Android Chess: A Real Time Collaborative Software Application for Multiple Mobile Clients

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering

By

Eduard S. Sarkisov

December 2011
The thesis of Eduard S. Sarkisov is approved:

_________________________________________  _______________________________________
Shan Barkataki, Ph. D.                      Date

_________________________________________  _______________________________________
Jack Alanen, Ph.D.                           Date

_________________________________________  _______________________________________
Richard Covington, Ph. D., Chair            Date

California State University, Northridge
Table of Contents

Signature Page .................................................................................................................... ii
Abstract .............................................................................................................................. vi
Chapter 1: OBJECTIVE AND GOALS ............................................................................. 1
Chapter 2: PROBLEM DESCRIPTION ............................................................................. 3
Chapter 3: BACKGROUND ............................................................................................... 5
  3.1 Why Chess................................................................................................................. 5
  3.2 Why Android & Java............................................................................................... 5
  3.3 Initial Research ........................................................................................................ 6
  3.4 Starting Point .......................................................................................................... 7
Chapter 4: CONTRIBUTION ............................................................................................. 8
  4.1 Contribution to the Field of Software Engineering ................................................... 8
  4.2 Contribution to Android Community ...................................................................... 10
Chapter 5: DEVELOPMENT PROCESS ......................................................................... 13
  5.1 Requirements & Specifications ............................................................................. 13
  5.1.1 General Description .......................................................................................... 14
  5.1.2 Functional Requirements .................................................................................. 17
  5.1.3 External Interface Requirements ...................................................................... 20
  5.1.4 Performance Requirements ............................................................................... 34
  5.1.5 Design Constraints ............................................................................................ 34
  5.1.6 Attributes .......................................................................................................... 35
  5.2 Development Model ................................................................................................ 43
  5.3 Development Milestones ......................................................................................... 43
  5.4 Testing Process ........................................................................................................ 45
  5.4.1 Conversion Testing ........................................................................................... 45
  5.4.2 Job Stream Testing ........................................................................................... 45
  5.4.3 Interface Testing ............................................................................................... 45
  5.4.4 Security Testing ................................................................................................ 46
  5.4.5 Recovery Testing .............................................................................................. 46
  5.4.6 Performance Testing ......................................................................................... 46
<table>
<thead>
<tr>
<th>Chapter 5: IMPLEMENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.7 Regression Testing</td>
</tr>
<tr>
<td>5.4.8 Integration Testing</td>
</tr>
<tr>
<td>5.4.9 Acceptance Testing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6: LESSONS LEARNED</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 High Level Summary</td>
</tr>
<tr>
<td>6.2 State Diagram</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 7: CONCLUSION</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Chapter 8: CONCLUSION</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Chapter 9: FUTURE WORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 Recommendations for Future research</td>
</tr>
<tr>
<td>9.2 Working toward unfulfilled goals</td>
</tr>
<tr>
<td>9.3 Real live applications</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEMO</th>
</tr>
</thead>
</table>
ABSTRACT

Android Chess: A Real Time Collaborative Software Application for Multiple Mobile Clients

By

Eduard S. Sarkisov

Master of Science in Software Engineering

This project explores the feasibility of effective and stable implementation of a paired client/web-based software package that is integrated among multiple mobile and stationary computing devices with such devices utilizing various operating systems. It takes an existing software application, adds collaborative web-server driven capabilities while exploring porting it to various platforms and operating systems using a single development approach.

In pursuit of the goals outlined above a large reverse engineering effort has been undertaken to understand existing open source software, setup the necessary development environments, add additional development libraries to have the software compatible with Android OS 2.2, create all of the necessary design and requirement documentation from scratch to support software development, and reprogram existing code base while also adding new functionalities which in total amounted to an increase in the code base by about 75%. Proper documentation has been provided turning this project into a case study of software engineering processes within a realm of small open source projects.

The creation of the necessary design and requirements documentation is discussed in detail and the project now provides elements of a case study of software engineering processes within a realm of small open source projects.

Collaboration features added to existing open source software provide a practical implementation of real time collaboration between user groups engaged in a real time participation based on the requirements defined for each of the groups. A web-server feature has also been created to support any number of client-based applications for the purposes of collaboration, data processing, data storage, and user identification. The project demonstrates techniques of off-loading large resource-heavy features to outside web based servers to support less advanced mobile platforms.
Research conducted on the porting and language automation processes determined that the fundamental structure of the original program and tools available were not adequate to support completely general cross platform application. Therefore this project instead focused on collaborative and multiplayer experience within a client-based mobile application while utilizing web server to eliminate additional burden on a resource-limited mobile platform.

Through the use of an open source Android environment the functionality and scope of the original Chess program has been dramatically increased with addition of various online gaming options, online login and authentication modules, and options to save and open existing games that have been enhanced and added. Also a contribution has been made to a larger community of chess enthusiasts. An unconventional approach compared to typical Chess game software has been introduced (designed after “real-life” gaming experience) by adding an unlimited number of game viewers to a particular game whether live or archived. This enhanced Chess software also supports addition of various community and expert-generated game commentaries and analysis modules in future releases of this software.
Chapter 1: OBJECTIVE AND GOALS

The objective is to provide a proof of concept through a working prototype, of an effective and a stable cross platform software implementation that consists of a client-based installation and a web-based platform. This software prototype will connect multiple mobile and computing devices supported by various operating systems in real time using a web-based server platform as the medium of information exchange and distribution. The software package will support session recognition in order to segment access to the web-based platform and identify groups of users collaborating on a shared software based activity, playing a computer chess game.

The two primary goals that must be achieved by this two tier software implementation are online real-time collaboration by multiple users, and lowered impact on a client’s application with the web-based components taking on a larger burden of information processing based on the resource and hardware limitations inherited by the majority of the mobile devices. Through the use of load balancing techniques, workload is distributed across the web-server and client-based application in a way that allows the web server to specialize in data processing, storage, and user handling while the client-server is used in parsing XML-generated data with a limited number of calls to the web server and applying necessary changes to generate the appropriate GUI experience.

The notion of a collaborative software with an online component, driven by multiple levels of participation (roles), and supported by a client-based installation on mobile computing devices implemented within a prototype chess game could be relevant in more complex mobile applications including office related spread sheet and document producing software, complex design and architecture software packages, and other packages requiring complex data processing and constant collaboration of multiple users on the same activity.

The availability of a web-based component to take on a larger burden of information processing will make the experience and utilization of the complex software applications over mobile devices much more practical and widespread which will increase the presence of online collaborative software overall encouraging human interactions and supporting various business and social processes.

The original secondary goal was also to provide a client-side application compatible with various computing devices, architectures, and operating systems. This would create a special purpose application for any mobile platform with no dependencies on proprietary technology utilized through a single development concept. Automated language-porting solutions have been explored; however the fully implemented portable solution was never achieved and the goal was determined to be beyond the scope of this project.
In pursuit of the stated objective, primary and secondary goals, the following lesser milestones have also been realized during design and development phases of this project:

1. Direct contribution to the mobile development community. By utilizing open source software processes and sharing general findings as well as specific solutions discovered during development of the project over publicly accessible web source the goal is to enhance available programming references of the mobile development community and provide an information source for future projects of a similar nature.

2. Gaining experience with the Android development platform, setting up a software workbench and having exposure to integration of various versions of Android-based applications. Since the Android open source mobile operating system is the biggest mobile platform utilized by open source development and has been steadily gaining acceptance over the past 40 months the extensive exposure and development experience with the platform are essential for a software engineer.

3. Demonstration of an open source development process based on the extreme programming development model. Due to the limited resources on this project, the Extreme Programming type of Scrum development methodology has been utilized during design and development.

4. Case study of the development process. Creating through reverse engineering extensive design and specification documentation from scratch as well as going through the full cycle development process allows for a case study of the development process for mobile as well as cross property (mobile & web) applications; the compatibilities and differences between software development and web development; and the intricacies specific to mobile and web properties during the development process.

---

1 “Definition of Scrum” by TechTarget.com @ 2011
http://searchsoftwarequality.techtarget.com/definition/Scrum
Chapter 2: PROBLEM DESCRIPTION

Recent years have seen a great increase in a number of smart mobile devices, including smart phones, with the ability to access the internet, high computing power, and a large enough storage unit. Such devices are currently manufactured using proprietary architecture (see Apple iPhone, Blackberry, and phones that utilize Google’s Android operating system). These mobile devices often take the place of stationary computers or laptops and with the advent of new technologies increasingly strive to utilize the same type of applications (Adobe Photoshop, Microsoft Word, and alike) as the traditional computers.

As the number of smaller mobile devices increases, they take up a larger proportion of the total internet traffic, and more web-based applications are introduced in order to take most of the burden away from the client’s device on to the web. The unique conditions supported by the mobile technology, especially the ability to keep uninterrupted internet connectivity for prolonged time spans, support increasing demand for sophisticated web-based applications. Examples of the most popular web-based applications include Google Docs and Microsoft Office Online.

Currently two of the most important outstanding issues limiting successful utilization of sophisticated online applications are: inadequate computing capabilities of some of the mobile devices (most of the smart mobile devices now on the market are limited to 256 MB or less of RAM size with flash memory not exceeding 5 GB); and the use of proprietary operating systems by manufacturers of mobile equipment which often limits the ability of those devices to communicate with specific web technologies (aka different web plugins) utilized by the online applications (examples include Nokia’s Symbian OS and Apple’s iPhone OS).

Furthermore, a growing demand for ability to share and edit documents in parallel by a number of users adds a level of complexity to web-based applications. As more documents originate and reside on publicly accessible servers, real time collaboration on content and copyright creation becomes a norm and often a necessity. Some of the larger documentation files require groups of people working in parallel editing the same documents.

Traditionally any web–based software should also satisfy the following requirements: stability of web servers, quality of software performance, on demand access to information, adequate levels of data security and reliable data management system.

Therefore, it is likely that successful software applications designed for mass utilization and usage need to meet at least the following set of criteria:

- Web-based software platform (in addition or instead of a client-based solution)
- Secure data sharing and storage
- Compatibility with various software platforms
- Compatibility with various hardware platforms/devices
- Uniformity in functionality, design, and performances.
- Support parallel content editing, creation, and access in real time

In this context, online email systems such as gmail.com provide some of the most successful examples of online applications that largely meet all of the described criteria. These and many other applications depend on internet browsers such as Internet Explorer, Mozilla Firefox, and Safari to enable users to interact with the applications. Hence, a major requirement placed on the mobile hardware is to be able to support specific internet applications as a prerequisite to utilization of any third party online software.

The proposed solution will eliminate the need to rely on third party online software and instead will focus on a less expensive open source client based application.
Chapter 3: BACKGROUND

The aim of this project is to provide a proof of concept through a working prototype of an effective and a stable cross platform software implementation that consists of a client-based installation and a web-based platform. This software prototype needs to be able to connect multiple mobile and computing devices supported by various operating systems in real time using a web-based server platform as a medium of information exchange and distribution. Software is browser independent and supports session recognition in order to segment access to the web-based platform and identify groups of users collaborating on shared software based activity.

The decision has been made that the client based prototype is an interactive chess computer game which allows users to play against a computer, play online against each other and observe and/or analyze games played by others in real time. Initial development is done in JAVA adopted for Google’s Android OS.

