, cognitive science, and behavioral science, for at least three reasons. First, most rapid mobile prototyping tools, such as RhoMobile Suite [RhoMobile 2010], Appcelerator Titanium [Appcelerator 2011], PhoneGap [PhoneGap 2011], and Corona SDK [Corona 2010] require textual programming to define interaction logic – these tools are built for programmers. This leaves end-user researchers with a sizeable gulf of execution [Hutchins 1985], or gap between goals and the actions needed to translate them into...
programming language statements. Second, existing tools typically support development of general purpose applications like games or mobile clients for web sites, rather than providing low threshold environments for designing interactive and exploratory mobile experiments. Third, common requirements such as distributing mobile experiments, collecting user data, and monitoring task progress are crucial for running experiments in the wild, but such requirements are not addressed in general purpose mobile prototyping tools.

To address these challenges, we present e-Chimera, a visual end-user programming environment for designing and prototyping experiments on mobile devices. The e-Chimera environment includes three parts: 1) the e-Chimera IDE (Figure 1), which provides a statechart-based, multi-level visual language to create experiments at the workflow, trial, and screen interaction level; 2) The e-Chimera runtime, driven by a Domain-Specific Language (DSL) [Deursen et al 2000], which enables high-fidelity, cross-platform application deployment; 3) The e-Chimera server, which provides a centralized location to manage ongoing experiments, collect experiment data, and visualize user activities.

Overall, e-Chimera is an end-to-end toolkit for designing, testing, deploying, and analyzing mobile experiments. e-Chimera lowers the threshold for researchers and practitioners in psychology, cognitive science, and the behavioral sciences to run experiments on mobile devices.

This paper offers four major contributions:
— A new model for designing mobile experiments, inspired by both contextual inquiries with domain researchers and a comprehensive literature survey.
— A statechart-based, multi-level visual language for declarative design of mobile experiments at the workflow, trial, and screen interaction levels.
— A Domain Specific Language (DSL) extending the expressive power of the visual language, deployable through the cross-platform e-Chimera runtime.
— An end-to-end, architecture for designing mobile experiments.

In this paper, we first survey related work. We report the contextual inquiry and literature survey that inspired the design of e-Chimera. Then, we detail the major components of e-Chimera. Finally, we report results of a 12-subject user study for e-Chimera and summarize future directions for development.

2. RELATED WORK

Related work spans three categories: mobile programming and prototyping environments; end-user programming; and application toolkits for domain experts.

2.1 Mobile Programming and Prototyping Environment

Significant efforts have been made by both researchers and developers to reduce the time and effort required to build mobile applications.

Damask [Lin and Landay 2008], ActivityDesigner [Li and Landay 2008], and POP [POP 2013] were built as informal prototyping tools for designers and researchers to use during early stages of design. Among them, Damask [Lin and Landay 2008] provides developers with design patterns and pre-built UI fragments to sketch out the high-level structure of interfaces, independent of platform. ActivityDesigner [Li and Landay 2008] provides a scenario-based storyboard for researchers to develop mobile human activity-related applications for ubiquitous computing. Prototyping on Paper (POP [POP 2013]) lets designers draw mock-up interfaces on paper and load interactive UI sketches on a smartphone via camera snapshots. Although interactive low-fidelity prototypes created through these systems run on mobile devices, additional code and testing are needed before they can be made into fully usable and functional applications.

RhoMobile [RhoMobile 2010], Appcelerator Titanium [Appcelerator 2011], PhoneGap [PhoneGap 2011], MoSync, IBM Worklight [IBM 2012], Corona SDK [Corona 2010] and Xamarin [Xamarin 2012] are toolkits and frameworks for rapid, cross-platform mobile prototyping. These tools often provide unified programming APIs in platform-neutral scripting languages (JavaScript [Appcelerator 2011, IBM 2012, PhoneGap 2011], C# [Xamarin 2012], Ruby [RhoMobile 2010], Lua [Corona 2010]) to lessen learning time and porting effort for mobile programmers. While these toolkits are effective for developing certain applications types like mobile clients for web portals, they suffer from the lowest common denominator effect on multiple platforms. More importantly, these toolkits require textual programming experience and skill. As a result, non-expert programmers will have difficulty developing interaction logic, data collection, and trial management for experiments when using these toolkits.

