Software Requirements Specification

for

Ptolemy on Android

Version 1.0

Prepared by Team HandSimDroid

Carnegie Mellon University

02/09/2011

Document Approval

Elizabeth Latronico, *Bosch RTC*	Date
Charles Shelton, *Bosch RTC*	Date
David Root, *Project Mentor*	Date
Philip Bianco, *Project Mentor*	Date
Anar Huseynov, *Team HandSimDroid Member*	Date
Peter Foldes, *Team HandSimDroid Member*	Date
Justin Killian, *Team HandSimDroid Member*	Date
Ishwinder Singh, *Team HandSimDroid Member*	Date

Revision History

Name	Date	Description	Version
A. Huseynov, J. Killian, P. Foldes, I. Singh	1/28/2011	Initial Version	0.1
P. Foldes	2/9/2011	Added change control	0.2

Table Of Contents

Introduction

1.1 Purpose

The purpose of this document is to document the functional and non-functional requirements for the Ptolemy on Android project. The Ptolemy on Android project is a project to port Ptolemy, an open source modeling and simulation tool, to run on the Android OS and handheld hardware with a user-customized interface.

1.2 Document Conventions

In the "System Features" section, the priority of all system features will be assumed to be high since the project is not complete without them. Each system feature will include an itemized requirement list. Priority of the requirements listed there will be tracked in a separate document and updated as the project progresses.

In the "Use Cases" section, some phrases are bold faced. This is an indication that the definition of the phrase is included in the glossary section.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document is primarily the clients of the project, Bosch RTC, the MSE mentors, and future MSE students.

This document makes reference to other project artifacts, and the reader can locate them by following links available in the "References" section.

1.4 Product Scope

Our project is to enhance the existing Ptolemy application and create a system that is able to run simulations on an Android handheld device. Ptolemy is an open-source modeling and simulation tool that currently runs on the personal computers (PC) and is undergoing research by the Bosch RTC to model, optimize, and simulate the embedded software they develop. Our client, the Bosch RTC, envisions that this software would be useful for reading, simulating, and calibrating data that is being developed or already running through embedded systems. In the vision, a handheld system would provide its users with agility that they currently do not posses. Team HandSimdroid has been tasked with discovering, demonstrating, and exploring the possibility and usefulness of such a system.

Since handheld devices have limited screen real-estate and largely touch screen based input methods, the current user interface of Ptolemy would not be usable on the handheld device. One of the challenging aspects of our project is making the user interface (UI) elements of the desktop application presentable on a smaller screen. We have to design the UI of the handheld application in such a way that we divide the UI of the application in small, manageable chunks so that the user is not overwhelmed by the amount of functionality.

At the highest level, the scope of the project is to deliver the following requirements:

- To enable a user to run simulations on an Android-powered, handheld device

- To design a custom user interface for a handheld based on an existing Ptolemy model
- To deliver a solution that will facilitate the development of new visual actors

For more information regarding the scope and deliverable artifacts, please refer to the Statement of Work.

1.5 Change control of the requirements

It is possible that the requirements are going to change after the creation of this Software Requirements Specification. To control and keep track of the changes the following process will be used.

The list of requirements will be accessible by all parties mentioned as the intended audience of this document. It must be under version control. The list will contain at least the following information:

- ID of the requirement
- Description of the requirement
- Date of the last change
- Priority of the requirement

After the signature of this document, any additions, modifications, deletions, or re-prioritizations made to the project must be accepted both by the client (Bosch RTC) and the members of the HandSimDroid MSE student team. The changes must be reflected in the change control table in the appendix and the referenced list of requirements. In the event that the table is filled, the SRS must be reviewed and signed again by both parties.

1.6 References

Statement of Work (SoW)
Usability Report
Use Cases Report
List of Requirements

2 Overall Description

2.1 Product Perspective

Robert Bosch GmbH is a worldwide organization that designs and develops embedded systems for automobiles. Because of the inherent complexity of modern engine systems and necessary precision to ensure safe operation, Bosch uses model-driven development. These systems are currently modeled by the various business units using a tool called ASCET that is developed by a Bosch subsidiary, ETAS. Though ASCET has sufficient capabilities for their current operations, the Bosch RTC has been researching additional capabilities that could be incorporated into ASCET and provide benefit. The project is not to be considered a replacement for the ASCET tool, nor is it expected to be a continuation of the previous MSE team's efforts to include database support within Ptolemy. Instead, the resultant application extensions and simulation demonstrations are to serve as a proof-of-concept of a set of features, particularly the ability to run simulations on-the-go within a customized user interface, for potential inclusion in Bosch's existing commercial tool.

2.2 Product Functions

The major product functions can be grouped into three affinity groups: the simulation controller operating on Android, the user interface designer tool, and the internal extensions to the existing Ptolemy development framework that will simplify the creation of new display actors after this project has concluded. At a higher level, they represent the user's ability to run and control a simulation in real-time on an Android-powered handheld device, tablets and phones being the current target, and visually display the output within the confines of a user-specific, customized layout. These will be described in greater detail and broken into specific features under the heading, "System Features".

2.3 User Classes and Characteristics

Ptolemy Developers - These users, having some background in software development, would be using the underlying actor display framework developed by Team HandSimDroid to create new visual displays of simulation data that target multiple platforms (Android, desktop, etc).