3.1 Why Chess.

Chess is a sophisticated game with a well-established set of rules, static interface, finite possibilities, and a large following. It is popular among players of different demographics, cultures, and ethnicities. The mathematic nature of the game insures its compatibility with programming algorithms and code base solutions. Furthermore its semantics are well understood and written notation is compact, uniform, and able to describe the state of the game.

3.2 Why Android & Java.

Android is a Linux-based platform from the Open Handset Alliance. It is supported by over 34 major software, hardware and telecoms companies. The Linux kernel is used as a hardware abstraction layer (HAL). Application programming is primarily done in Java. The Android specific Java SDK is required for development although any Java IDE may be used. Performance critical code can be written in C, C++ or other native code languages using the Android Native Development Kit (NDK). JAVA for Android is a relatively new variation of JAVA and presents a number of unique features suitable for this project.

A. New development in the field of Software Engineering. Android specific JAVA is a relatively recent phenomenon with limited programming samples available in public domain and any contribution/research is valuable to the entire community of developers.
B. Java based code structure should allow for easier further translation to Oracle’s version of JAVA and use on PCs. Supports concept of single development.  

C. Android specific Java implementation is specifically created to support mobile operating systems and deal with mobile specific architecture and resource challenges. It has many performance related libraries specifically built to support mobile hardware. Some of the code written in C, C++ and Objective C could be used through the Android Native Development Kit, and therefore it may be simpler to translate this code, already adopted for mobile hardware/OS into Objective C to run on an iPhone.  

D. Java is a modern object oriented programming language that supports complex solutions while Android OS is up to date and built to support mobile devices. Therefore these two components put together will give the tools and solutions that would not be readily available in other languages and operating systems.  

E. Since this project is done by one person it becomes important to have a solid understanding of the programming language involved and experience solving complex problems using this language. The author’s personal experience with Java is purely academic and concentrates around the course work completed in pursuit of a Bachelors of Science degree in Computer Science at the California State University, Northridge (CSUN); however this experience is diverse enough to allow for a solid understanding of the language and a foundation needed for this project, for example, objected oriented, modular design, networking, GUIs and software engineering.  

3.3 Initial Research  

The initial research concentrated on figuring out major differences between conventional Java and Android Java as well as installing Android SDK, Eclipse, and running simple

---


3 Discussion of Objective-C and Android @ Stackoverflow.com 2010-2011 http://stackoverflow.com/questions/2394236/objective-c-and-android  


5 “Eclipse” by Eclipse Foundation @ 2011 http://www.eclipse.org/org/
Android programs. One clear differentiation factor is Dalvik\textsuperscript{6}, which is a virtual machine (VM) in Google's Android operating system. Dalvik is thus an integral part of Android, which is typically used on mobile devices such as mobile phones, tablet computers and netbooks. Before execution, Android applications are converted into the compact Dalvik Executable (.dex) format, which is designed to be suitable for systems that are constrained in terms of memory and processor speed. Unlike Java VMs, which are stack machines, the Dalvik VM is a register-based architecture. Generally, stack-based machines must use instructions to load data on the stack and manipulate that data, and, thus require more instructions than register machines to implement the same high level code, but the instructions in a register machine must encode the source and destination registers and therefore tend to be larger.\textsuperscript{7}

3.4 Starting Point

Creation of a sophisticated chess program from scratch is outside of the scope of this project. Therefore one of the open source chess projects has been chosen as a starting point for further developments. The project is Honzovy Sachy developed by Jan Němec and can be found on SourceForge\textsuperscript{8}.

The original project has all of the major GUI components and game rules developed. It contained a simple artificial intelligence module which allowed a player to play against computer. It also had Undo/Redo features as well as a Save Game option represented as a static PNG file. Design and other documentation were not available and had to be developed from scratch.

\textsuperscript{6} “Dalvik (software)” by Wikipedia @ 2005-2011 http://en.wikipedia.org/wiki/Dalvik_(software)
\textsuperscript{8} Source Forge is a portal supporting variety of Open Source projects refer to http://honzovysachy.sourceforge.net
Chapter 4: CONTRIBUTION

By taking on a challenge of a cross platform software implementation of a client-based installation and a web-based platform, this paper aims to make a substantial contribution to the mobile and cross platform research community in the field of Software Engineering in general and specific programming contribution to the Android development community. A contribution to the online chess playing community is also foreseeable by creating a direct link from a client based chess tool to collaborating software extended to the online chess playing domain across multiple platforms and enhanced with additional libraries of archived games.

Contributions to the Field of Software Engineering include specific contributions to:

- Client/Server Applications
- Android’s Java Development
- Automated Language Porting and Translation
- Mobile Load Balancing Solutions

Contributions to the Android’s Development Community include specific contributions to:

- Android Client/Server Application
- Android Mobile Chess game development
- Android Online Multi-Player game development

4.1 Contribution to the Field of Software Engineering

Contributions to the field of Software Engineering are a desired outcome of this research. The four contributing factors are: providing concrete example of a two component client/server software application through a working prototype; providing better understanding of the differences between Oracle’s JAVA and Android specific implementation of Java; exploring issues in regards to automated language porting between different versions of JAVA and Object C supported by iPhone’s operating system; and exploring load balancing issues to redistribute the resources between client side installation and web-server.

Client/Server Applications
The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server machine is a host that is running one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. The support that client/server applications provide to collaborative software applications such as online document writing and spreadsheet software as well as peer-to-peer applications is well established and used widely. Some of the mobile examples of such a distributed approach could be found in video sharing applications including Skype and mobile version of peer-to-peer applications such as AIM. However, this approach intends to go further than most of the client/server applications have gone. It introduces collaboration techniques that support a large number of participants per session in real time (most peer-to-peer including Skype are limited in number of participants per session), providing a single client side code base compatible with a number of mobile and traditional computing platforms (which should also minimize the financial burden of developing platform based programming solutions as well as increase time to market turn around), and show that such development is possible by a development team as small as one asset in a relatively short time period.

**Oracle Java vs. Android Java**

As the development of the Android Operating System continues and Dalvik VM used by Android is developing new features such as a just in time compiler, both Android specific Java and Oracle’s Java (which is undergoing its own changes with the introduction of ME and other initiatives) are growing more and more apart. Recent legal challenges set by Oracle to try to standardize Android’s Java and merge both platforms based on Sun’s original set of guidelines have revealed how much further both companies plan to take their products. This project should give a good understanding of how a particular subset of a language can derive its own functionality over a period of time based on a different platform and in the case of Java a different virtual machine implementation.

**Automated language porting and translation**

---

9 Wikipedia “Client-Server Model” @ 2011
http://en.wikipedia.org/wiki/Client%E2%80%93server_model

10 “Oracle vs Google over Java: Android lawsuits may begin to compile” by Larry Dignan, 08/13/2011 @ http://www.zdnet.com/blog/btl/oracle-vs-google-over-Java-android-lawsuits-may-begin-to-pile-up/38019
The idea of a universal algorithm used for automated language porting, especially between families of languages similar in its structure, is not new. There have been a few recent attempts analyzed further in this paper and their success has been varied but so far without any definite breakthroughs. Therefore the goals of this paper have been transformed to analyze the possibility of such cross platform porting, outline advantages of such a process and possible roadblocks and research on various techniques currently available and suited for a larger application across a variety of languages as a part of a uniform porting algorithm.

Mobile Load Balancing Solutions

Defining appropriate load balancing rules and allocating offloading mechanisms supported by a mobile client-side application as well as a remote web-server provide another relevant example based on a set of unique conditions that could contribute to a further research on this matter in the field of Software Engineering. The collaborative nature of this prototype requires offloading of most of the user generated actions to a remote web-server leaving a client application blind to a number of users participating within some of the processes (for example: game viewing) and therefore freeing up limited mobile resource for processing of XML data feeds as well as providing web – server with relevant game playing data. Similarly, a web-server is set to behave in a passive mode without permission to send any communications to a mobile client-side application; instead it only responds to simple http requests by displaying appropriate XML formatted files with relevant data. Such an approach minimizes the need to keep track of a number of connected devices or search for mobile clients that could be unavailable (for example: not connected or resource limited). In addition, a number of calls sent from a mobile device to a web-server are also minimized whenever such is appropriate. A solution has been designed for client-side application instead of directly accessing a web-server database for each move or a user action to request an entire XML file in some instances which is later parsed by the application. This limits the need to keep constant connection with the web-server which is relevant in the unstable mobile internet networks.

4.2 Contribution to Android Community

The Android community is a newly formed entity among software developers. The first production version of the Android OS was released less than 40 months ago and gained a significant following and acceptance from the very start. Today the Android OS is in over 40% of all smart phones in the United States and firmly in the first place among all of the smart phone operating systems in the country. Due to such a rapid expansion of the operating system, the software applications designed for the Android operating systems
have also undergone significant expansion in numbers encouraged by the open source nature of the Android OS. While the number of applications grew, companies involved in this development were usually composed of a few programmers and identified themselves as “start-ups”. Market trends and rapid commercialization of the Android OS impacted development time of these software projects causing popularization of just in time development. Under these circumstances, specifically lack of time for research, lack of time for quality assurance, and the open source nature of the platform at hand, the Android development community became especially reliant on online resources and contributions provide by its members. As a result of this each Android application developer has a unique opportunity to provide a significant contribution to the development community.

The aim of this research project is also to provide significant contribution to the original open source chess game, which it is built on, as well as to the Android programming community and to the development community at large. A dedicated publicly accessible online blog has been setup by the author to post snippets of newly developed and tested code. Some of the general areas of contribution are: android implementation of client/server application, contribution to Android based mobile chess development, and Android multi-player and multi-session game development.

**Android Client/Server Application**

The concept of a client/server application is well established in traditional software development. A set of Android specific challenges, especially in the area of resource management, port management, and data parsing, should be resolved in order to establish the same level of efficiency between client and server components of this application as provided in traditional software development. Many of the existing peer-to-peer and other client/server applications have resolved underlying issues; however the scope of the challenge presented by this application is slightly different, especially in realm of group collaboration and strong emphasis on development for a cross platform client. Introducing programming solutions developed for this application online would enhance overall understanding of this challenge by the Android development community and allow Android developers to utilize open source code samples within their own applications, thus perfecting the original solutions proposed and standardizing programming approaches. Just like this application would use a number of third party developed modules to work through well understood programming challenges resolved by other Android developers and tested by many community members, the new code introduced by this application will be perfected and used by many future Android programmers.

**Android Mobile Chess game development**
This project would contribute to Android Mobile Chess game development by expanding capabilities of the existing open source chess software used as a base line for this project, and introducing chess game specific solutions that could be used by other chess programs. Contribution to Honzovy Sachy, an open source chess program used as a base line for this project would include:

- load and save game functionality enhancements
- multiplayer capabilities
- online games viewing capabilities
- integration with an online portal
- possible introduction of a cross platform client installation (not implemented)

Introduction of a web server as an intermediary between a number of chess client applications serves as an alternative solution that could be used by other chess games developed for Android OS as well as other operating systems and architectures.