Google App Inventor [Google 2010] allows programming learners to create mobile applications via Block-based Visual Programming (BVP). While the “interlocking puzzle piece” development paradigm of App Inventor can effectively lower the challenge of creating syntactically-valid programming statements, App Inventor and similar environments suffer from a “low ceiling” for creating complex applications and a lack of extensibility.

MyExperience [Froehlich 2007] is a system for collecting user opinion data through the Experience Sampling Method (ESM) on mobile devices in the wild. While designers can use JavaScript embedded in XML to specify sensors to poll at specific
times to collect data, front end interaction is restricted to static questionnaires in pop-up windows.

In summary, although existing research considers many stages of the development process for diverse users, there is still no toolkit optimized for designing and deploying psychological experiments on mobile devices.

2.2 End-User Programming

The design of e-Chimera is motivated in part by previous research on end-user programming, especially visual programming environments [Burnett 1999, Kurlander 1993].

d.tools [Hartmann 2006] is an iterative design-centered system for prototyping physical UIs. The visual languages in d.tools and e-Chimera are similar because both systems provide statechart-based environments on top of Eclipse.

iStuff Mobile [Ballagas et al 2007] allows researchers to rapidly design new sensor-based interaction techniques on mobile devices. While the event pipeline interface in iStuff Mobile (an extension of Apple Quartz Composer) is easy to learn, iStuff Mobile was designed for exploring new physical sensor-based interaction techniques and not for deploying applications in the wild – generated prototypes require a host PC for continuous event handling.

Dedicated end-user programming tools sometimes support specific features rather than entire applications [Hartmann 2007, Lu and Li 2013]. Gesture Studio [Lu and Li 2013], for example, supports multi-touch interaction design via a video-editing metaphor. Examplr [Hartmann 2007] supports the design of sensor-based interactions through programming by demonstration. PICL [Fourny and Terry 2012] provides machine learning-supported programming-by-demonstration of circuit prototypes. These tools can co-exist with e-Chimera if specific features they support are needed in mobile experiments.

2.3 Toolkits for Psychological/HCI Experiments

E-Prime [E-Prime 2001]\(^1\), Presentation [Presentation 2005], PsyToolkit, Psychtoolbox, Tscope, PsychoPy, and other tools have been created for programming and running psychological experiments. Available commercially, E-Prime [E-Prime 2001] and Presentation [Presentation 2005] are perhaps the two most mature solutions. Both systems allow researchers to define the structure of experiments through forms, to layout on-screen stimuli via a visual editor, to implement the experiment logic in a built-in scripting language, and to run the designed experiments on PCs. E-Prime [E-Prime 2001] and Presentation [Presentation 2005] are sometimes called the “MATLAB”s of psychology researchers. Code snippets [Script 2010] for both systems are maintained and widely circulated. Unfortunately, neither system supports creating experiments for mobile devices. Besides the holistic solutions above, tools such as TouchStone [Mackey 2007] and ExperiScope [Guimbretière 2007] address specific experiment design challenges like exploring alternative parameters and visualizing experimental results.

In comparison, e-Chimera goes beyond a “mobile port” of existing tools in at least two ways. First, the multi-level visual language in e-Chimera provides a low-threshold, end-to-end solution for creating experiments without textual programming. The visual language is also extensible through code for high-ceiling

---

\(^1\) The name e-Chimera is a word play of E-Prime [E-Prime 2001] and Chimera [Kurlander 1993], a pioneering visual end-user programming environment. The lowercase ‘e’ implies mobile devices.
tasks. Second, mobile experiments created via e-Chimera provide built-in support for in-the-wild deployment and real-time visual analytics. Such mechanisms are essential for making in-situ experimentation on mobile devices practical.

3. NEED FINDING

To get a better understanding of major challenges and opportunities for designing and running psychology experiments, we conducted contextual inquiries with seven domain researchers and ran an in-depth literature survey of 76 classic psychology experiments.