Model Developers - This class encompasses users that would model real-world systems using the Ptolemy constructs. It is also possible that this user class would design the layout of outputs on the handheld device for distribution to the various consumers.

Handheld User - Though this class was originally characterized by the activities of the calibration engineer at Bosch during contextual design, this user class can apply to any user with a need or desire to run, control, analyze, or demonstrate a simulation on an Android handheld device.

2.4 Operating Environment

The developed solution includes several parts. The application used to configure, operate, and control the simulation on the handheld device, particularly tablets and phones, must use the Android operating system (2.2 and forward) and work within the hardware limitations of the currently available devices. The processing power of the device is not sufficient to perform the processing portion of the simulation and thus, must be offloaded to a more powerful machine. For this reason, the handheld device is required to support WiFi connections. The second and third portions, the layout configuration tool and actor display framework, must be written in Java and conform to the coding and display conventions of the desktop Ptolemy tool and operate under similar environment conditions. Whether the layout tool will be incorporated into Ptolemy or exist as a standalone tool is undecided, but regardless, both enhancements will be checked into the Ptolemy source repository.

2.5 Design and Implementation Constraints

The team is prevented, through design and implementation, from including any third-party components that implement a more restrictive license than that of the Ptolemy simulation software. This is done in order to continue the promotion of Ptolemy as a widespread modeling and simulation tool. Our solution has also, as described later in the document, been limited by the constraints of the hardware available on the market and thus, will be the subject of several experiments as our design and architectural activities progress.

2.6 User Documentation

In addition to the software development portions of the project, Team HandSimDroid will also provide user documentation for configuring Ptolemy to build and use the user interface layout tool and all design/architectural documents that would assist in future development.

2.7 Assumptions and Dependencies

There are several assumptions at the time of this document's drafting that are still being explored. We are assuming that the performance (I/O), memory, battery, and other constraints of the available Android hardware will not prevent us from providing an acceptable solution. We have also assumed that all sensors available on the provided hardware will be fully supported by and accessible through the Android OS and it's collection of APIs. Lastly, we have assumed that, though development is continuing on the Ptolemy Project in parallel with our project, our solution will not be impacted by those enhancements.

3 External Interface Requirements

3.1 User Interfaces

The project will have two user-facing software components, the user interface layout designer and the Ptolemy simulation controller on Android. The user interface layout designer must conform to the current look and feel of the Ptolemy desktop application in order to minimize training time. Since the Ptolemy desktop application's user interface is written using standard Java libraries, the designer tool must be designed in the same way. However, the preview of the layout it will depict must be as close as possible to the user interface of the model layout on the Ptolemy simulation controller application. In addition, the handheld Ptolemy application should conform to the look and feel of Android standard UI elements. The rationale behind this decision is to minimize maintenance cost when the Android OS is upgraded or changed.

3.2 Hardware Interfaces

Part of the system is heavily dependent on specific hardware, namely handheld devices. As the system is geared towards smartphones and tabloids, there are certain expectations towards the hardware components of the system.

The following device types are supported:

- Android mobile phones (for example an HTC EVO 4G)
- Android tablet devices (for example Samsung Galaxy Tab)

Commonly these devices have the following attributes:

- Various dimension screen sizes, usually between 3.4"-10"
- Touchscreen interface
- Integrated sensors. These might include, for example, microphone and accelerometer.
- Able to connect to wireless network through Wi-Fi.
- Minimum of 1 GHz chipset
- Minimum of 512 MB random access memory

3.3 Software Interfaces

Our solution is expected to interact with, extend, and use the foundational classes from Ptolemy II v8.0. The application is a multi-threaded simulation engine written in the Java programming language and thus, any extensions to the framework which aid in visually displaying output to the user must also be written in Java.

3.4 Communications Interfaces

The method of communication, at this time, is still undergoing experimentation (TCP & UDP sockets, XMPP, MQTT). However, remote method invocation (RMI) has already been ruled out as an option as it uses proprietary technology and has limited or no support on our target version of the Android operating system, Froyo (2.2). The architectural and detailed design documentation will be updated as further design decisions are made.

4 System Features

4.1 Create User Interface Layout

4.1.1 Description

The objective of this system feature is to enable a user to create a customized user interface for the handheld device for a model developed in Ptolemy. Since a new user interface is created for each model, layout creation must be a very efficient and easy process. Users must be able to “cook up” the layout in couple of minutes without worrying that the layout is perfectly sized and/or aligned. Hence, correct alignment of individual user interface components, proportions, and exact sizing are secondary aspects to the focus area. The most important usability aspect of the tool is to give user a direct mapping between the Ptolemy model actors and the layout he/she is creating. The user must be able to understand to which actor each individual user interface widget is related. Since layouts will be created for a target handheld device for a specific purpose, a complex layout management mechanism that stretches layout to any screen size is unnecessary. Consequently, user will be able to set or select from a list of predefined, fixed screen sizes during layout process. Since the tool will either be built into Ptolemy or used alongside it, it is necessary to maintain the same look and feel as the Ptolemy application. Also, because of the limited screen real-estate available on handheld devices, the user must be able to group different user interface components into tabs.

4.1.2 Use Case

The use case describing this functionality is included below along with one possible alternative flow. For details regarding use cases of the system, please refer to the Use Cases report.

