A Framework for Remote Usability Evaluation on Mobile Devices

Bachelorarbeit in Informatik
Daniel Bader
Introduction

Usability
- An indicator for the ease of use and acceptability of a system

Usability Evaluation
- Methods for determining the usability of a system
- Usability data are collected and analyzed by human evaluators

Common activities (Ivory and Hearst, 2001)
Introduction

Automated Usability Evaluation
- Some or all phases of an Usability Evaluation are automated
- Various degrees of automation exist

Remote Usability Evaluation
- Evaluators and test users are separated in space and/or time

Automated Remote Usability Evaluation
- One of the “ultimate goals” in the field of usability evaluation
Introduction

Usability Evaluation is especially important on mobile devices
- User interfaces on mobile devices became more complex
- Mobile context creates new challenges:
  - different environments, distractions, ...

Current mobile devices have a rich ability to track their surroundings
- Microphone, GPS, accelerometer, video camera, ...

Usability Evaluation can be (partly) automated
Problem statement

A lack of software support for Usability Evaluation on current mobile platforms
- No automation support in existing tools
- No support for built-in tracking capabilities (microphone, accelerometer, GPS, ...)
- No support for collecting user interactions graphically (screenshots, video)
Proposed solution

The **muEvaluationFramework** *(mobile usability Evaluation Framework)*

An application-independent software framework for remote usability evaluation on mobile platforms which supports automation in all phases of a usability evaluation.

![Diagram of common usability evaluation activities]

- Capture
- Analysis
- Critique
Outline

✓ Introduction
✓ Problem statement
  • Proposed solution
    – Requirements specification
    – System design
    – Object design
  • Prototype demo
  • Future work
Functional requirements

Capture phase
- Session-based usability data collection
- Support for multiple usability-data sources
  - User input events
  - Application events
  - Device sensor events
- Live preview for collected events
Functional requirements

Analysis phase
- Interpretation of the captured data
- Detection of usability problems
  - e.g. Human Interface Guidelines (HIG) violations
- Usability summary generation
  - Most used views and UI widgets
  - Heat maps for user input events
  - ...
Functional requirements

Critique phase
- Report generation
  - Reports summarize the interpretation results
  - Reports can be viewed using a web browser
- Report configuration
  - Selection of the report sections
Nonfunctional requirements

Minimal setup work required by the developer
- Low entry barrier
- Quick setup for existing projects
- Goal: “link against one library and make one method call”

Wireless communication with the mobile device
- Test users can move freely and do not feel restricted
- Evaluation can be performed outside of a laboratory

Extensibility
- Easily add new sensors, interpreters, and sections
- Framework intended as a research platform
Use case model
Object model and Dynamic model

Conceptual sub-models for each of the three phases
Object model and Dynamic model

Conceptual sub-models for each of the three phases
Capture phase

The life cycle of an **EvaluationSession**
Capture phase

- **MobileDevice** runs on **HostApplication**
  - HostApplication interacts with **TestUser**
    - TestUser uses **EvaluationSession**
      - EvaluationSession initiates **Developer**
      - EvaluationSession participates in **UsabilityData**
        - UsabilityData collects **Sensor**
          - Sensor monitors
            - Sensor attaches to (t : SensorTarget)
              - Sensor enables
                - Sensor disables

**EvaluationSession**
- start()
- stop()
- selectSensors(sensorSet : Sensor [])
Capture phase

Sensors and Events
Capture phase

Sensor specializations
Capture phase

Event specializations
Object model and Dynamic model
Interpreters work on data collected in the Capture phase and generate InterpretationResults.
Analysis phase

Interpreter specializations

- ViewChangesInterpreter
- ScreenshotInterpreter
- UIGuidelineChecker
- ViewDurationInterpreter
- TouchHeatmapInterpreter
- OverviewInterpreter
- AnalysisStatisticsInterpreter
Object model and Dynamic model
Critique phase

- Developer views Report
- Report summarizes InterpretationResult
- Report recommends changes to HostApplication
Critique phase

Report document model
High-level dynamic model

TestUser

Capture

SessionJournal

UsabilityData

Events

Analysis

ResultStore

InterpretationResults

Critique

Report

Inputs and outputs of the three phases
System design

Subsystem decomposition of the framework
System design

Deployment of the framework
Object design

- Capture
  - CaptureLibrary
  - CaptureServer
- Analysis
  - AnalysisController
  - Interpretation
  - ResultStorage
- Critique
  - ReportController
  - SectionGenerator

Visual Paradigm for UML Standard Edition (TUM - Institut für Informatik - Lehrstuhl 1)
CaptureLibrary - Method interception

UIWindow sendEvent() behavior before method interception
CaptureLibrary - Method interception

UIWindow
+sendEvent()
+...
+...