3.1 Contextual Inquiries

The seven participants (6 males and 1 female) in our contextual inquiries were researchers in psychology and cognitive science from two local universities. 3 were research scientists, 2 were professors, and 2 were PhD candidates. We collected background information on the experiments they designed, and the coding experience of both themselves and their colleagues. We use pseudonyms S1 - S7 to represent them below. Major findings include:

**Programming experience is ubiquitous, but proficiency is not.** While all psychology researchers have exposure to programming, the proficiency level varies per research direction and individual. For example, researchers in visual psychology did more hands-on coding compared to those in education research (S2). However, most researchers did not think they were proficient in programming. They usually designed experiments with environments created for psychology researchers, including E-Prime [E-Prime 2001] and Psychtoolbox. Depending on availability of existing resources and the nature of the experiments, tools such as MATLAB and Python were also used. All researchers admitted that they tried to avoid general purpose programming languages like C, C++, and Java because of their “complicated syntax and language constructs” (S6).

**Existing experiments are usually revised instead of starting ‘fresh’.** Domain researchers rarely wrote programs from ‘scratch,’ but they frequently made revisions to existing experiment code created by themselves or other researchers. For them, new experiments could be formed from existing experiments by changing parameters or the order of sub-tasks. As S1 told us, “I’m stealing code from other things I’ve done, like everybody else, right?” While code reuse led to shorter prototyping time, it restricted the scope of researchers’ exploratory experimentation.

**Randomization and counterbalancing are tricky to get right.** S2 raised the point, and other researchers confirmed, that randomization and [task] counterbalancing were the root of many errors that could jeopardize the internal validity of experiments. “Sometimes, randomization is the biggest issue when coding the experiment” (S2). Unfortunately, the randomization of tasks and experiment factors were needed in almost every experiment we surveyed.

**Researchers are eager to build mobile experiments.** All the researchers were strongly intrigued by the possibilities mobile devices could bring into their research. Some researchers explicitly named experiments they would have spent time investigating, in areas including language acquisition (S6), healthcare (S5), and human motivation, if they had mobile programming skills or resources. However, existing mobile programming environments, including “rapid prototyping” environments, were too complicated for them to learn. “We keep looking for a mobile [programming] environment equivalent to E-Prime for PC users for years, but no one is even close” (S5).
3.2 Literature Survey

In addition to contextual inquiries, we also surveyed 76 classic experiments in psychology [Script 2010] in depth, to get a better understanding of their programming-related requirements, idioms and patterns. The list of experiments was recommended by one interviewee (S7), and was confirmed to be representative by other researchers. Although some experiments were initially reported more than 60 years ago, such as Fitts’s Law [Fitts 1954] and Stroop Effect [Stroop 1935], researchers still actively investigate experimental variations of these in highly diverse contexts [Bi et al 2013] today. For each experiment, we identified the key information in experimental design including stimulus types, response types, experimental structure, and factor design. A portion of the data we collected is shown in Table 1.

We observed several interesting patterns in experimental design. Many patterns were confirmed by the findings of our contextual inquiries. The patterns we discovered were:

**Table 1.** Key design features for the 10 out of the 76 psychological experiments we surveyed [Script 2010]. Features are divided into 4 categories: stimulus type, input type, timer controller, and factor variation.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Materials/Stimuli</th>
<th>Output</th>
<th>Mouse Input</th>
<th>Keyboard Input</th>
<th>Timer Control</th>
<th>Subject Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baddeley 1966</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barsness 1994</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barsness 1995</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bilodeau 1959</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boland 1990</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bower 1972</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brumland 1971</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brewer 1977</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Broadbent 1954</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brown 1958</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Task-Block-Trial(TBT) Hierarchy.** Most experiments had a hierarchical structure. The typical psychological experiment is composed of one or multiple tasks at the top level. Each task consists of a sequence of blocks; each block was divided into multiple trials; at the lowest level, a trial consists of a number of stimulus-response interactions. A subject can get exposed to all tasks (within-subject design), or a subset of the tasks (between-subject design or mixed design). Trials controlled by the same parameter can be repeated multiple times (replication). Randomization and counterbalancing are usually performed at the task and block levels. 61 out of the 76 experiments (80.3%) adopted this TBT hierarchy. Of the remaining 15 experiments, one had a unique structure, and the structural details for the other 14 experiments were not explicitly disclosed.

**Stimulus–Response Pattern.** A trial typically consisted of the presentation of experimental stimuli, like text or images, followed by collection of the subject’s response to each stimulus. Often the response was in the form of mouse clicks or key press. 95% of experiments we studied used this pattern².