Use Case Name:	Create User Interface Layout
Use Case ID:	UC1
Primary Actor(s):	Model Layout Creator (aka User in this use case)
Secondary Actor(s):	Storage

Brief Description:	The Model Layout Creator constructs the **user interface layout** for a specific **Ptolemy model** on the PC version and saves it to the storage. Each user interface layout is customized for a specific Ptolemy model. The user drags and drops the appropriate actors, sets their **attributes**, and aligns the **outputs** (such as graphs and console output) to a grid to create a simplified/focused view of a model and its running state.
Pre-conditions:	The Ptolemy model is created and opened in the Ptolemy tool running on the PC
Flow of events:	1. The Model Layout Creator clicks the create user interface layout button 2. The system presents a **hierarchy of actors and their respective parameters** that are present in the current Ptolemy model 3. The user drags and drops these **user interface components** to the **grid** 4. The user arranges user interface elements on the grid by resizing, removing, and moving them around the layout canvas 5. The user can add a tab to the grid and place user interface elements on it 6. The user can move a user interface element from one tab to another by dragging or copying it 7. The user clicks ‘Save’ to save **user interface layout** as a **configuration file** along with **Ptolemy Model File** to storage **Note:** Steps 2-6 are iterative and can be repeated.
Post-conditions:	The user interface layout configuration file is created
Priority	High
Alternative Flows and Exceptions:	● The user tries adding an actor that is not integrated with layout system **(**See alternative flow of this use case) ● The user adds too many elements that don’t fit the grid/screen**.** System presents an appropriate error message and does not allow further addition of user interface elements. ● The user tries to save an empty layout. In this case, system displays the appropriate error message. ● The user overlaps user interface elements. The system uses some kind of user interface elements to emphasize where the overlap happened (by changing border colors of user interface element, for example) and displays the appropriate error message.

Non-behavioral requirements:	● The user must be able to easily create a layout without spending too much time on this activity (15-30 minutes depending on complexity) ● The user interface for layout creation must be very simple and require little or no training for anybody familiar with software packages similar to Visio
Assumptions:	● The grid is of a fixed size and user interface layout does not need to be stretched ● All actors have meaningful names so that user understands what to pick and add ● Actors are developed using a layout system that will expose functionality of an actor related to how to display and interact with it. See *Integrate Ptolemy Actor into layout system* use case for details.
Issues:	● What is the minimal set of functionality needed to successfully run the Ptolemy model? ● What is the target dimension of user interface layout? ● Does layout need to include any tabs for grouping elements better?
Addl. Requirements:	System should be able to automatically calculate optimal **dimensions** of all user interface elements to allow user arrange them on the grid.
Source:	Client meeting 09/24/2010, Client meeting 10/08/2010, Team meeting 09/07/2010, Ptolemy Maintainer (Berkeley) meeting 10/14/2010, Ptolemy documentation