*Android Online Multi-Player game development*

Game development process and methodology are well established within the Android community. Many games have been introduced to the Android OS and mobile devices ranging from complex ones developed by companies and groups of programmers to simple games developed by single developers. Multi-Player functionality is a feature available within a limited number Android games; multi-player games seem to be lagging behind on the Android especially given the rapid pace with which the Android market is growing. Most multi-player games developed for Android either single machine games where an Android device is shared among two or more parties (for example: 2 Player Reactor), or true multi-player games shared across a dedicated network (for example: HomeRun Battle 3D). This project takes the multi-player gaming environment a step further than most Android games. It assigns different levels of players within a single game: active players and viewers. It allows for an unlimited number of participants. It allows viewing archived games. And it provides functionality to save a game in order to come back to it later. Introduction of these features through an open source nature of the project will contribute to further enhancements of multi-player functionality within Android development community.
Chapter 5: DEVELOPMENT PROCESS

The following software development process or life cycle required the use of a structure imposed on the development of a software product. There are several key attributes of such processes, each describing approaches to a variety of tasks or activities that take place during the process.

Defining clear requirements is the first and most essential step in any software engineering project. Unfortunately, due to the open source nature of the original Chess program, no documentation was ever discovered, leaving original requirements in the dark. An attempt has been made below to reconstruct some of the original requirements as well as to introduce new ones important to the development of new features of the program. Requirements have been categorized into functional, external interface, and performance. Based on this set of requirements, appropriate specifications have been developed identifying software attributes, necessary system characteristics, and functional specifications.

Based on identified requirements and available resources, an appropriate development model is outlined in this section as well. This model represents a variation of the Extreme Programming software development methodology as is appropriate for open source development as well as development with limited available resources.

Development milestones have been scheduled based on the described specifications and established development model. Milestones are mostly driven by the software development schedule and implemented development model.

Lastly, the testing processes section describes a number of testing techniques within software engineering and their relevance to the current project. Some limitations of the Extreme Programming development model in connection to testing are also identified.

5.1 Requirements & Specifications

The first step within the development process is extracting the requirements of a desired software product. Unlike in most of the consumer sponsored applications author solely relied on requirements provided by other open source developers which rely on their skill and experience in software engineering to recognize incomplete, ambiguous or contradictory requirements.

Since there were no requirements obtained with the original open source code some of the core requirements have been re-created: by analyzing software behavior; going
through source code to see how data is passed between activities, classes, and methods; and translating variable names, method names and comments from Czech to English.

Following requirements gathering, specification is the task of precisely describing the software to be written, in a mathematically rigorous way. Based on author’s experience, most successful specifications are written to understand and fine-tune applications that were already well-developed, although safety-critical software systems are often carefully specified prior to application development. Specifications are important for external interfaces that must remain stable.

5.1.1 General Description
My chess game contains the initial business requirement of a system that must allow for a multiplayer collaboration among different agents each of them with its own installation of the client software through the web server.

5.1.1.1 Product Perspective (Marketing Context)
The product must form a system that must contain client software installed on user’s hardware and a web server that would assist with collaboration of users, where each user must interact with the web server via the installed client software. The role of a web-server in this software is expanded over a typical role that a web-server would play in other multi-player applications.

5.1.1.2 Product Functions

Client-based software
The client-based software is responsible for basic gaming functions, multiplayer gaming capabilities, and sending requests to the web server, as well as login in to the web server for online game play.

In a multiplayer environment a client-based application doesn’t directly communicate with another player but rather requests a feed from a web-server to display a move played by the opposing side. Furthermore, the application doesn’t keep track of a number of observers directly involved in watching the game. None of the observers ever contact a client for any information but rather deal directly with the web-server to received game updates. Similarly, a web-server also never initiates any communication with the client’s applications. The client’s application must directly request information from the web-server in order to establish any communication pattern with it. This approach insures a lowered processing time and resource allocation for the client-based application.

The Functions listed below are marked according to the following criteria: functions created by the original authors and never modified (marked as O for original); functions created by the original author and subsequently modified to fit the expanded scope of the
Chess board [O] - represented as a two-dimensional array for any internal functions to use and via a graphic interface presented within an XML layout.

Chess pieces [O] – each chess piece is defined by its own rules which are defined by the pattern of possible moves as well as a color.

Chess players [O] – white and black chess players are defined; a player can either be a computer or a human.

Flip board function [O] – can flip the board from default (white at the bottom, black on top) to a reverse representation.

Move function [O] – prompts the computer to make a move without a delay.

New Game function [O] – resets board setting to its original setup.

Undo function [O] – reverses the last move played.

Redo function [O] – repeats the last move played after it was reversed.

Save Game function [M] – saves a game in a readable .pgn format. The modified version of this function allows for a text based copy of the game to be saved for further reuse.


Load Game function [N] – loads a previously saved game.

Login Online [N] – allows a user to login to an online portal.

Register Online [N] – allows a user to register for the online portal.

Logout Online [N] – allows a user to logoff from an online portal.

View Online Game function [N] – allows users to view games from an online library (web server) as well as live games played online.

Join Online Game function [N] – allows a super to join a game that is open and awaiting for another player to join.

Start Online Game function [N] – allows a user to initiate a game online and wait for other players to join.

Settings function [O] – defines game level and language.

About function [M] – provides author credits and description.

Setup board [O] – allows a user to setup a particular position on the board from scratch.

Web Server

A Web Server is responsible for storing game data, processing and formatting game data, processing client’s software requests in a passive response mode, supporting multi-player
live gaming processes, and registering/saving user information. The following services are notable within the web server environment:

- User Database – stores user data.
- Game Database – stores games.
- XML generator – generates XML files.
- XML update/rewrite function – updates/rewrites current XMLs files with new data.
- View Game function – responds to clients request to view games.
- Join Game function – facilitates users and directs request to join.
- Start Game function – registers new games and manages requests.

5.1.1.3 User Characteristics

A User is anyone using client’s software installed on a mobile or stationary platform with reliable internet access.

5.1.1.4 General Constraints

- User must have client’s software installed.
- User must have reliable internet access to play multi-player game.
- Web Server must be online at the time of client’s request.
- Client always makes requests to the web server.
- Web Server never makes requests to the client software but only displays data based on client’s request.
- Web Server outputs .XML data for client server to use.

5.1.1.5 Assumptions and Dependencies

- Client installation and Web Server are interdependent for multiplayer game experience.
- Client software must be installed on user’s device in order to play this chess game.
- User can only view games that are in the web server’s database.
- Every game in the web server database can be accessed via .XML.
- XML documents are generated based on the client’s request.
- User must be registered with the web server in order to utilize its services.
5.1.2 Functional Requirements

5.1.2.1 Overview

This system consists of: 2 subsystems (Internal Play and Online Play); at least 18 different use cases; 5 main actors; and 15 interfaces, both internal and external.

The use case approach is an especially effective technique for deriving software requirements, analysis models, and test cases\textsuperscript{11}. Every use case follows a stated goal that the user needs to accomplish, in other words, a reason someone would use this application. Also various actors that have been identified for this application (internal player, active external player, AI module, passive external user, and external environment) may be involved in each of the use cases. Estimating the anticipated frequency of execution for each use could provide a preliminary indication of concurrent usage loads, and data storage capacities or transaction throughput. Use cases can further be used to derive necessary test cases in the testing phase of application development.

The emphasis on identifying key interfaces and focusing on interface performance has an important component within a modularized application development as a common pattern of interfaces derived from a common pattern of subsystem interactions which could be shared across various modules. This is also true across multiple subsystem interactions as well. Modules within multiple subsystems could re-use common interfaces as well. There are two subsystems within this application: internal play and online place. Each subsystem has a number of unique interfaces as well as a number of commonly shared interfaces\textsuperscript{12}.

5.1.2.2 Use Case Diagram

The following use case diagram describes internal paly and online play sub-systems of this application as well as individual use cases associated with each subsystem and their internal interactions.

Internal play sub-system contains use cases responsible for internal play processes. These use cases include: new game, setup board, undo move, flip board, human-human, redo move, make move, select best move, declare winner, quit game, open game, and save game.

\textsuperscript{11} "Listening to the Customer's Voice" by Karl E. Wiegers published in Software Development, March 1997 http://www.processimpact.com/articles/usecase.html

\textsuperscript{12} "Managing Project Interfaces-Key Points for Project Success" by Peter W. G. Morris http://gspa.grade.nida.ac.th/pdf/PA%20780%20(Pakorn)/18.Managing%20Project%20Interfaces-key%20Points%20for%20Project%20Success.pdf
Online play sub-system contains use cases within the client software responsible for online multi-player game play processes. These use cases include: registration, login/authentication, view game, start game, join game, and play online.

5.1.2.3 Specification for use cases

This section describes in details the development of use case id # 1. Other use cases were developed separately and can be found in Appendix A. Deriving this use case from application requirements required identification of one recognized high level requirement such as starting a new game (New Game). This is an original requirement built into the inherited open source Chess program. Re-creation of this use case is essential in understanding the software as well as in creation of new use cases. There are two actors
involved in this use case: Internal User and AI Module; these two actors can start a new
game and play the game, respectively. The new game feature is one of the options
identified within the application menu and therefore could be directly started from the
application by an internal actor. However, a number of preconditions should be satisfied
before the new game feature could be initiated. In a similar manner, once this feature is
requested, certain post conditions outlined within the use case must be true as well. The
use case below also describes: the normal flow and alternative flow of the process to
request a new game; any exceptions; assumptions; priority; frequency of use; and special
requirements associated with it.

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>New Game</td>
</tr>
<tr>
<td>Created By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Last Updated By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Date Created:</td>
<td>02-09-11</td>
</tr>
<tr>
<td>Date Last Updated:</td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

**Actors:** Internal User of the Client System and AI Module

**Description:** User initiates new game for internal play

**Trigger:** User navigates to the main menu and initiates a new game

**Preconditions:** 1. User must have the software running

**Post conditions:** 1. User can either play against himself, a computer using AI Module,
or another human on the same device
2. New game is started with the board setup in the original condition

**Normal Flow:**
1.1 User turn the software on
1.2 Clicks on Menu
1.3 Select New Game option
1.4 Board is changed to original setup and AI Module is triggered by default

**Alternative Flows:**
1.5 User navigates to the submenu to choose Human-Human game mode for the New Game

**Exceptions:** This is for internal play only

**Includes:**
1. Selecting a menu choice
2. Returning command to re-setup the board
| Priority: | 1. Select New Game  
2. Reshuffle the board  
3. Turn the AI Module On |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency of Use:</td>
<td>Every time internal game is initiated</td>
</tr>
</tbody>
</table>
| Special Requirements: | 1. New Game Interface.  
2. Main Menu Interface.  
3. AI Module |
| Assumptions: | 1. User clicks on New Game to start and internal game. |

5.1.3 External Interface Requirements

Understanding and implementation of external interfaces play is important in software development. Therefore early identification of requirements for these interfaces is an essential part of requirements and specification analysis. This section predominantly describes requirements for GUIs (Graphical User Interfaces) as well as their representation. Also, other types of interfaces such as command line, hardware, and software are identified.

5.1.3.1 Graphical User Interfaces

The following user interface is one of a number of interfaces that is used within the system. Please check Appendix B for a full list of interfaces. This interface combines two smaller interfaces that work together to represent the playing environment in chess. This is an original interface defined with the original open source Chess program used as a basis for this project. The relationship between this interface and associated classes has been derived from the source code provided by the original developer. All of the interactions within this interface are defined via the BoardControl module while basic rules are set within the Task class. A coordinate system is used to identify all of the available moves on the board while a set of two digit numbers is used to represent chess pieces. Control of the chess board is either assigned to a human player via computer mouse, a computer via generated command through an AI Module, or via custom system calls.