² The other 5% used physical objects, like levers, as stimuli.
Stimulus Type Distribution. The majority of stimuli were static pre-generated text, images, audio, and video; dynamic stimuli like images rendered “on-the-fly” were rarely used. Text and image was the most common stimulus; each appeared in 45% of experiments, followed by sound (26%). As suggested by S1, “Dynamic stimuli are very rarely used.” S2 told us, “for vision scientists [it is] uncommon to use video... It’s more common to use pictures, rich text, or animated [ui] that were generated [ahead of the time].” (S2)

Millisecond-level Precision. Experiments we surveyed often required high resolution timing, often as precise as 1 millisecond. As S1 described, “Precise timing is very important, like presenting the stimuli within 5ms error and collecting time information with accuracy less than 1ms.” Both the stimulus and response often must be logged with a high-precision timer.

In summary, our contextual inquires confirmed the demand for and current lack of easy-to-use, “researcher-friendly” environments for designing and prototyping psychological experiments for mobile devices. Our contextual inquires and literature survey identified patterns that differentiate the implementation of psychological experiments from the process of general-purpose programming. Informed by these findings, we have designed e-Chimera, a visual end-user programming environment for designing and prototyping experiments on mobile devices.

4. E-CHIMERA: AN EXAMPLE

To illustrate how a researcher in psychology can create a fully functional, deployable mobile experiment by using e-Chimera, we follow an example researcher, Alex, who wants to study the impact of walking on target acquisition tasks. The mobile experiment he wants to build will show a set of 1D target acquisition tasks with varying target sizes and distances. Factors of the experiment include target size, target distance, feedback type (including ‘no feedback,’ ‘audio feedback,’ and ‘tactile feedback’), and walking speed. Event timestamps, the parameters of the stimuli, the locations of user responses like finger touches, and the outcome of the response (hit, miss) need to be logged in a pre-defined format for future analysis. Trials need to be randomized and counterbalanced.

Such an experiment will require around 3,000 lines of code and days to implement when using a state-of-the-art mobile development environment such as Android SDK or XCode for iOS, even for programmers proficient in mobile programming. Instead of hiring a programmer to build the intended application, Alex can create it himself within an hour with e-Chimera.

Alex launches the e-Chimera IDE (Fig. 1) and creates a new project. Since this experiment includes four factors and the randomization of trials could be tricky, Alex decides to adopt the Task-Block-Trial (TBT) Hierarchy in e-Chimera. The hierarchy lets him create a complete experiment using the workflow editor (Fig. 1.a), the trial editor (Fig. 1.b), and the screen editor (Fig. 1.c) sequentially.

3 Simpler experiments can be created in e-Chimera without using the workflow editor, or even without using the trial editor.
4.1 Workflow Editor

Alex first inserts a block widget into the timeline of the workflow editor (Fig. 1.a and Fig. 2). The width of the block widget is a rough estimation of the expected duration of the block, which is adjustable through parameter settings.

The block widget has indicators (Fig. 2) that describe the block’s content. The \( \mathcal{F} \) stamp notifies Alex whether factors are defined for the block. The \( \mathcal{O} \), \( \mathcal{I} \), and \( \mathcal{A} \) icons tell him whether a timer, image, or audio stimulus, respectively, is used in the block. The \( \mathcal{X} \) stamp counts the number of trials, or repetitions, of the block. The \( \mathcal{X} \) icon estimates the block’s total time in seconds. Double-clicking the \( \mathcal{F} \) icon, he opens a Factor View (Fig. 1.d) that shows trial-by-trial permutations of factors when the block is run.

By hovering his mouse over the block widget, Alex previews the task’s statechart as it would appear in the trial editor. He double-clicks the block widget and opens the trial editor (Fig. 1.b and Fig. 3) to design the trial’s statechart and to determine block factors.

4.2 Trial Editor

In the trial editor, Alex adds screens to the statechart from the palette. For this trial, he inserts three blank screens, which will become the fixation screen, initial stimulus screen, and result screen. In the statechart, each screen is a preview of the mobile screen to be shown to the experiment’s end user. Screens contain both stimuli and actions to take following user response.