Event

Original implementation of sendEvent()

UIWindow sendEvent() behavior before method interception
CaptureLibrary - Method interception

**UIWindow**

+sendEvent()
+...
+...

Event

Original implementation of sendEvent()

**UIWindow sendEvent()** behavior *before* method interception
CaptureLibrary - Method interception

UIWindow
+sendEvent()
+...
+...

Original implementation of sendEvent()

Our implementation of sendEvent()

UIWindow sendEvent() behavior after method interception
CaptureLibrary - Method interception

UIWindow

+sendEvent()
+...
+...

CaptureLibrary

Original implementation of sendEvent()

Our implementation of sendEvent()

UIWindow sendEvent() behavior after method interception
CaptureLibrary - Method interception

UIWindow
+sendEvent()
+...
+...

Original implementation of sendEvent()

Our implementation of sendEvent()

CaptureLibrary

UIWindow sendEvent() behavior after method interception
CaptureLibrary - Method interception

UIWindow
+sendEvent()
+...
+...

Original implementation of sendEvent()

Our implementation of sendEvent()

Event CaptureLibrary

UIWindow sendEvent() behavior after method interception
Object design
Analysis subsystem

Usage of the **Blackboard design pattern** in the Analysis subsystem
Analysis subsystem

Dependencies between Interpreters are resolved automatically

Generated order of execution:
Analysis subsystem

Dependencies between Interpreters are resolved automatically

Generated order of execution:

\{C\} → \{B\} → \{A, D\}
Object design
Critique subsystem

```
 Critique subsystem
```

```
Critique subsystem

Element
- body : String

Element

Section
- title : string
- header : string
- footer : string
- elements : Element[]
+ __init__(title)
+ add_element(e)
+ generate_html(result_store : result_store, filename)

Section

Report
- title : string
- header : string
- footer : string
- sections : Section[]
+ add_section(s)
+ generate_html(result_store : result_store) : string
+ generate(result_store : result_store, filename)

Report

ReportGeneratorFacade
- reportFilename : string
- eventsFilename : string
- interpreters : string[]
- sections : string[]
+ addInterpreter(i : string)
+ addSection(s : string)
+ setEventsFilename(s : string)
+ setReportFilename(s : string)
+ generateReport()

ReportGeneratorFacade

Facade
```

```
UIGuidelineIssuesSection
ViewDurationSection
TouchHeatmapSection
NavigationPathSection
OverviewSection
```
Prototypical implementation

Full vertical prototype for Apple iOS
- Implemented in Objective-C and Python
- Tested with the applications *Wordpress for iOS* and *PlainNote*

Capture support
- Automatic network setup using Zeroconf/Bonjour
- Custom, stateless XML-based network protocol
- Data transmission is performed on background thread

Analysis support
- Six interpreters were implemented

Critique support
- Generates HTML output
- Five section types were implemented
Demo
Future work

Add more sensors and interpreters
- e.g. audio and video recording

Use the framework in a real world scenario

Add support for Google Android
- Java has rich support for reflection

Add ability to playback sessions
- Play back captured user interactions on the device

Integrate framework in FastFix
- Perform usability error detection
- Provide access to user interaction data and context information
Thank you!

Questions?
Backup slides
Use case model (with refinements)
Capture phase

Sensors and SensorTargets
Capture phase

Event detection example
Capture phase

- **ActiveSensors**
  - **Initialize Sensors**
  - **Initialize SessionJournal**

  *After Initialize Sensor, Observe SensorTargets for each Sensor*:
  - **Event**: Create Event
  - **[targets did not change]**: Continue with next Sensor
  - **[targets changed]**: Go to SessionJournal

*For eachSensor in ActiveSensors:*
- Observe SensorTargets
- Create Event
Analysis phase

Run analysis phase

Initialize ResultStore → Initialize Interpreters → Run Interpreter

Update ResultStore → ResultStore

<<structured>>

Loop

SessionJournal → Event → InterpretationResult
Critique phase
System design

The **Capture** package
System design

The Analysis package
System design

The Critique package