There are three cases for taking control of the chess board: a human player uses the mouse to move a piece, the AI module can directly issue a move command, or a human player can issue a move received from the web via custom system call.
1. Chess board and chess pieces

<table>
<thead>
<tr>
<th>Interface Identity</th>
<th>Chess board interface defines the structure of the board while chess pieces interface represent each piece of a chess game. Both used during game play</th>
</tr>
</thead>
</table>

**Resource Provided**

a. Resource syntax  
   - Program Name: BoardControl and Task modules  
   - Parameter Type: Input using mouse  
   - Parameter Info: Coordinates and Type of Chess piece  
   - Exception: None

b. Resource semantics  
   - Any coordinates allocated on the board

c. Resource usage restrictions  
   - Typing and other types of input

Locally defined data types  
   - X and Y coordinates generated by the GUI matching coordinate systems established on the board

**Exceptions**  
   - None.

**Variability provided**  
   - None

**Quality attribute characteristics**  
   - This is limited by the capabilities of the system and the device in terms of processing speed and number of requests processed.

**Elements Requirements**  
   - Board with proper coordinate system, chess pieces, and the mouse like input device.

**Rational & Design Issues**

1. What data is returned under normal conditions?  
   - Board is refreshed and the new position is show to the user
2. What data is returned if the system is down?  
   - Standard system error. Software is closed or restarted.
3. What data should be returned?
Only board related graphic data is returned.

Implementation Notes  None

Usage guide  None

5.1.3.2 Graphical Representation of Key Interfaces

A group of key Graphic User Interfaces displayed below is limited to those that have been added to the application during this project and those that are especially important to the entire application and utilized by a number of original as well as recently added modules.

The chess board interface (including chess pieces) shown in Figure 5.1.3.2.1 is used within all of the gaming options that require game play both internal (between computer and a user) as well as online (play online game or view online game).

Figure 5.1.3.2.1
The menu interface in Figure 5.1.3.2.2 shows a sub-menu portion of the menu. The following options have been added or significantly redesigned within this menu during the development phase of this project: Save Game, Load Game, and Online Games.

![Menu Interface](image)

**Figure 5.1.3.2.2**

The Save Game interface shown in Figure 5.1.3.2.3 allows an internal user to enter a number of parameters used to save an existing game to mobile flash memory. In the original version of this application, the current state of the game was saved along with other parameters without any regards to any previous moves. The current version keeps track and saves the entire transcript of the game up to the current state together with all of the original parameters collected by this interface. Such change enables loading of the saved game upon request from a client-side application as well as allows for an option to submit a saved game to an online archive (or a cloud) which could be developed during further code releases.
The Browser interface in Figure 5.1.3.2.4 is triggered via the Load Game option from a sub-menu. This interface is defined by a third party module integrated within the original Chess application during this project. This module allows a user to browse for a specific type of file type (PGN in this case) within a device’s internal memory and within connected external memory.
The Login Screen interface shown in Figure 5.1.3.2.5 is triggered by the sub-menu’s Online Game option if a user has not accessed the Online Game option before and doesn’t have access credentials saved within the application.
The Registration Screen interface in Figure 5.1.3.2.6 is accessed from the Login Screen interface if a user isn’t registered on the web-server. It is developed to allow a user to submit a user name, password, and an email to the web-server in order to create a unique user record and register for further access to the web-server in order to participate in any of the online games.
The Welcome Screen interface in Figure 5.1.3.2.7 is triggered once a user is logged in to the web-server. This interface was originally introduced within a third party login module but has been entirely redesigned to fit the needs of this application.
The View Game Selection Screen in Figure 5.1.3.2.8 is triggered by selecting View Game within the Welcome Screen interface of the Online Games sub-menu. A list of games available to view on the web-server is displayed. Future development phases of this software could include additional category options as well as search functionality within this interface to accommodate for a large number of games on the web-server. Some of the games displayed are live games being currently played while others are pulled from a game archive.
View Game – The View Screen interface in Figure 5.1.3.2.9 is used once a particular game has been selected for viewing. The client-side application received a full game (in the case of an archived game) or a partial game (in the case of a live game) feed from a web-server and gives user control over displaying game moves one move at a time.

![Figure 5.1.3.2.9](image)

The Join Game interface in Figure 5.1.3.2.10 is similar to the View Game interface where it allows for a selection of games. The games shown by this interface are live games waiting for another party to join. This interface is triggered by the Join Game option from the Online Games sub-menu.
The Start Game Screen interface in Figure 5.1.3.2.11 is triggered from the Welcome Screen accessed through the Online Games sub-menu by selecting the Start Game option. This interface collects appropriate information from the user, and submits this information to a web-server in order to create a record for a new online game request.
The Logout Screen interface in Figure 5.1.3.2.12 is triggered from the Welcome Screen by selection of the Logout button. Once this interface is available the user is no longer logged in to the web-server and application level preferences are cleared of the user’s username and password.
The Settings Screen interface in Figure 5.1.3.2.13 triggered from a sub-menu allows selecting time per move interface. This is an original interface used with this open source chess application.

The Setup Board Screen interface in Figure 5.1.3.2.14 is triggered from a sub-menu and is used to setup a board to a particular state. This is an original interface used within this Chess application.
5.1.3.3 Command Line Interfaces

No command lines interfaces are used.

5.1.3.4 Hardware Interfaces

The client-side application utilizes hardware compatible with Android OS 2.2. The Web Server utilizes hardware interfaces compatible with the LAMP (Linux, Apache, MySQL, and PHP) platform.

5.1.3.5 Communication Interfaces

Communication interfaces are used to establish communication between the client software installation and web server. The following interfaces are used: HTTPS for establishing connection, XML for processing server data, and SAX parse for reading and processing of the XML interface.
5.1.3.6 Software Interfaces

The following software interfaces are used:

- Android OS – to support execution of the Chess application.
- Linux OS – to support web server activities in a LAMP software environment.
- Apache – web server that runs on Linux OS.
- MySQL – database to support the web server in storing game and user information.
- PHP module - scripting language used by the web server to output data for the client’s software.

5.1.4 Performance Requirements

The client’s software performance requirements are based on general Android Operating System hardware requirements. The application is executed on an Android OS platform and doesn’t require any special performance requirements.

The Web Server performance requirements are based on general requirements set by the LAMP environment:

5.1.4.1 Speed

Software: ARM9 @ 200MHz
Web Server: dual 5420 2.5 GHZ

5.1.4.2 RAM

Software: 64 MB or more
Web Server: 3 GB

5.1.4.3 Network Bandwidth

Software: High speed internet, 3G is recommended
Web Server: Unlimited

5.1.4.4 Disk Space

Software: 1 GB of disc space is preferred; at least 256 MB is needed.
Web Server: Unlimited

5.1.5 Design Constraints

5.1.5.1 Standards Compliance
The Software complies with the standard used by Java for the Android development language as well as hardware standards set for applications executed on the Android OS platform. The Web Server complies with standards set by the LAMP environment. All of the communication between the client’s software and web server comply with HTPP and XML protocols.

5.1.5.2 Hardware Limitations
the client’s installation is limited to hardware compatible with the Android OS. The Web Server is limited to hardware compatible with LAMP environment.

5.1.5.3. Platform Limitations
The client’s installation is limited to hardware compatible with the Android OS. The Web Server is limited to hardware compatible with the LAMP environment.

5.1.6 Attributes

5.1.6.1 Security
Security is the capability of a system to reduce the chance of malicious or accidental actions outside of the designed usage affecting the system, and to prevent disclosure or loss of information. A few of the key security issues are addressed below:

- **Spoofing of user identity.** The system uses server side authentication mechanisms to prevent such spoofing. A token is generated per customer login and this token is passed to the client’s software and saved in the cache for future use. The client’s software never saves an actual password to login to the system. The system generates a unique token per login and checks this token against the last token generated. If the match is identified, a customer is allowed to login to the system automatically; otherwise a manual login is required.

- **Minimizing damage caused by malicious input such as SQL injection and cross-site scripting.** The client software never posts or retrieves information to or from the database directly. All of the interactions are done through a web server which acts as a security firewall. The web server generates XML files that are passed to the client’s application and processes client’s requests using PHP filters before sending requests to the database.

- **Data tampering.** Web server access is partitioned into anonymous, identified, and authenticated users; a client side application is used to log and expose behavior that can be monitored. Simple HPPS calls are used as secured transport channels, with all of the data encrypted while being sent across the network.
- **Repudiation of user actions.** This is outside of the scope of the initial development and if requested can be done on the web server without requiring additional software release/update. The server can introduce instrumentation to audit and log all user interaction for application critical operations.

- **Information disclosure and loss of sensitive data.** To prevent access to user sensitive and system sensitive information, all such data resides in separate databases, and access to this data is mediated through a gateway that is associated with a particular login and token. Game specific information is available in a “games” database which is a standalone database that holds less sensitive data.

- **Interruption of service due to Denial of Service (DoS) attacks.** Implementation of code or hardware to detect and mitigate such attacks is outside the scope of this phase of the development. Hardware is outside the control of the client’s software and therefore any mitigation will be limited to code changes. Reducing session timeouts may not be used as one of the preventive methods due to the fact that client’s software is always set to look for new data set by the server for no longer than 30 minutes.

**5.1.6.2 Reliability**

Reliability is the ability of a system to remain operational over time. Reliability is measured as the probability that a system will not fail to perform its intended functions over a specified time interval. A few of the key issues are addressed below:

- **Output is inconsistent.** All of the information sent to as output user interfaces is processed through an output function to make sure that all of the data outputs are consistent. This function checks for any special characters that might have been received from any XML or html feeds.

- **The system fails due to unavailability of external agents such as systems, networks, and databases.** The client side software is solely dependent on the utilization of the internal system for all of the essential activities with the exception of the Online Games. Once a user requests the Online Games option, a check is made to determine whether an active internet connection is in place and if the web server is responding to the request of the client. If either one of the conditions fails, an alert tells the user that the Online Games option is not available at this time. If a network/database/or web server fails during utilization of the Online Games mode, an exception is executed and an exception handling function will alert the user of the network issues encountered by the application. Due to implementation constraints however, multiple failure points are not distinguishable.
5.1.6.3 Maintainability

Maintainability is the ability of the system to undergo preventive maintenance with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the application’s functionality in order to fix errors, or to meet new business requirements. The original design of the application allowed for a great use of modularity. Every functionality and component has been set up as an independent module with a series of interfaces which are communicated through the use of Activities. The goal of this project is to retain this modularity while integrating new and third party components to the original design. A few of the key issues and common inefficiencies are addressed below:

- Excessive dependencies between components and layers. Dependencies are minimized through the use of AllActivity class which is responsible for transfer of information between various interfaces which in turn deal with different modules of the software application. This structure is naturally derived from Android’s implementation of Java.

- The use of direct communication prevents changes to the physical deployment of components and layers. A pluggable architecture approach is used to design for utilization of plugin modules. Any communication between modules is done through the main activity which allows for easier interchange, upgrade, or replacement of inefficient modules.

- Reliance on custom implementations of features such as authentication and authorization prevents reuse and hampers maintenance. Commonly available features are re-used and custom implementations are reserved for domain specific solutions. Widely available common modules are used. The big advantages of such open source code use are that it is: already tested by the development community, relatively bugs free, and often much more efficient than many custom implementation modules. Example of third party modules include: file browsing capability, user authentication and login feature, and SAX XML parser.