Alex defines the factors that change for each trial – target size, target distance, feedback type, and walking speed – by inserting factor widgets into the trial editor area. Via a side panel, he determines the levels for each factor and whether each factor varies between subjects or within subject.

It’s time for Alex to insert stimuli into his screens. He double-clicks on a screen to open up the screen editor (Fig. 1.c).
4.3 Screen Editor

In the screen editor, Alex selects stimuli (text, image, and sound) from the resource library and positions them on the screen. Alex uses two rectangle images in different colors to represent the starting position and the target of the target acquisition task. The location and size of the rectangle images are linked to factors defined in the trial editor.

4.4 Finishing Touches

After laying stimuli for all screens in the screen editor, Alex defines transition between screens in the trial editor. Transitions are triggered by user events such as key presses, mouse clicks, or timers. In this case, Alex drags the mouse cursor from the target icon in the initial stimulus screen to the result screen, hence adding a screen transition that is triggered by tapping on the target region (Fig. 3).

Returning to the workflow editor, Alex selects the counterbalancing method for the factors he defined. e-Chimera generates permutations of the factors to his specification. It also recommends the number of replications and subjects needed in order to get the expected predictive power by integrating TouchStone [Mackey 2007].

4.5 Deploying the Experiment

Alex clicks the Build + Run button to simulate the application on his host PC. The simulator is loaded with the experiment, and Alex steps through the application. If Alex is not yet happy with the iteration, he returns to the IDE to make changes. Once he has polished the experiment, he clicks the Deploy button to load the completed work to his mobile target.

While Alex runs the experiment on the mobile target, it generates a log file containing subject ID, factor values, and timestamps of screen transitions, click and timer events. The logs are uploaded to the server in real time if a network connection is available. If not, they will be uploaded when the network next becomes available.

The experiments Alex creates in the e-Chimera IDE run on iOS, Android, and Windows, using a single cross-platform script. They have built-in support for user management and remote data upload. Each experiment starts with a user login screen that prompts the user for a unique ID to determine the trial order and factor permutations.

Later, Alex can implement additional complex experiment logic and rich features with e-Chimera’s built-in Domain Specific Language (DSL), injecting customized logic, such as custom vibration, animations effects, and logging event formats,
through custom DSL code. This will be described in greater detail in the following section.

5. DESIGN OF E-CHIMERA

The e-Chimera environment has 3 components (Fig. 4): the e-Chimera IDE, the e-Chimera runtime, and the e-Chimera server. In the coming sections, we introduce these components and their capabilities.

![Diagram of e-Chimera System Architecture](image)

**Fig 4. e-Chimera System Architecture, including IDE, runtime, and server.**

5.1 e-Chimera IDE

e-Chimera IDE, as Alex's example showed, is a tool for designing and deploying mobile experiments. For these purposes, the IDE provides a multi-level, statechart-based visual language. Once development is complete, the IDE translates interaction logic and experiment parameters into DSL scripted programs. Applications are created by bundling generated scripts and experiment resources with the e-Chimera runtime for a target mobile platform, such as iOS, Android, or Windows.

The e-Chimera IDE differs in 3 significant ways compared to traditional visual programming environments and rapid prototyping tools.

_The e-Chimera IDE provides a visual language and interaction logic optimized for psychology experiments’ TBT Hierarchy._ Inspired by our contextual inquiries and literature survey, this visual language provides and supports four hierarchical visual language constructs (VLCs): _workflows, blocks, trials_ and _screens_. A _screen_ contains experiment stimuli, such as images, text and audio, that need to be displayed at the same time. A _trial_ is composed of two or more screens arranged in a state diagram, encapsulating the stimulus-response pattern. A _block_ is a number of trials that run either in a fixed random sequence. Finally, a _workflow_ consists of multiple blocks. The e-Chimera IDE has 3 dedicated editors for assembling experiments at the workflow, trial, and screen levels.

_e-Chimera IDE provides a Domain Specific Language (DSL) that acts as a point of access to extend the visual language._ e-Chimera’s visual language is encoded into DSL prior to deployment, and can be modified by researchers to implement more complex logic and visual effects. While the visual language provides a low threshold
development method for researchers without programming experience, the DSL offers a high ceiling for rich feature development. In the IDE, users can select which screens will need supplemental logic coded through DSL.