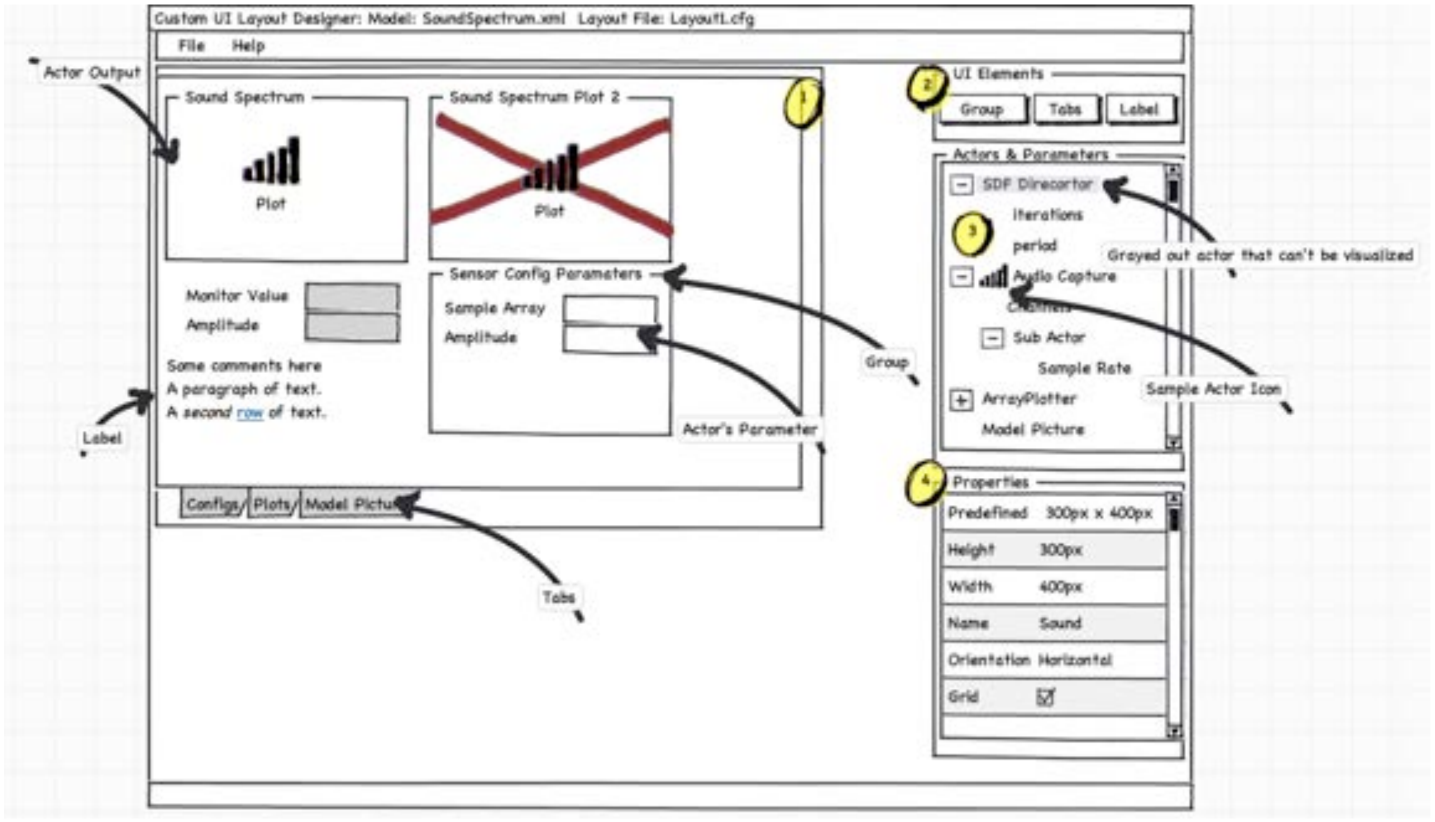
ALTERNATIVE FLOW

Alternate Name:	Reject actors not integrated into layout system
Use Case ID:	UC4
Primary Actor(s):	The Model Layout Creator (aka User in this use case)
Brief Description:	The Model Layout Creator tries to add actor to the user interface layout that is not integrated with the layout system.
Insertion Point:	Step 4, the user drags user interface components onto the grid.
Pre-conditions:	The user interface layout creation process is started

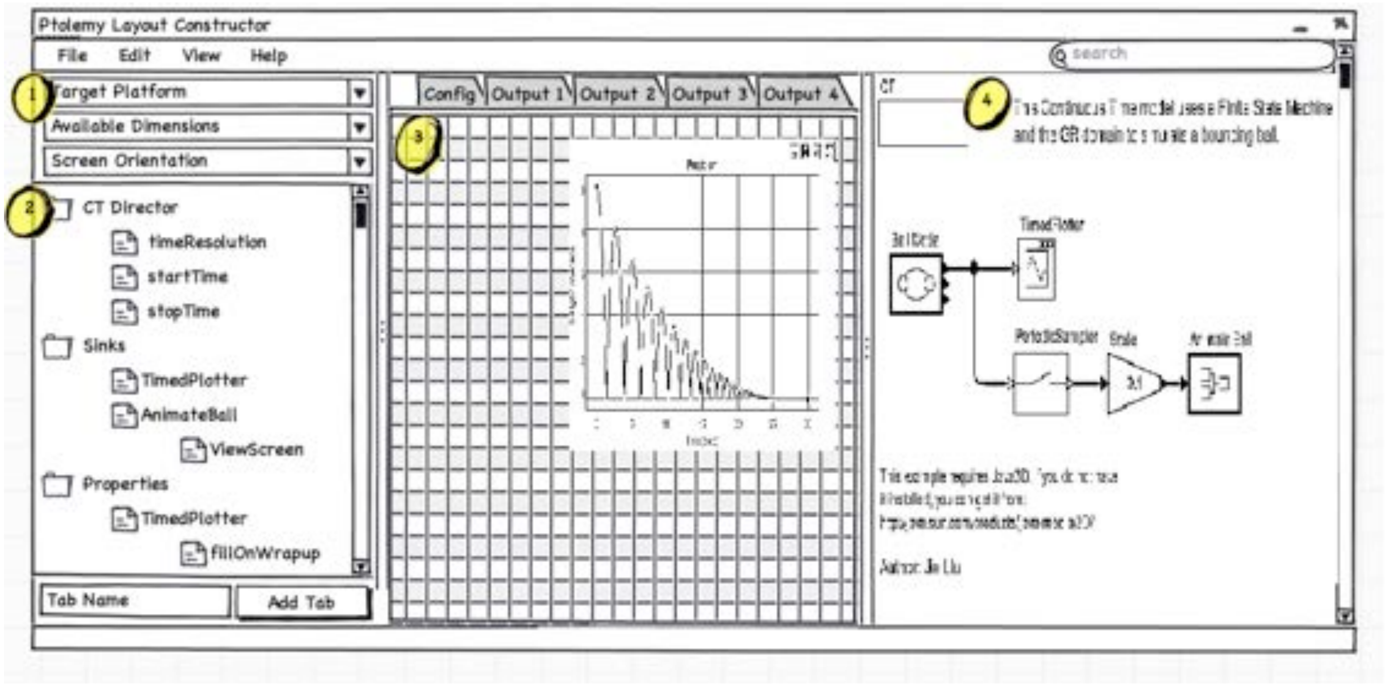
Alternative flow:	1. The Model Layout Creator selects an actor that is not integrated with the layout system 2. The user drags and drops that actor or its attributes to the layout 3. The system displays an error message and does not add the actor to the user interface layout
Post-conditions:	The actor is not added to the layout
Priority	High
Non-behavioral requirements:	An error message must be simple and user should not be distracted from the task he/she is doing
Assumptions:	The layout system is in place and not all Ptolemy actors available in the system are integrated with it
Issues:	Is it possible to implement minimal layout system integration for all Ptolemy actors?
Addl. Requirements:	The system can detect whether an actor supports layout system or not
Source:	Client meeting 10/08/2010, Ptolemy Maintainer (Berkeley) meeting 10/14/2010