- The code base may be large, unmanageable, fragile, or over complex, and refactoring is burdensome due to regression requirements. The original design of this software has a large code base; however, this code base is well divided into modules designed around different functionalities. In order to maintain this modularity and code manageability, any new modules need to be functional additions to the software, and modules must be subdivided into multiples classes in order to minimize code complexity and increase maintainability of the code.
- The existing code may not have an automated regression test suite. This is outside of the scope of this phase of the project. Automated regression tools might be used in the future to test client based applications as well as interactions with the web server.

- Lack of documentation may hinder usage, management, and future upgrades. This document provides the most extensive documentation to this project available to date. More documentation on the original project can be obtained online at the sourceforge.com site dedicated to the original development. Most code changes, as well as all of the new classes introduced contain technical notes explaining how these classes are used and the interworking of the classes. The overall structure of the application is outlined in this document.

5.1.6.4 Performance

Performance is an indication of the responsiveness of a system to execute specific actions in a given time interval. Defining a key success matrix for performance and establishing performance measurements and levels are outside of the scope of this phase of the project development. However, we can still account for factors affecting system performance including any demand for specific actions and the system’s response to demands specifically in relation with the communication channels between the client's side application and web server. Performance optimizing consists of moving major data processing functions to a web server. Actual timing tests have not been performed; however, due to the modularized nature of the system architecture, alternative modules could be derived to reprogram distribution between the web server and client side application for further comparison.

A few of the key performance issues are also addressed below:

- Increased client response time, reduced throughput, and server resource overutilization. The application is designed in such a way to produce the minimum number of calls between the application and web server to get to the desired result. Often one call to the web server is enough for a certain function of the application to be executed. Web Server resources are increasingly used instead of the resource of the client’s application in order to avoid burdening client’s software/hardware infrastructure which is outside of the control of the development team.

- Increased memory consumption, resulting in increased processing time, excessive cache misses (the inability to find the required data in the cache), and increased data store access. Since the entire data storage is outsourced to the web server, increases in capacities can be approached in real time without any changes
to the software application. Memory consumption should stay consistently on the same level due to the fact that online one game option (play, view, etc.) is supported by this application at any one instance.

- **Increased database server processing, resulting in reduced throughput.** This is outside of the scope of the phase of the software development and must be addressed at a later time. Increase database server processing can be handled by adding additional servers and creating layer/virtual server distribution/management schemes.

- **Increased network bandwidth consumption, resulting in delayed response times and increased load for client and server systems.** Increased network bandwidth consumption is an unlikely scenario because of the fact that calls are limited to a specific set of possible data calls/requests and the name of simultaneous games/actions is limited to one. Delays in response times are handled through exceptions. The clients side is always looking for additional information that is posted by the server in regards to a particular game/request; however if the time for this request is over a certain limit then an alert is displayed to the user and alternative gaming options are executed.

### 5.1.6.5 Scalability

Scalability should resolve increased network bandwidth consumption, resulting in delayed response times and increased load for the client and server systems. There are two methods for improving scalability: scaling vertically (scale up), and scaling horizontally (scale out). To scale vertically, you add more resources such as CPU, memory, and disk to a single system. To scale horizontally, you add more machines to a farm that runs the application and shares the load. Neither one of the approaches are viable for the client’s side application. For this reason, most of the load and burden are moved onto the web server which is a controlled environment. A few of the key issues are addressed below:

- **Applications cannot handle increasing load.** Load is limited by the range of interactions with the web server and ability to execute one game option at a time.

- **Users incur delays in response and longer completion times.** Delays in response must be resolved by limiting each wait time and defining specific time parameters. Spikes in traffic and load issues related to the web server are outside of the scope of this phase of the project and must be addressed separately.
- The system cannot queue excess work and process it during periods of reduced load. In order to mitigate the impact of this issue, data grabs are introduced whenever possible. In such a scenario all of the data will be grabbed from the server in one call (whenever such is possible) and all of the data parsing is done internally by the client’s side application. In this case there would be no need to maintain a steady, high speed, network connection at all times.

5.1.6.6 Reusability

Reusability is the probability that a component will be used in other components or scenarios to add new functionality with little or no change. A goal of this project is to maintain reusability principles set within the original application and to introduce reusability principles such as minimization of duplication of components without affecting modularity. A few of the key issues are addressed below:

- The use of different code or components to achieve the same result in different places. Some of the key interfaces and activities have been adopted for use within different modules including custom as well as third party modules.

- Using several systems to implement the same feature or function. The original system defines a set of core functions that could be used by different modules. Those functions include such basic chess functions as making a move and setting up a task. Some of these functions have been re-used within newly added modules in order to increase reusability but still keep as much modularity as possible.

5.1.6.7 Conceptual Integrity.

Conceptual integrity defines the consistency and coherence of the overall design. A coherent system is easier to maintain because you will know what is consistent with the overall design. In this particular case, since it is an open source project and original development has been done by an outside party (developer) it becomes essential to adapt to the style and methodology used by the original developer to keep consistency and coherency of the overall design in order to keep the system maintainable and manageable. A few of the key issues are addressed below:

- Mixing different areas of concern within your design. This issue is relevant to the design of this software. As already mentioned, this application mixes code by its original developer, code developed by third parties and included in this project, and unique code developed for this project. Therefore a number of different programming styles and approaches have been used. An attempt was made to redesign as much of the newly added code as possible (whenever a functional
change to a third party code was needed) to get the new code base adhere to the original code to use the same level of modularity and reuse already developed modules.

- Inconsistent or poorly managed development processes. The extreme programming development process is adopted for this project. The methodology used (release early and often) allows for release of functionality with additional modules every time a new module is added. This is also in line with the original development methodology and in line with the open source development methodology in general.

- Lack of collaboration and communication between different groups involved in the application lifecycle. This doesn’t apply to the project because there is one developer working on the project.

5.1.6.8 Manageability

Manageability defines how easy it is for system administrators to manage the application, usually through sufficient and useful instrumentation (documentation and error logging features) exposed for use in monitoring systems and for debugging and performance tuning. Modularization should help with manageability of a large code base, such as in this project. A few of the key issues are addressed below:

- Lack of health monitoring, tracing, and diagnostic information. In general, implementation of various monitoring tools is outside the scope of this project.

- Lack of runtime configurability. This is outside of the scope of this phase of the project. Infrastructure of the hardware on which the client’s application runs is outside of the control of the developer.

- Lack of troubleshooting tools. Instead of troubleshooting code/tools which are outside of the scope of this phase of the project, exception handling and notification are introduced. Trouble shooting code/tools are only to be used during development in order to troubleshoot specific errors/issues.

5.1.6.9 Interoperability

Interoperability is the ability of a system or different systems to operate successfully by communicating and exchanging information with other external systems written and run by external parties. This system in particular is designed towards most efficient interoperability possible due to the fact that the client’s software installation and web server present the same application solution. Communication protocols, interfaces, and data formats are the key considerations
for interoperability and some are standardized and optimized for efficiency. A few of the key issues are addressed below:

- **Interaction with external or legacy systems that use different data formats.** This is resolved due to the fact that the only data format that is used is XML.

- **Boundary blurring**, this allows artifacts from one system to defuse into another. Systems are being strictly isolated by physical isolation (one is on the client’s installation and the other one is a publicly accessible web server which is not running on client’s machine), as well as isolation via control in such a way that client has no control of the web server and the web server has not direct access or control of the client’s installation.

- **Lack of adherence to standards.** Standards established by the original developer were used as well as standard communication protocols and interfaces to work with a web server. This should help further maintainability as well.

5.1.6.10 Testability & Test Strategy

Testability is a degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.\(^{13}\)\(^{14}\)

Major attributes of testability include operability, observability, controllability, understandability, and automatability.\(^{15}\) It is outside of the scope of this research to address these attributes in detail.

A few of the key practical issues regarding test strategy are addressed below:

- **Complex applications with many processing permutations are not tested consistently.** Modular design of the application solves complexity issues and allows for isolated testing of individual modules through the use of interfaces.

- **Lack of test planning.** Testing initiated during development process. Using extreme development process allows early testing of individual code components every release and testing of separate modules and how each module is

---


incorporated within the overall system. Testing with extreme development process is done iteratively and continuously.

- **Poor test coverage, for both manual and automated tests.** Automated tests are outside of the scope of this phase of the project. Manual tests are done after each functionality release (iteration) using a standard test script testing overall existing functionality as well as a specific test script testing new functionality developed within a specific release.

- **Input and output inconsistencies.** All of the inputs are in the form of mouse generated commands or XML formatted data. Outputs are either in a form of graphical representation or textual data provided to the web-server. All of the information sent to the client system is parsed by web-server for data inconsistency and special character issues. Similarly, all of the information received from the client-side application is treated in the similar fashion. Client-side application performs only limited data cleanup to avoid additional resource usage.

### 5.2 Development Model

The Extreme Programming (XP) development methodology has been adopted for this project. It is a type of agile software development. The intention behind utilization of this concept is to improve software quality and responsiveness to changing requirements, given limited resources available for this project. Software releases are concentrated around each milestone/functionality referred to as a module which allows for quick quality assurance testing as well as evaluation of scope of the project after each module is completed. Each software release requires development of interfaces some of which may not be used in the final version of the application.

Some of the elements of the original XP methodology such as pair programming, close collaboration with customers including ongoing review sessions, and flat management structure, are not applicable to this project.

On the other hand, such methods as unit testing and avoiding programming of features until they are actually needed have been used to evaluate every additional module as well as its impact on other modules and the adherence to project schedules.

### 5.3 Development Milestones

- **Compile and Run Original Open Source Code.** This is an essential milestone; defining the right version of the API and changing the existing codebase to
conform to this version will allow for better integration with third party modules and code snippets used within this software, and an array of mobile devices/architectures.

- **Recreate save functionality.** Rework of the save functionality previously defined within the software is required in order to implement proper load game functionality based on the correct gaming notation.

- **Develop load functionality.** This functionality consists of a number of modules that must be integrated within the system. A third party file browser mechanism needs to be selected and integrated within the existing structure, a file loader element needs to be developed, and the file parser needs to be set with specific criteria unique to chess notation. In addition, board control elements must be called to recreate the loaded game on the chess board.

- **Develop ONLINE mode.** The online mode will contain a number of interfaces defining online functionality such as: Login to online account, Register Online, View Game, Join Game, Start Game and protocols and communication channels used to access information stored on the Web Server.

- **HTTP/XML Protocols.** Method uses HTML and XML protocols to communicate with the Web Server. Without this milestone further work on the online functionality is impossible.

- **Password authentication.** Password Authentication functionality involves communicating with the Web Server using previously developed secured communication protocols.

- **Login/Logout functionality.** Allows for user session initiation and modification of user preferences in order to store user credentials on the client’s machine for further automated usage. Logout functionality would reset preferences and erase sensitive user data.

- **Online account registration.** Processes user inputs and posts information to the Web Server. Retrieves web server response and interprets it as either a registration success or a failure with the reason.

- **SAX parser and parsing XML documents.** SAX parser is introduced via a third party solution in order to process XML formatted data from the web server.

- **View Online Game functionality.** This module is composed of a series of functions that allow the user to select an online game from a set of available games to view, retrieve appropriate data from the web server, and display data on the board with a user controlled frequency.