Applications generated in e-Chimera are more than informal prototypes. They are complete applications that can be deployed and run on target devices.

5.2 e-Chimera Domain Specific Language

The e-Chimera DSL is based on the Lua programming language (http://www.lua.org). DSL scripts are generated in two steps. An e-Chimera Syntax Tree (EST) is built in real time. This contains the visual layout of experimental stimuli and sequence of hierarchy elements at the workflow, trial, and screen level. When a user clicks the Build button in the IDE, DSL Generator (DSLG) traverses the EST to generate DSL according to the two major rules:

**Screen Layout.** DSLG generates a separate Lua script for each screen. The script includes code to load stimuli, initialize graphics and handle events. If a screen is marked to require user-provided extension code, 2 Lua files (1 stub file and 1 extension file, with the same filename but different suffix) are generated. The stub file loads stimuli, handles events and default transitions. Users can modify the extension file with custom logic.

**Transition Tracking.** To order DSL screen scripts according to user-defined trial statecharts, a topological sorting algorithm is used. As screen order may depend on user response, the order cannot just be hard-coded. After sorting, the source node is the first screen to be shown. All states with no transitions out are considered terminal nodes, and there can be multiple terminal states. DSL event handler methods represent screen transitions.

Once DSL scripts are generated, four injection points exist for users to insert their custom DSL: 1) Global injection for loading global resources and defining global variables; 2) Trial injection, for adding adaptive behavior at the trial level; 3) Stimulus injection, where users alter the content or layout of a stimulus; 4) Event injection, for defining custom event handlers such as conditional jumps, on-screen feedback.

5.3 e-Chimera Runtime

![Fig. 5: A user-created experiment, “Complex Working Memory,” running on iOS, Android, and Windows.](image)

The e-Chimera runtime is an executable that runs cross-platform DSL scripts on popular mobile platforms such as iOS, Android, and Windows. The runtime provides experiment features including user account management, time-stamping of experimental events, and logging. A unique runtime executable was compiled for each supported platform. Applications generated by the IDE are bundles of the runtime, DSL scripts, and stimulus resources.

e-Chimera runtime uses four techniques to achieve responsive interactions and precise timing. a) Instead of the traditional event dispatching model used in graphical user interfaces [Ng et al 2012], the e-Chimera runtime adopts a
synchronized coupled game loop model [Rollings and Morris 1999] to ensure guaranteed response time. b) The e-Chimera runtime relies on hardware accelerated OpenGL ES API for display rendering. c) The e-Chimera runtime uses a separate thread for time consuming file I/O and network operations. As a result, I/O delays and temporary networking failures won’t interfere with the primary experiment logic. d) The e-Chimera runtime uses platform dependent, milli-second accuracy timer on the platforms supported. On iOS, we choose `mach_absolute_time()`. On Android, we use `clock_gettime()` with the `CLOCK_MONOTONIC` parameter.  

5.4 e-Chimera Server

After an experiment has been run on a mobile device, users upload session logs to the e-Chimera server. Administrators monitor target device information including operating system, browser type, and time zone. All log files generated can be read and interpreted by the administrator.

5.5 Implementation

e-Chimera IDE was written in Java as a plug-in for the Eclipse IDE. Not counting code from third-party Graphical Editing Framework (GEF) and TouchStone [Error! Reference source not found.], e-Chimera IDE was built from 44,663 lines of code in Java. The e-Chimera runtime was written in C++, implemented on top of the Cocos2d-X game engine and its Lua language interpreter. The e-Chimera server was written in Ruby on Rails, with 1,492 lines of code in Ruby. Communication between the runtime and server is built on JSON and REST.

6. SAMPLE EXPERIMENTS IN E-CHIMERA

To evaluate the e-Chimera’s expressive power, we used the software to build 3 classic psychology experiments and 2 active research projects for mobile devices. Each required leveraging specific capabilities of e-Chimera, which we describe below.

The 3 classic experiments were implemented as follows:
— *Face Recognition* [Kapoor 2008]. Built with a 1-block, 1-trial workflow with 1 screen per image, for a total of 6 images.
— *Stroop Effect* [Stroop 1935]. Click-triggered transitions were activated by each selection of each color prompt.
— *Fitts’ Law* [Fitts 1954]. Random factors were defined for target distance and width. 6 lines of code at 2 injection points were needed to enable hit/miss-based audio feedback and the processing of random factors. The touch event handler for each screen was extended to record touch data.