4.1.3 Low Fidelity Prototypes

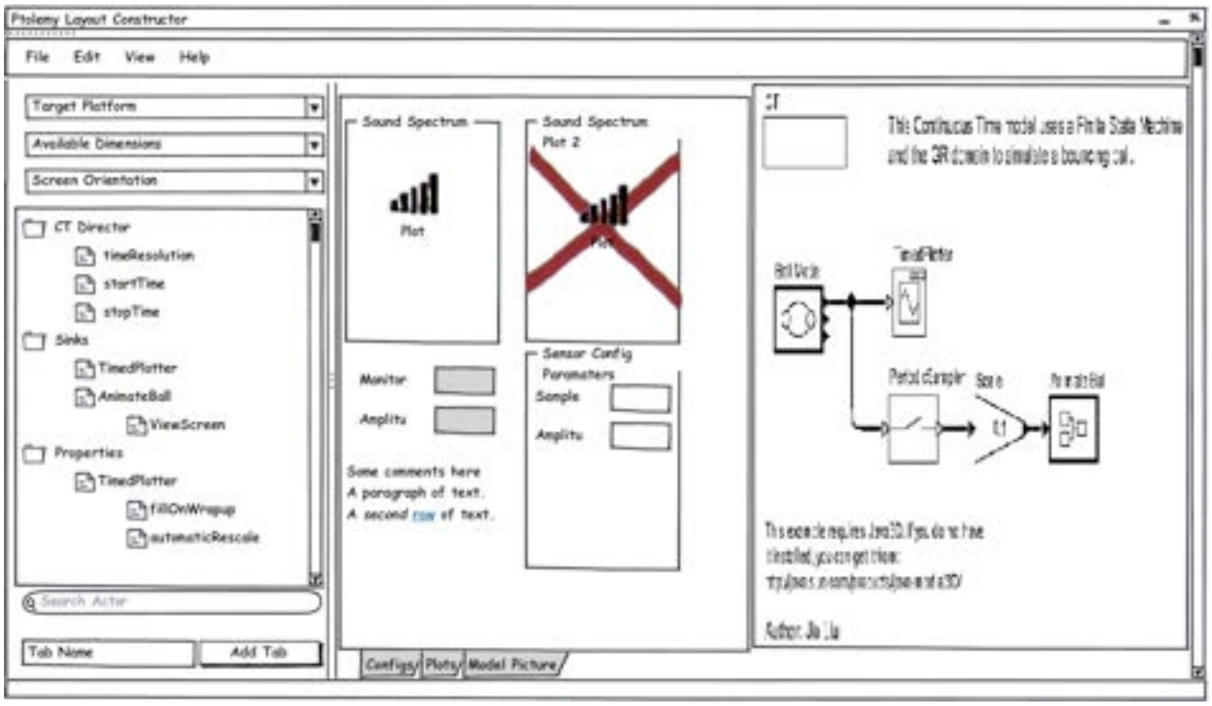
Three low fidelity prototypes were created that depict the potential user interface of the layout tool. The first two prototypes were done in parallel by two different team members. The final one is a consolidated prototype that also incorporates the clients' feedback. For details, please refer to the Usability Report which contains the description of functionality performed by each prototype, analysis of usability issues, and trade-offs associated with them.



Prototype 1: The detailed description of the prototype can be found in Usability report, Focus Area #1: User Interface Layout Designer, Prototype #1 section.



Prototype 2: The detailed description of the prototype can be found in Usability report, Focus Area #1: User Interface Layout Designer, Prototype #2 section.



Some comments here
A paragraph of text.
A second row of text.

This Continuous Time model uses a Finite State Machine and the SR domain to simulate a bouncing ball.

This example requires Java3D. If you do not have it installed, you can get it from: http://java.sun.com/products/java-media/3D/

Author: Jie Liu

Consolidated Prototype: The detailed description of the prototype can be found in the Usability report, in the Conclusion section.

4.2 Integrate Actor

4.2.1 Description

This feature enables a Ptolemy developer, a user responsible for creating and modifying actors to be used in constructing Ptolemy models, the ability to integrate new actors with visual output into the handheld application. The underlying framework is expected to empower visual display actors to draw themselves on a variety of platforms, though Android is the only target for this project.

4.2.2 Use Case

Use Case Name:	Integrate Ptolemy actor into the layout system
Use Case ID:	UC3
Primary Actor(s):	Ptolemy Developer (aka Developer in this use case)
Secondary Actor(s):	n/a
Brief Description:	Since Ptolemy actors that are developed by third party Ptolemy developers don't have the ability to be visualized with custom user interface on Android, the Ptolemy developer needs to integrate them into the new layout system. They will use a layout system **API** to do so. The objective is to extend/expose the user interface capabilities of the