- **Join Online Game functionality.** This module is composed of a series of functions that allow the user to select an online game from a set of available games to join, notifies the web server and the opposing side that the game is joined, displays data on the board with online specific user data, and transitions into an online game playing mode.
- **Start Online Game functionality.** This module is composed of a series of functions that allow the user to create an online game by completing a form with preselected criteria, waiting anyone else online to join created game, displaying data on the board with online specific user data, and transitioning into an online game playing mode.

- **Online game playing functionality.** This module establishes the communication pattern and frequency between the client side software and web server in order to exchange moves during the online game play. Information received from a web server is replicated on the board; moves created by the user are displayed on the board and send to the web server. In all instances the client’s software is actively engaged in retrieving information from the web server. This is the most important module within the project. Creating this module last allowed for more time in understanding and identifying requirements as well as system limitations and availability of multiple alternative solutions.

**5.4 Testing Processes**

The main focus of this section is to outline major testing procedures performed during the development phase of every module of the software.

5.4.1 Conversion Testing

Testing to ensure that all data elements and historical data are converted from an old system format to the new system format, specifically ensuring the transition of the Android OS 1.5 based original open source software to a newer Android OS 2.2 platform. This is applicable during the initial integration development phase as well as during testing of integrated third party components used within the newly added features of the software.

5.4.2 Job Stream Testing

Requires communication with the server during testing of online modules to test production related issues. Every online module needs to undergo this testing.

5.4.3 Interface Testing

This type of testing is done to ensure that a specific module operates efficiently and effectively outside the application boundary with all interfaces specific to this module. A similar approach applies to the application as a whole and all of its interfaces.
5.4.4 Security Testing

Security Testing is done to ensure that the control and auditability features of the application are functional. This is especially important during testing of communication protocols and the SAX Parser module. The authentication module is also subject to extensive security testing.

5.4.5 Recovery Testing

Testing done to ensure that application restart, backup and recovery facilities operate as designed. After a failure the system is designed to restore the last position on the board and preserve online login preferences while online connection to a specific game/opponent is expected to be terminated permanently.

5.4.6 Performance Testing

Testing done to ensure that that the application and every module within this application perform to industry standard expectations in terms of response time, availability, portability, and scalability. Such standards were defined earlier in the documentation.

5.4.7 Regression Testing

Testing done to ensure that applied changes to the modules and shared classes have not adversely affected previously tested functionality. Regression testing is performed after introduction of each new module, using permanent as well as temporary interfaces.

5.4.8 Integration Testing

Testing conducted in which software elements, hardware elements, or both are combined and tested until the entire system has been integrated. The purpose of integration testing is to ensure that design objectives are met and ensures that the software, as a complete entity, complies with operational requirements.

5.4.9 Acceptance Testing

Testing conducted to formally determine whether or not a system or a particular module satisfies the acceptance criteria set by the requirements listed above. Acceptance testing ensures that contractual requirements and key objectives are met and that all components are correctly integrated in a comprehensive software package.
Chapter 6: IMPLEMENTATION

This section covers high level implementation changes and outlines new classes added to the original software. Further in this section, a state diagram is presented describing the communication patterns of various modules and outlining the activity classes used by a number of modules of various types (original, newly developed, and third party integrated). Detailed descriptions of key classes and methods are provided subsequently. Three subsets of key classes and methods are identified as modified, third party, and new.

6.1 High Level Summary

The following high level changes were made to the original software.

- LOAD GAME - loads previously saved game
- SAVE GAME – saves any game in a uniform readable format
- VIEW GAME ONLINE – allows a registered user to view any past games in the online database and to follow live games.
- JOIN ONLINE GAME – this function allows a registered user to join an online game
- START ONLINE GAME – allows a registered user to start a game
- ONLINE GAMING MODE – allows for two devices to collaborate through the use of a web server
- ONLINE REGISTRATION – allows a device holder to open an online account to participate in all of the online activities.
- ACCOUNT AUTHENTICATION/LOGIN – allows for login/logout functionality

6.2 State Diagram

A state diagram shows the various “states” (reactions) that this application can be in, as well as what action or input (from the user) is necessary to get to a specific state.
6.3 Description of key classes and methods:

In general there are three subsets of key classes and methods described below: (1) classes/methods that were created by the original author and subsequently modified to fit the expanded scope of the project (marked as M for modified); (2) classes/methods created by a third party and subsequently modified to be applicable to this particular project (marked as T for third party); and (3) class/methods created from scratch (marked as N for new).

AktivitaSachovnice.Java (M) (English translation- Chessboard Activity)

This activity is at the core of the entire project. This is where the main menu is set and it also acts as a gateway that helps other activities to communicate with each other. This allows us to achieve the high degree of modularity needed for a project of this scale.

Menu changes:

I've added two new root menu items "load game" and "online games". Also, the save game has been redesigned.

New methods:

- `getInputStreamFromUrl [T]`- takes simple http inputs and processes those through the system.
- `downloadUrl [T]`- downloads content of the URL
- `onActivityResult [M]`- this method is modified to facilitate data transfer:
  1. From `AndroidFileBrowser` class to `OpenFile`
  2. From `OpenFile` class to `loadgame()` method in `BoardControl` class
3. From Welcome class to either ViewGame, JoinGame, or StartGame based on the status preference

BoardControl.Java [M]

This class controls all of the essential functionalities on the chess board. It extends the View and performs any kind of board related changes.

I've added the following methods:

- loadgame() [N] - this function receives game data as its input. It parses the data, removes anything that is not moves related. It associates each field with an array of numbers and then places one of the values of this array either in the fromMove or toMove array. Once that is done, a loop traverses both arrays and calls “makeMove” function.

- viewgame() [N] - this method changes the XML layout of the board by adding a button (play) and game specific information. Every move is parsed and loaded into one of two arrays in the same manner as done for the loadgame() method. The user controls the frequency of the moves displaying on the page by pressing the "play" button. Once all of the moves are played and the game itself has finished play, a button will display a game over message. If it is an ongoing game the system will access the online portal every 5 seconds using [gameid].XML and look for the next move. If new moves are available, the button message will change and the user will be able to click “play” to call a new move.

- onlinegame() [N] - both the JoinGame and StartGame class use this method for a multiplayer online game experience.

Task.Java [M]

This class is responsible for performing several base tasks such as converting the numeric representations of the fields into characters and back; the original save methods is also called from this class. The original class was significantly changed to accommodate the new save method in order make the format of the saved PGN file usable during the load functionality. Since it is only possible to load games that we previously saved, this change became essential for the overall project. The original save method only saved the end result/state of the game without recording any sequence of moves. In the new interpretation of the save method, the entire record of the game is saved move-by-move which makes it easier to replicate during the load procedure.

OpenFile.Java [N]
This class receives a file name as its input, and then retrieves a particular file with
the file name form /sdcard/ folder. It saves all of the content of the file into String
text and passes this string back to AktivitaSachovnice.Java for further processing.

AndroidFileBrowser.Java [T]

This class has been significantly redesigned from its original version (LOOK
HERE). The purpose of this class is to provide file browsing capabilities to
support the Load game functionality. A specific .XML layout has been introduced
to support this class. The current file directory has been setup to /sdcard/ and only
files from that directory are set to display initially. Selection of a particular file
name through a selectionRowID variable has proved to be a challenge due to a
versioning difference in Android OS builds. A different iterative solution has been
introduced to resolve this issue.

MyXMLHandler.Java [T]

This class parses the contents of .XML files using the SAX Parser.

SitesList.Java [T]

Contains getter and setter methods for variables used in XML parsing.

JoinGame.Java [N]

This class allows a logged in user (requestor) to join an existing game as a player.
It retrieves all of the games awaiting players at
http://www.gabbleboard.com/chess/games/joingame.XML. The following
information is retrieved: session (game id) username of the player (game owner)
and color which is available. Information is displayed to the user through a UI
interface. Requestor picks one of the available games. System sends a request to
id]&playername=[username]&status=request. In response the Web Server will
generate 'waiting' which means the name of the requestor is added to the request
to join a queue, or 'denied' which means someone has already been selected.
Following this, the requestor will go to the same URL every 5 seconds to check
status. If the system returns an 'approved' message, then the game starts; if
'denied' then another game to be picked. Game information is then passed to
AktivitaSachovnice.Java for further processing. Following an overall architecture
decision, the Web Server never initiates transfer of any information to the client-
based application and only provides information based on a request from a client.

ViewGame.Java [N]
This class accesses http://www.gabbleboard.com/chess/games/viewgame.XML and parses it to retrieve the following information per game available to viewing: session (unique game id), status (completed, in progress), white (name), black (name), result (ex: white won), moves (# of moves played), tournament (name of the tournament), commentaries (yes/no), date played (if live then now). This information is presented to a registered user through a GUI window, user selects a particular game to watch and then approves it/confirms through a alert popup. Once game is selected, all of the collected information about this game is passed to AktivitaSachovnice.Java for further processing.

LoginActivity.Java [T]

This activity is responsible for allowing user to login into the online portal. Activity has been slightly modified from its original version (http://androidsamples.blogspot.com/2009/06/how-to-use-http-connection-saxparser.html) to accommodate for a simpler login process. This activity also supports establishment of a global login status which is saved even after user exits the program and unless temporary files are cleaned through the system. Once username and password is entered by a user through the GUI, a call is made to the online portal (http://www.gabbleboard.com/chess/users/index.php?username=[username]&password=[password]). If user is identified as a registered user by the web server "yes" is returned, otherwise "no" is returned. If user is identified, LoginHandler.Java function is activated. If user is not identified LoginError.Java is activated.

If user has been identified as a registered user, user will be able to avoid the login screen every time program is used unless user logs out of the system or preference cache files are cleared. User will be automatically redirected to Welcome.Java

LoginError.Java [T]

This class displays an error message for unidentified users.

LoginHandler.Java [T]

This class parses login data and updates a status parameter for the saved login entry once the user has been logged in.

Welcome.Java [N]

This class has been significantly redesigned from its original version. The XML layout has been redesigned as well. Instead of displaying a simple welcome message, it now presents 4 buttons: Start Game, Join Game, View Game, and
Logout. The StartGame button calls StartGame.Java, while Join Game and View Game call JoinGame.Java and ViewGame.Java, respectively. The logout button clears values of global variables for username and password and deletes those from the global preferences queue resetting the login status of the user to guest.
Chapter 7: LESSONS LEARNED

This section provides a detailed analysis of the work performed within this project, outlines issues encountered within implementation of the prototype, and describes the key findings and solutions achieved.

The beginning of this section summarizes issues encountered within the project and separates them into the following areas: scope management, setup, and development. Next, management and quality processes are analyzed and broken down into the following topics:

Unimplemented requirements - provides a number of key requirements left out of the final implementation and describes issues associated with them;

Lack of available resources - resources outlines a number of lacking pre-requisites and their impact on the project; and successful resolutions – describes unique solutions introduced to the system.

Next, a number of unexpected events and technical issues are described. Finally, the Top 10 Project Issues are outlined and associated Quality Assurance (QA) and Testing issues are described.

7.1 Summary of Issues

Scope Management Issues

Defining the correct project scope has been one of the most important challenges of this project from the very beginning. The high level definition of the scope was originally to create a cross platform software application with two components: client based installation and a web server. The real challenge was to separate “must have” functional requirements from “nice to have” optional requirements, especially taking into consideration available human resources and technological solutions.