The 2 active research projects were implemented as follows:
— *N-Back* [Kirchner 1958]. Scripts were extended to show different stimuli for each trial and to adjust the difficulty of the test adaptively. This required extension at 4 injection points: global injection for loading resources by difficulty level; event injection to providing accuracy feedback; stimulus injection to load random stimulus letters; and trial injection to dynamically adjusting difficulty past performance.

4 Commonly used POSIX compatible timing APIs such as `clock()` and `gettimeofday()`, didn’t work well in our tests on mobile devices for two reasons. 1) Low accuracy. The error could be up to 150ms every two second (`clock()` on iOS). 2) The “wall clock” time returned by these functions could which can jump forward or backward suddenly if the network updates the time.
— *Pyramid and palm trees* [Howard and Patterson 1992]. This binary-choice task was completely implemented with the e-Chimera IDE.

![Fig. 6. Experiments created in e-Chimera. From left to right, top to bottom, they are: Face Recognition, Stroop Effect, Fitts’ Law, N-Back, Pyramid and Palm Tree Test, Paced Auditory Serial Addition Test (PASAT) [Gronwall 1977].](image)

To summarize, 3 of the above experiments were implemented in e-Chimera IDE without DSL extension. 2 experiments required DSL extension at multiple injection points. These samples show that e-Chimera IDE can be used to implement certain mobile psychology experiments with relative ease, often without requiring textual scripting.

7. USER STUDY

We conducted a 12-subject preliminary user study to examine e-Chimera’s usability. We had three goals for this study. First, we wanted to find whether the ideas behind e-Chimera visual language are easy to understand and whether the e-Chimera IDE is easy to learn to use. Second, we wished to see whether users could create psychological experiments, in this case classic experiments, within a short time. Our final goal was to collect feedback and suggestions for future iterations of e-Chimera.

7.1 Experimental Design

The study consisted of four parts:

— *Overview.* Participants were given an introduction to and live demo of e-Chimera. Users were asked to complete a toy experiment in e-Chimera as a tutorial.

— *Face Recognition.* Participants were asked to implement the classic “Face Recognition” experiment [Kapoor 2008] described in the previous section by using e-Chimera IDE. For both this task and the next, the task was timed and no step-by-step guidance was offered to participants. The experiment conductor did not answer subjects’ questions about the e-Chimera IDE.

— *Stroop Effect.* Participants were asked to implement a simplified version of the “Stroop Effect” experiment [Stroop 1935] described in the previous section.
—Qualitative feedback. Participants completed a questionnaire on their experience with e-Chimera. We asked participants to comment on each task, and describe their perceptions of the system’s usability.

![Fig. 7. Users interacting with e-Chimera during the study.](image)

7.1 Participants and Apparatus

We recruited 12 subjects (3 female) between 23 and 61 years of age (median age 25) from a local university (Fig. 7). 2 were research scientists in a research institution. The remaining 10 were graduate students.

Subjects used e-Chimera IDE on a Lenovo ThinkPad laptop computer running Windows 7. The laptop has an Intel Core 2 Duo T7250 CPU, 2GB RAM, 14-inch display and resolution of 1280x800. The IDE ran on top of Eclipse version 4.2 and Oracle Java JDK 7. Mobile experiments were run on an HTC smartphone running Android 2.3.5.

8. EVALUATION RESULTS

All 12 subjects completed all tasks within one hour. Participants took 8 minutes, 45 seconds on average to implement the “Face Recognition” experiment and 11 minutes, 8 seconds on average to implement the “Stroop Effect” experiment (Fig. 8).
On average, 208 significant IDE “user activities,” like triggering key e-Chimera functions or changing on-screen object parameters, occurred per participant per task (Fig. 9). In the IDE, participants spent over 92% of their time in the trial editor and screen editor. With subjective feedback we reveal later, this has two implications: First, e-Chimera’s simulation and feedback allow users to concentrate on specifying experiment logic and behavior instead of wasting time on implementation details. Second, the straightforward design of the workflow editor saved users from spending too much time setting up user identity management, logging, and trial management for their experiments. Of course, further controlled experiments are needed to quantify such observations.