	available Ptolemy actors so that they can be used on Android platforms, such as handheld devices.
Pre-conditions:	● Ptolemy Actor is available in the Ptolemy applications both on Android and PC versions ● The current Ptolemy code is available to the developer ● Developer is familiar with API of the layout system
Flow of Events:	1. The Ptolemy developer opens the Ptolemy **code base** of the Ptolemy actor that needs to be integrated in layout system 2. The developer integrates current Ptolemy actor code into the layout system by exposing certain functionality or by refactoring Ptolemy actor's code using layout system API 3. Ptolemy Developer recompiles Ptolemy (both PC and Android version) to include new actor integrated with layout system
Post-conditions:	● Ptolemy actor is integrated into the layout system, therefore it has user interface code for android platforms. ● Model Layout Creator who has a PC version of Ptolemy with above stated Ptolemy actor can use that actor in creating user interface layout ● The handheld user that has Android version of Ptolemy with above stated Ptolemy actor can run simulation on the handheld and see custom user interface developed by the Ptolemy Developer
Priority	High
Alternative Flows and Exceptions:	● The Ptolemy actor code is not portable to Android due to software/hardware restriction ● The Ptolemy actor is using sensors that layout system does not support
Non-behavioral Requirements:	● Ptolemy actor's code can be extended relatively easily using predefined layout system's API ● Layout system should mimic current implementation of Ptolemy actor's user interface such that developers won't need to refactor lots of code ● Ptolemy actors integrated with layout system must be easily extensible as requirement for open source project. Ptolemy Developers don't need to spend too much time exploring the API ● Complexity must be comparable to current API for Ptolemy actor development. ● Integration process should not to break code of other components. Consequently, layout system must have high

	cohesion and low coupling so that minimal amount of dependent files needs to be changed. In an ideal case, just file containing the actor code changes.
Assumptions:	● The Ptolemy actor user interface code can be extended ● The newly developed code adheres to the coding standards of Ptolemy so it can be shared back with Ptolemy community ● Not all actors need to be integrated with the layout system ● Deployment of the newly recompiled version of the Ptolemy applications is outside of the scope of the project
Issues:	● Should layout system be platform-agnostic or should it be specific to Android? ● Which parts of code base can be affected when actor is integrated with layout system?
Source:	Client meeting 09/24/2010, Client meeting 10/08/2010, Team meeting 09/07/2010

4.3 Run Simulation

4.3.1 Description

This feature allows the user to execute the simulation of a Ptolemy model and view the output on the Android application. After the UI configuration of the model is created using the UI layout tool, then the model is transferred to the Android handheld. Now the users can run the simulation of the model and view/analyze the results in the way that was specifically targeted for their use. Since Android handhelds come in different screen sizes, when creating the UI layout of the model, the user will decide the size of the screen on which the model will run.

4.3.2 Use Case

Use case describing the requirement is included below. For details regarding use cases of the system, please refer to the Use Cases report.

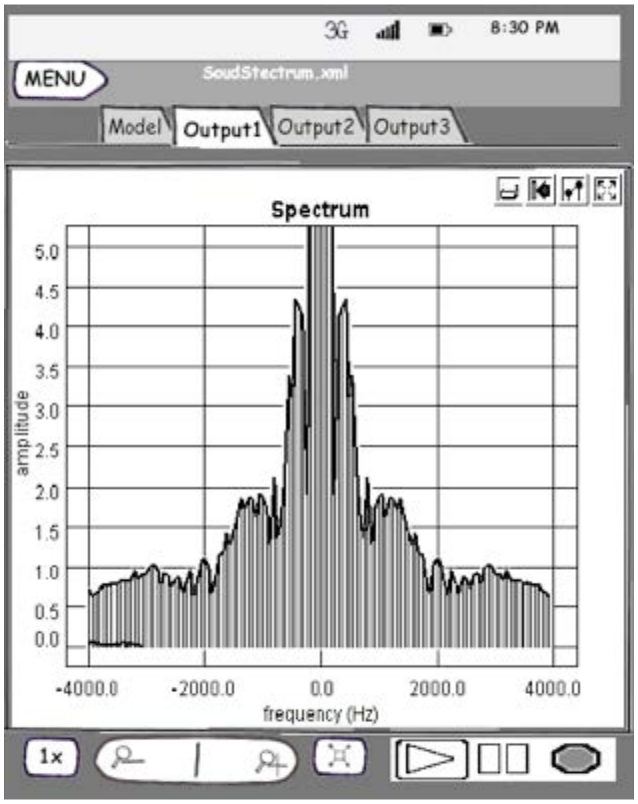
Use Case Name:	Run Simulation On Handheld
Use Case ID:	UC2
Primary Actors:	Handheld User (aka User in this use case)
Secondary Actors:	Sensor, Server
Brief Description:	The handheld user runs the **simulation** of a model on a handheld device with a customized user interface and with possible interaction with from the sensors.

Preconditions:	The **Ptolemy application on the handheld** has access to the server the running simulation of the model and for storing the outputs.
Flow of events:	1. The handheld user launches the installed Ptolemy application on the Android handheld device 2. The user connects to the Ptolemy server 3. The user selects the model that he/she wants to run simulation with. 4. The system layouts user interface based on the model's user interface layout configuration file. 5. The user configures the sensor to accept input from environment if the model requires a sensor for running 6. The user inputs data to input parameters exposed by the user interface layout configuration 7. The user enters required parameters before starting the simulation 8. The user starts simulation and based on the model and user interface layout, different graphs might be plotted on the screen. 9. Server and handheld communicate by sending data streams to each other 10. Any time when the simulation is running, the user can pause the simulation, for example, to alter some parameters
Post Conditions:	The simulation is run and relevant output data is analyzed and might be saved to the remote storage for sharing
Priority	High
Nonbehavioral requirements:	1. The running speed of the simulation should be fast enough to be comparable to the speed on a PC 2. User interface of the application must be catered for a touch screen use
Alternative Flows and exceptions:	● In case the Ptolemy model is badly constructed and is not executable, the system displays appropriate error message ● The system can detect if the data received from the attached sensor is not compatible for the model input. The user is notified accordingly. ● If invalid input is inserted into parameter, the system detects that and alert user to re-enter a valid value. ● In case if data stream from the server is laggy, handheld continues to operate correctly but warns user about that ● In case if data stream (sensor data) from the handheld is laggy, the server continues to operate correctly, skips or extrapolates