It was decided to move cross-platform automation into a “nice to have” category while developing all of the other requirements (such as picking the development platform) with an eye towards cross platform automation. Since existing technologies behind cross platform automation and automated language translation are not well developed it is risky to base success of the project on this factor knowing that this may be unattainable.

Software development has undergone significant scope change as well. Originally the idea was to create a software package from scratch based on some existing models and then to introduce the web server and multigame functionality in pursuit of creation of the
two component software package. After realizing that the goal of this project is not to create a particular software or a game, but rather to show that a two component approach allows for better resource management, performance, and possibly easier cross platform automation, it was decided to take as a starting point an existing open source software application that doesn’t have any online component and introduce such a component together with multiuse or multiplayer functionality.

Web server implementation underwent certain scope changes as well. Originally a fully functional web portal with front end interface was to be developed. However, based on the fact that the major use of the web server would be the client side application rather than a user directly accessing the site, it was determined that the web server should only respond to requests sent by the server and be closed to direct user access at this time.

Setup Issues

Get the open source code. It was anticipated that dealing with an open source project, especially with an open source project that had been abandoned, would bring some communication, process, and knowledge transfer challenges. However, as it turned out, dealing with an open source project that has had only one original developer on board proved to be even more challenging. Even though the project was available on SourceForge, there was no publicly accessible way to download working source code. Since the project had been abandoned for over a year already, it was difficult to establish a communication channel with its original creator, and getting the latest version of the source code required special requests and unintended delays.

Create the environment. Setting up an Android Testing Environment, including the Android SDK, Eclipse development environment, Android emulator, and Android plugin for Eclipse, proved to be a challenging process by itself. Resource availability proved to be another issue. While the minimum hardware requirements for an Android emulator to run are a P4 3.0 GHz and 512 MB of RAM, a far more advanced machine might also have problems running the emulator. Firstly, there is an issue with the ADT plugin and Helios (version name for Eclipse 3.6) which cause a lag with looking up Android classes - so Galileo (Eclipse v3.5) was used instead. Secondly, the emulators become more resource hungry depending on the version of Android OS used by the software.

Source Code Compatibility. While compiling the original source code I ran into a compiling error “default.properties file not found error”. At that time all I wanted to do is to take the source code and run it in the emulator. I was simply trying to compile the existing source code as-is and run it in the emulator. It turned out that the original version of the software was written for the Android 1.5 OS, which still depended on the standard Java.util package from Sun. However, in subsequent versions of Android (2.2)
this utility package has been replaced with an Android-specific library that required me to change some of the import statements in order to compile the software.

Development Issues

Understanding the software. It is always a challenge to understand the architecture of an existing software package even with all of the appropriate documentation available. However it becomes even more challenging when all of that body of knowledge is gone and all that one has available is the pure source code and a limited number of comments written in a foreign language. The original software code contained two packages: HonsozySachi2010 – this contains all of the activities and modules and MyseleniAPravidla – this contains all of the classes responsible for AI code as well as general games rules, board definition, and chess pieces definition. Understanding the relationship between those two packages and the interworking of classes and modules within each package was the key towards understanding the architecture of the software itself.

Human language issues. Even though the application’s front end GUI is created in English, all of the comments left by the original developer are in Czech; the same applies to all of the variable, method, and class names. In order to understand the inner workings of different classes and methods I had to translate almost everything to English and trace definitions of all methods to make sure the method names had been changed throughout the entire source code. My knowledge of Russian helped me to translate everything much quicker.

Following the original coding styles. Every developer has his/her own style of coding which is also in line with other software goals, especially for single developer projects. Following the single development style preserves the original functionality, helps software manageability and maintenance, and assists in further development, quality assurance, and testing. Code encapsulation and modularization have been introduced by the original developer and should be followed in order to support further modularized functionality. Also, the level of development knowledge processed by the original developer, based on the earlier analysis of the source code, seemed to be superior to the skill level of the author and therefore required additional training and changes in programming habits.

Compatibility with third party functions. One of the key development tools that have been used during this project is the ability to introduce third party solutions for specific functionalities (modules) and integrate those solutions within the current body of code and architecture while modifying third party code to fulfill specific goals of this project. Aside from the usual cosmetic and functional programming issues expected with this kind of integration, the author ran in to code and version compatibility issues. The
Android platform is a new and dynamic operating system with major changes added to every release. However, the Android’s adaptation of Java also changes with every Android OS modification, with many original packages supported by the standard Java implementation changed to Android Java specific solutions which internally are supposed to make the implementation mobile-friendly, while saving hardware and processing resources. This evolution of Android Java often makes some code written for a previous version of Android OS not readily compatible with a later version. The same issue is also relevant with re-use of third party programming solutions (modules/classes/or functions) within larger software applications.

7.2 Management and Quality Processes

Discussions of Management and Quality Processes within this project include identification and analysis of unimplemented requirements, analysis of unavailable resources and its impact on the project, as well as identification of successful resolutions. All of the three factors are discusses as related to the management and quality processes.

7.2.1 Unimplemented Requirements

A number of requirements unidentified at the beginning of this project have been left unimplemented. However, in all cases outlined below, signification research and effort have been spent on promoting these requirements which resulted in project scheduling delays as well as re-evaluation of the goals and milestones of the project.

7.2.1.1 Cross Platform Automation

What exactly went wrong. Considerable time was spent researching different methods of cross platform automation, especially porting from a similar Android based Java Platform to a conventional Java platform in order to run client application on PCs, as well as further automation from Android’s Java to Objective C supported by Apple. However, there was no success in producing any working prototype. Nevertheless, in my research I found companies such as Titanium and PhoneGap that achieved considerable success in cross platform code translation. The challenge with both of these approaches is that they start with a simple set of web based instructions coded in JavaScript, HTML, and CSS and port those to different platforms creating all of the interfaces and unique code for each platform from the original web based instructions. However I feel that these web based instructions wouldn’t be able to provide enough support for more complex

16 “Showcase of Titanium Platform” Appcelerator @ 2011 http://www.appcelerator.com/showcase/
17 “How PhoneGap Works” PhoneGap @ 2011 http://www.phonegap.com/about
applications such as the one described here. Specifically, I believe that the complex functionality of a chess program, such as strict chess rules and artificial intelligence used by chess application cannot be simulated via simple web instructions. On the other hand, Java applets implemented by both Titanium and PhoneGap Solutions are not portable.

Why did it go wrong (analysis). There are at least two factors that contributed to the lack of success in cross platform automation for this prototype:

1. Lack of understanding of the process of automation and miscalculating the scope of work. The original approach was to create a software prototype in Android-based Java which would be similar to conventional Java and also be a mobile language accounting for specific architecture and memory management issues unique to mobile environments. However, due to a unique set of Android libraries and activities, I wasn’t able to automate porting to conventional Java. Similarly, having a virtual Dalvik machine as an integral part of Android’s Java implementation disrupts any similarity to iOS and Objective C development.

2. Possible choice of wrong platform. Conventional Java should have been chosen as the base platform for development of the prototype. An expanded knowledge base is already available for conventional Java PC development, and automation techniques established to port such software to Android and iOS would have made it easier to automate the port to multiple mobile platforms.

Is it worth fixing? Fixing issues outlined in the previous paragraphs requires an overhaul of the entire prototype and a complete rework of the code base to adhere to conventional Java rules and syntax. Furthermore, some of the activity-based programming impossible in conventional JAVA would have required structural redesign of the software, which would have led to recreation of some of the classes from scratch. Such changes are outside of the scope of this project and would affect deliverability of the software.

How do we fix it? Alternatively, instead of extensive redesign of the prototype, additional research into automation and porting as well as requests for collaboration from various community members can bring a solution to this problem in the long term. A phase two of this project can be proposed to specifically address cross platform automation.

7.2.1.2 Adherence to Original Development Schedule/Milestones

What exactly went wrong. The original development timeline for this project was set from February of 2010 to August of 2010 with all of the major quality assurance work and testing finished by September of 2010. However, all of the original LOE (level of effort) estimations proved to be wrong or inaccurate largely due to major development setbacks throughout the process.
Why did it go wrong (analysis). I’ve identified three major factors that contributed to substantial schedule delays. These factors are:

1. Lack of appropriate development experience, especially for a project of this scale using an unfamiliar language and platform, contributed to the wrong LOE estimation on all of the major milestones throughout the project.
2. Lack of proper architectural design was another contributing factor. Following the same model of extreme development as the original developer, I tried to release new features as often as possible which caused me to create temporary graphic interfaces for testing purposes and also contributed to some of the delays.
3. Lack of proper project documentation and a log of existing errors caused me to spend additional time on understanding functions that looked trivial at the beginning of the project, which also contributed to delays.

Is it worth fixing? Each of the three issues described above required individual analysis to determine appropriate resolution. However, it is also clear that these issues are fundamental and their complete resolution is outside of the scope of this phase of the project. For instance, gaining enough programming experience to be able to set realistic LOEs requires time, participation in many projects of various scales and sophistication, and many coding hours. Similarly, in order to understand the original architectural design of this project or to develop the required documentation artifacts for this application, many hours and a complete understanding of the software are required.

How do we fix it? How do we stop it from happening again? I introduced a number of short term solutions which helped me resolve immediate issues at hand and move on to other tasks associated with the project. Those solutions were:

1. In order to account for the gap in experience, I integrated many third party solutions available on-line to the public. These solutions are tested, readily available, and efficient. One obstacle in getting such solutions integrated was the fact that none of them did the exact things that were required by the project, so I had to modify the third party code to make it work according to the project’s requirements.
2. Instead of determining the exact architectural structure, I assumed that a certain modularized structure had been in place based on the analysis of the classes and modularization of functionality represented by the naming convention used by the original code developer. Once that was settled, I was able to use the same architectural design with all of the additional functionalities created by me.
3. Similarly, instead of introducing proper documentation, I added detailed comments describing certain classes of the original code base that I was able to understand. I also extended the same comments structure to all of the new classes that I introduced to the project. This approach allowed me to use a temporary
documentation technique to get around the code without spending additional resources on proper documentation setup.

7.2.2 Lack of Available Resources

7.2.2.1 Lack of Documentation and Log of Errors

*What exactly was lacking?* A major obstacle getting this project off the ground and developing quality code was the lack of proper project documentation including any kind of error log. From the very start I had no idea what errors/unresolved issues were still left in the code by the original developer, and any errors encountered by me within the code consequently were assumed to be errors within my code. I was reluctant to change any of the existing code for fear of introducing dependencies leading to more errors.

*Why is it lacking?* I assumed that the original developer was working on this project alone, using some style of Extreme programming development and was concentrating on releasing as many versions as possible while adding new features to each version. Due to lack of resources and skills associated with proper documentation and error logging, the original developer never spent the required time putting any of this information together.

*Justification for including an Error Log.* As a project gets larger and more functionalities are added it becomes essential to get proper documentation in place in order to promote further development, maintenance, quality assurance work, and use of the software. Especially when this remains in the open source realm, it would be practically impossible for any other developer to continue working on this without a sufficient level of documentation.

*How do we put it in place?* Introducing proper documentations requires a detailed understanding of the architecture of the software, interworking of different classes and methods, and reasoning behind specific programming solutions. At this time I still do not understand how every component of this software works. Certain areas, such as the AI module and some of its rules, are still treated like a black box. I can work with certain interfaces and get the input that I require, but how it really works is still a mystery without background design and implementation documents. Implementing reverse engineering techniques allowed me to move forward without original documentation however recreating this documentation fully is outside of the scope of this phase of the project and will be a standalone development project of its own.