Participants demonstrated two development strategies when constructing the TBT Hierarchy. Some participants adopted a top-down approach, adding all screens in the trial editor before switching to the screen editor to arrange stimuli. Others took a bottom-up approach, filling screens with stimuli, before connecting them in the trial editor. These strategies were consistent for users across both tasks. We suspect these strategies correspond to users’ problem solving preferences. The e-Chimera TBT Hierarchy implementation supports both strategies well.
Participants reviewed tutorials and past tasks when creating new experiments. Most of the time, this was in to check on past implementation of complex or hybrid screen transitions. We plan to add more detailed contextual help for transitions in e-Chimera IDE in the future.

8.1 Qualitative Feedback

Overall, participants were highly positive about e-Chimera. This is summarized by their closing questionnaire responses (Fig. 10). 8 subjects strongly agreed and 4 users agreed that “It was easy to learn how to use this environment. 8 subjects strongly agreed, 3 agreed, and 1 was neutral that “It’s easy to design a psychology experiment in e-Chimera.” One subject was particularly intrigued, discovering complex features like multi-trial and multi-block workflows and branch events without prompting.

![Figure 10: Aggregate results of the closing questionnaire on a 5-point Likert scale. Error bars show ±1 SE.](image)

Regarding the workflow editor, users appreciated its “flexibility” and that its “timeline implies [an experiment’s sequence].” Furthermore, they liked the visual design of block widget, finding it “easy to use.” For the trial editor, subjects commented, “it's easy and straightforward to manipulate states in trial[s], and the logic between states.” Most participants found the hierarchy of editors easy to understand and use.

Users' open feedback on the general experience was positive. One participant told us, “A lot of things become much simpler, especially if you've used PowerPoint,” mentioning the system’s resemblance to existing common visual editing tools. One psychology researcher commented that e-Chimera IDE is “simple and intuitive. I don't need to write code. No need to program, easy to design.”

Participants suggested new features and identified several usability issues. Some wanted reuse already-built components to lessen the amount of repeated work required. We believe such a mechanism could both speed up future experiment design and make complex experiments more elegant in the e-Chimera IDE.

Users suggested richer transition capability, for example within one screen or between trials. The current visual language enforces between-screen transitions within the same trial only. One user also encouraged us to simplify e-Chimera by hiding unused Eclipse features from Eclipse. One told us, “Eclipse provides a lot of unnecessary functionality for our system and also feels chunky. A dedicated editor, for commercial or widespread distribution to psychology researchers, would be best.”
9. FUTURE WORK

Our current research is a first step in providing effective toolkit support for mobile experimentation. For the e-Chimera IDE, we are interested in incorporating several enhancements. These include defining and reusing already-built experiment components, polishing the design of workflow editor, supporting semantic zooming for large screen Internet tablets, and enabling background triggers. The current e-Chimera currently supports only widely-used stimulus types and user response mechanism from classic psychology experiments. However, we plan to modify both the e-Chimera IDE and runtime to introduce support for rich mobile sensors, such as microphone, camera, GPS, and accelerometers. We would do this by extending the e-Chimera visual language and providing an interface to integrate the capabilities of third-party authoring tools such as Examplr [Hartmann et al 2007] and Gesture Studio [Lu and Li 2013]. For the server, we could add fine-grained access control interfaces, in-the-air code updates, and more visualization of experiment progress.

Most importantly, we would like to conduct longitudinal studies to explore how domain researchers can leverage e-Chimera to run large-scale mobile experiments in the wild. In particular, we are interested in finding how we can use e-Chimera to explore the impact of different motivation and scaffolding mechanisms in mobile learning through collaborative research.

10. CONCLUSION

In this paper, we present e-Chimera, a visual end-user programming environment for designing and prototyping experiments on mobile devices. The e-Chimera IDE provides a statechart-based, multi-level visual language to create experiments at the workflow, trial, and screen interaction level. The e-Chimera runtime, driven by a Domain-Specific Language (DSL), enables high fidelity, cross-platform application deployment. Overall, e-Chimera integrates design, testing, deployment, and analysis of mobile experiments.

REFERENCES

Percept Mot Skills. 44(2), April 1977.


XAMARIN (previously MonoDroid), 2012, http://xamarin.com