	missing sensor data frames, and the user is notified about the problem
Assumptions:	The processing power of a handheld device is sufficient enough to handle the simulation
Issues:	● What are the minimum **hardware** requirements for the handheld to be able to run the simulation? ● What kind of sensors can be connected to a handheld? ● What happens when users opens model layout that does not fit screen of handheld? ● What happens when there are not enough hardware resources?
Additional Requirements:	There is a continuous network connection between the handheld and the server when simulation is running
Sources:	Client meeting 09/24/2010, Client meeting 10/08/2010, Team meeting 09/07/2010

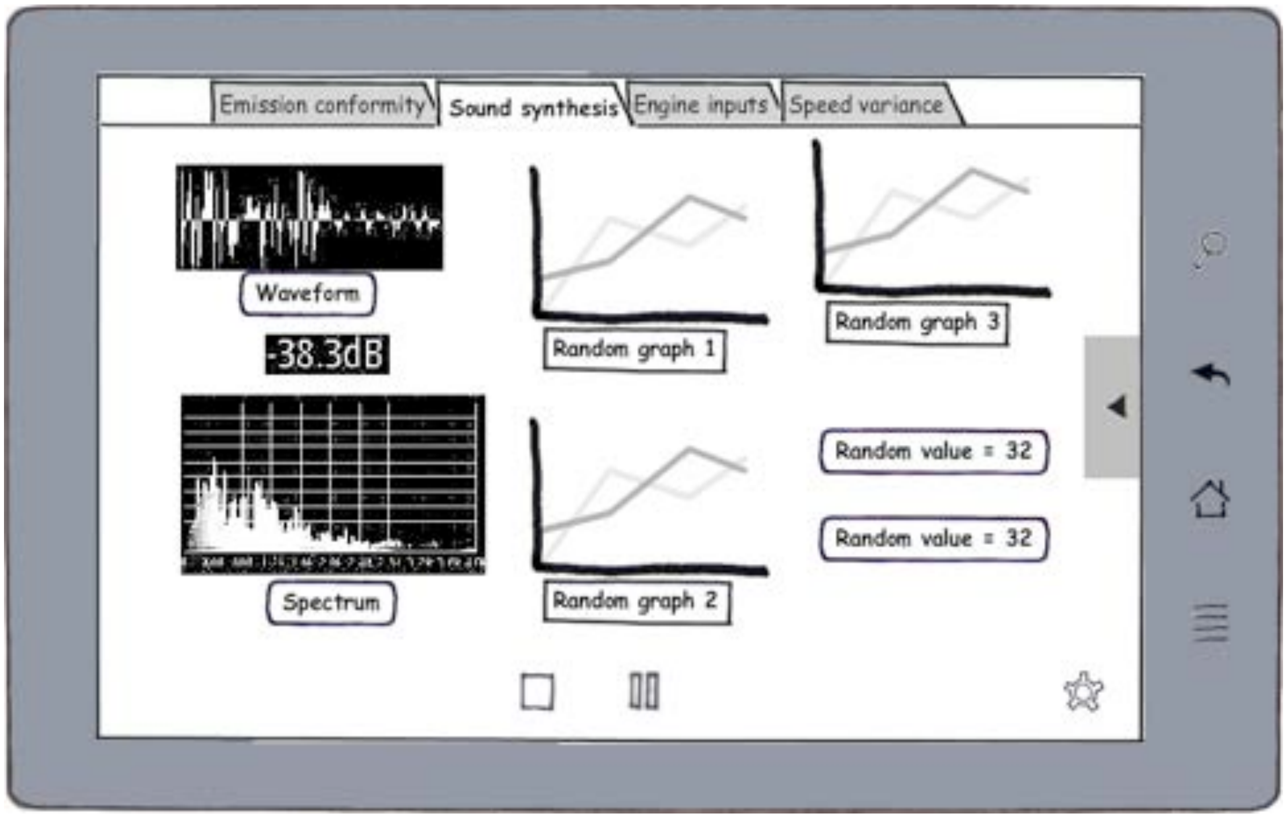
Low Fidelity Prototypes

The two low-fidelity prototypes were created that depicts user interface of the tool in parallel by two different team members. The following prototype depicts a possible UI that is run on a smaller screen such as a mobile phone.



Prototype 3: Description of the prototype can be found in Usability report, Focus Area #2: Model on Handheld, Prototype #3 section.

The following prototype depicts a possible UI that is run on a tablet like device with a larger screen.



Prototype 4: Description of the prototype can be found in Usability report, Focus Area #2: Model on Handheld, Prototype #4 section.

5 OTHER NONFUNCTIONAL REQUIREMENTS

This section provides the non-functional requirements of the system. These requirements reflect the systemic properties that are expressed with different quality attributes. To verify the system in terms of nonfunctional requirements, quality attribute scenarios are used. These can be found in the main requirements table.

5.1 MAJOR SOFTWARE QUALITY ATTRIBUTES

Non-functional requirements deal with system-wide attributes that are needed for the system. For the system to be validated against these attributes, quality attribute scenarios will be used. These scenarios specify a certain expected behavior of the system given a stimulus under specified circumstances, and expecting a certain response with a measure to be validated against. The scenarios are enumerated in the requirements list.

5.1.1 WOW-ABILITY

This quality attribute is actually an aggregation of different attributes to help achieve a business goal, namely that the system will adhere some requirements during its demonstration.

5.1.2 PERFORMANCE

Parts of the system are performance-sensitive. For example, it’s important when sensor data is captured on the handheld device and fed into the simulation or when the simulation is executing and the handheld needs to visualize it for the handheld consumer. Performance is a major factor in the system.

5.1.3 Extensibility

Since the system will not cover all Ptolemy actors and since Ptolemy actors can be added easily to the Ptolemy system, it is important that this system promotes the addition of new actors. There are also cases when a new hardware comes out with different screen dimensions or a new version of Android comes out that the system needs to support.

5.1.4 Portability

The system is focusing mainly to support Android based devices, but the market of handhelds and similar hardware are improving fast. For the system to be usable in the future, it must take into consideration the improving market of these devices.

5.2 Other attributes

There are certain other system properties that the system must promote, including usability, reliability, and maintenance. These are explained with quality attribute scenarios in the requirements list.

6 Other Requirements

6.1 Legal limitations

As the system uses and adds to the Ptolemy modeling tool, it must adhere to the current Ptolemy license restrictions. The current license an be found at http://ptolemy.eecs.berkeley.edu/ptIIcopyright.htm

7 Appendix A: Glossary

7.1 Actor glossary

Actor Name: Model Developer
Description: Model Developer is the person familiar with Ptolemy and it's simulation capabilities. This is the user that constructs Ptolemy models using the available actors and distributes (if necessary) to those responsible for executing and analyzing the results.

Actor Name: Model Layout Creator
Description: Model Layout Creator is a person who is familiar with underlying Ptolemy model and wants to expose relevant parameters and graphs for a Handheld User. This can be a person who created Ptolemy model for a specific user in mind for specific needs that are pertinent to work on the handheld device on the go.

Actor Name: Handheld User
Description: The handheld user is an actor that is responsible for initiating and tracking the model execution (simulation) on an Android handheld device. The user also had the ability to modify the value of the model's input parameters to some extent.

Actor Name: Sensor
Description: Sensor is hardware devise like a microphone that is responsible for taking input from the environment and passing it on the tablet. As an example, this recorded sound could be used as input for the Ptolemy models.

Actor Name: Ptolemy Developer
Description: Ptolemy developer creates custom actors for the Ptolemy for some specific use that is not supported by the software. This can be done either by developing it in Java or composing the actor from multiple actors. For our system, this person will be responsible for integrating his/her actors with the layout system.

7.2 Domain Objects

Domain object	**Description**
Ptolemy Actor	Ptolemy actor is a component or filter of a Ptolemy model that performs certain predefined functionality such as adding several numbers, plotting a graph, or accepting sound input.
Parameter	A Ptolemy actor's input parameter that influences how actor performs its operation
User Interface Element	A general term for Ptolemy actor's parameters, outputs (graphs, logs, file output) and Ptolemy actors themselves.
User Interface Layout Configuration File	Configuration file that contains layout of user interface elements created specifically for some model. User interface layout stores custom user interface for some particular Ptolemy model and allows easy/focused interaction with underlying Ptolemy model.
Hierarchy of actors with parameters	A tree that represents Ptolemy actor as the root element and parameters of that Ptolemy actor as its child leafs.
Output	Output data that is displayed in a form of plot or console output

Grid Layout	Layout where position of user interface elements are constrained by grid nodes (boxes)
Handheld Device	Physical hardware and software combination on which simulations may be run. This is an abstraction away from actual specifications, dimensions, screen layout, and other details particular to one device or another.
Simulation	Running process of a Ptolemy model created to mimic real world environment
Ptolemy Model	File containing interconnected Ptolemy actors that perform certain tasks to simulate some real world environment. The model defines how actors run and behave as a system.
Layout System	System that will expose what user interface elements an actor has, their look-and-feel and behavior. Create User Interface Layout use case depends on this framework for layout creation.
Screen	Screen of a handheld device
Dimensions	Minimum and maximum height and width of a user interface element
Ptolemy Application on Handheld	Application that we will port to run on Android
Output Data	Output data that can be collected while running simulation
Hardware Resources	CPU, memory and other resources that might affect the run time performance of the system.
Layout System API	API that layout system will have so that Ptolemy developers could integrate their actors with the layout system
Code Base	Ptolemy's and Ptolemy actors' Java code

8 Appendix B: Analysis Models

TBD

9 Appendix C: To Be Determined List

- Communication Interfaces
- Analysis Models

10 Appendix D: Requirements change control table

Name	Date	Description of change	Reason of change	Signature of the liaisons of both parties
P. Foldes, A. Huseynov	2/9/2011	Created initial version of the requirements list documentation	To adhere with the SRS.	