7.2.3 Successful Resolutions
At the very start of the project I successfully ran the original application using a more advanced version of the Android OS. I updated the original code by replacing the standard Java packages with the Android-specific versions. Once the packages were replaced, I had to modify the code due to method signature differences between the two versions of the API. It is also important to highlight that a concept of Swing widely used in the conventional Java is omitted within Android’s implementation possibly due to differences between Java VM and Dalvik. Instead an adopted version of Swing is used with some new concepts such as Activity and Intent added. Below is a summary of some of the classes used by this software, along with their package prefixes:

- `android.app.Activity;` - The Activity class is perhaps the most important class for an Android application and differentiates the Android version of Java from conventional Java in a dramatic way. By definition, an Activity is any one thing that a user can do – and hence the main class in the application will be sub classed from this.
- `android.app.AlertDialog;` - This class allows a developer to fire special dialog popups called Alerts. AlertDialog is a subclass of Dialog that can display one, two or three buttons. If you only want to display a String in this dialog box, use the `setMessage()` method. If you want to display a more complex view, look up the FrameLayout called "custom" and add your view to it.
- `android.app.Dialog;` - This class provides a facility to manage the creation, saving and restoring of dialogs. Often you will want to have a Dialog display on top of the current input method, because there is no reason for it to accept text.
- `android.content.DialogInterface;` - Defines the interface of a Dialog box, with such features as Buttons, and Input boxes defined.
- `android.content.Intent;` - An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.
- `android.content.SharedPreferences;` - Interface for accessing and modifying preference data.
- `android.net.Uri;` - This class allows the system to efficiently parse the URL passed by the software. In the interest of performance, this class performs little to no validation. Behavior is undefined for invalid input. This class is forgiving--in the face of invalid input; it will return garbage rather than throw an exception unless otherwise specified.
- `android.os.Bundle;` - Bundle is a utility class that lets you store a set of name-value pairs. Bundles are used heavily by Intents to move data between Activities. You will always find this import along with the import for Activity class because
both the onCreate() and onFreeze() methods take Bundle as a parameter. Into a Bundle object, you can put integers, longs, strings, arrays, etc. along with the keys to identify them. When needed, these values can be obtained by using those keys.

- **android.textEditable;** - This is the interface for text whose content and markup can be changed (as opposed to immutable text like Strings).
- **android.util.Log;** - Helps to output some log messages. Generally, uses the Log.v() Log.d() Log.i() Log.w() and Log.e() methods. The order in terms of verbosity, from least to most is ERROR, WARN, INFO, DEBUG, VERBOSE.
- **android.view.LayoutInflater;** - This class is used to instantiate an XML layout file into its corresponding View objects. It is never be used directly -- use getLayoutInflater() or getSystemService(String) to retrieve a standard LayoutInflater instance that is already hooked up to the current context and correctly configured for the device you are running on.
- **android.view.Menu;** - Interface for managing the items in a menu.
- **android.view.MenuItem;** - Interface for direct access to a previously created menu item.
- **android.view.View;** - Represents a basic class that is responsible for drawing on screen elements and event handling.
- **android.view.Window;** - The Window class contains constants for setting up the look of the window (for example, FEATURE_NO_TITLE takes out the title of the window and FEATURE_LEFT_ICON puts an icon on the left side of the title bar). The Window class also contains a whole bunch of getters and setters for various window attributes like setTitle, setStyle, getStyle, and getcontext.

- **android.widget.Button;** - Represents a push-button widget. Push-buttons can be pressed, or clicked, by the user to perform an action.
- **android.widget.CheckBox;** - A checkbox is a specific type of two-state buttons that can be either checked or unchecked.
- **android.widget.EditText;** - EditText is a thin veneer over TextView that configures itself to be editable.
- **android.widget.TextView;** - At its basic representation, TextView is a label (meaning, it just displays text that is not editable). As a label (uneditable TextView), it can be used as captions and textual displays.
- **android.widget.Toast;** - A toast is a view containing a quick little message for the user, like a tool tip. The toast class helps you create and show those. When the view is shown to the user, it appears as a floating view over the application. It will never receive focus. The user will probably be in the middle of typing something else. The idea is to be as unobtrusive as possible, while still showing the user the information you want them to see. Two examples are the volume control, and the brief message saying that your settings have been saved.
• android.content.DialogInterface.OnClickListener; - Interface used to allow the creator of a dialog to run some code when an item on the dialog is clicked.

Another major success was the fact that I was able to define a simple and reliable method of communicating with a web server and a way of processing data by the Web Server that didn’t require in depth web development at this phase of the project. The Web Server is required to produce either an XML document on demand by the client’s application or provide a simple “yes” or “no” response in plain text format based on a set of specific client requests. A client-side application would never post anything to a web server database, has no knowledge of any internal web server processes or architecture, and doesn’t hold any connection-specific information aside from a set of URLs that define web server services. The Web Server is required to create compatible .XML interfaces for every type of online activity supported by the client application; it will never send any active calls to any client application but instead only displays information based on a url parameter request or direct access to XML files.

Additionally, I created a milestones plan connected to functional releases and was able to follow it. Specifically, I released the following milestones with working prototypes in sequential order:

• Load Function Release
• Save Function Release
• Registration and Authentication Release
• View Game Release
• Join Game Release
• Start Game Release

For every functional release I was able to create temporary and permanent interfaces that allowed me to do unit and integration testing for the application, in general, as well as the newly created functionality, specifically, and to evaluate any dependencies in regards to functionality as well as usability of the software. Another advantage of such an approach is that for any given release I had a working program that could be presented to anyone upon request. This helped me evaluate my progress and adjust scope against the timeline and schedule associated with the project.

7.3 Unexpected Events

This section covers events that had a direct and tangible impact on the project plan, budget and timeline resulting in a deviation from the initial agreed upon plan. The events are as varied as they are random.
7.3.1 Abnormal Proxy Issues with Android Emulator.

In light of heavy hardware requirements set by the Android SDK and Eclipse Development Environment, at the time of the start of the project I was only able to run both on a single machine available to me. This machine was used by me in connection with my employment and was behind a proxy firewall with the ability to VPN to my employer’s network during remote working sessions or telecommutes. Once set up and able to run and compile my application, I quickly discovered an abnormal behavior within the Android Emulator. Specifically, I was not able to connect to the network from a function responsible for sending HTTP requests within my software or the emulator’s Internet browser behind the proxy of the working computer. However, as soon as I transferred to the company’s VPN this connectivity problem seemed to resolve itself. After some research on this issue it became evident to me that this is a widespread Android Emulator error documented in many support forums and discussion groups18. Below are some key points in regards to this abnormality:

How did it impact this project? Since I wasn’t aware of this issue initially, it took additional development hours to identify the root cause of the error. My initial assumptions were more in line with a general code error on my part or an incorrect Android Emulator installation. Only by luck, while working on a job related assignment remotely via VPN I was able to successfully run online modules of my software and identify the abnormal behavior. The immediate impact on the project was that it set back the project’s development schedule by approximately three working days.

Why wasn’t it considered as a risk and mitigated? This was an unknown that I never accounted for during the initial design phase of the project. This error is only specific to certain versions of the Android Emulator, such as version 2.2, that I am running for this project.

How can it be avoided next time? There is no immediate fix or resolution that is available as an update to the Android Emulator. However, I was able to use a manual fix to adjust wireless network settings within the Emulator which resolved the issue. I followed the following series of steps: I opened "Settings" -> "Wireless & Networks" -> "Mobile Networks" -> "Access Point Names" -> "Telkila". Once there I set the proxy host name in the property "Proxy" and the Proxy port in the property "Port".

18 Discussion of Proxy with Android Emulator @ Stackoverflow.com 2010-2011
7.3.2 Frequent loss of connectivity by Android Emulator.

This is another abnormal event associated with Android Emulator. Based on my research, it appears to be a problem that few other Android developers understand, possibly due to lack of collective experience. I’ve encountered this error first while switching between the login and logout function, the application would login just fine initially however after being logged out and upon subsequent login attempt it would display “connecting…” message which I associated with searching and establishing of internet connection. I initially looked at the authentication modules to identify root cause of this error; however I was able to replicate this while working with the internal emulator’s browser as well. Specifically I’ve noticed that “wireless network connection” modules inside of the Emulator disconnects from the network at random from time to time without any apparent reason. This may in part have something to do with the proxy issue described above, but I believe that at present it is outside of the scope of this project to carry on further research on this issue.

How did it impact this project? I initially looked inside the authentication module to identify the root cause of the issue and to try to re-work the module. After confirming that this error cannot be replicated at all times during execution of this module I looked at other potential causes before actually identifying this system-wide abnormality. The immediate impact on the project was that it set project’s development schedule by at least one working day. Additionally, I still have to restart some functions from time to time while using the software whenever this issue is encountered.

Why wasn’t it considered as a risk and mitigated? This was an unknown that I had never accounted for during initial design phase of the project. I suspect this error might be specific to a particular release of Android Emulator or a specific build. Something similar is described on developer forums and other online resources[^19].

How can it be avoided next time? There is no immediate fix or resolution that is available as an update to the Android Emulator. I wasn’t able to figure out a manual fix to this issue either. I have tried to change different wireless and network settings with Android Emulator but wasn’t able to get any of those changes to help with this particular error.

7.4 Technical Issues

This section describes a number of significant technical issues worth emphasizing. Please note that not every issue experienced during the development phase of this project is covered. Most issues have either been mentioned earlier in this paper or are relatively common to be expected within any development project of such scope, size, and effort.

7.4.1 What Went Badly

Among numerous issues experienced throughout this project, there is a few that stand out based on the extensive time spent on resolving those issues, importance of such issues to the entire project, as well as the contribution that a resolution of such issue would bring to the overall Android development community. Unfortunately, issues described in this section have never been resolved. Instead, for each problem, a workaround has been introduced which allowed me to carry on development with only slight changes to the original scope of the application.

7.4.1.1 File Picking error within File Browser module.

While introducing a third-party file browser module to the application I read complaints about file selecting methods used within this function described on module’s development forum. This File Browser function was originally developed in late 2007 to serve a small number of early Android developers doing some of the first mobile applications, and as such it was built on a platform which was in its infancy, and in many versions didn’t yet separate itself from classic Java development.

In my search for qualified compatible third-party functions to integrate within this project I have always relied on a rule that those open source modules with a sizable community of users must be the most tested, and therefore the safest to use especially in terms of quality and code stability. Such widely available functions may not always have the most elegant up-to-date solutions; however they usually have an extensive forum knowledge base which often serves as both a database for possible error fixes as well as a suggestion box for more elegant variations and solutions.

As I untangled the File Browser function and integrated it within my existing code structure by changing XML layout documents, adding appropriate classes, and linking those classes to the main activity class, I followed the same assumption picking a module that I found to be widely used and popular among developers. After an initial test run, I’ve encountered small bugs and issues which were successfully resolved through simple

---

20 “Building an Android FileBrowser” Tutorial # 67 @ http://www.anddev.org/building_an_android_filebrowser_list-based_-t67.html
testing, trial and error, as well as by help from a support forum\textsuperscript{21}. I wasn’t however able to resolve one particular error that jeopardized usability of the entire module. During the test run, upon picking a particular file from a group of files in the File Browser’s window, my application suddenly crashed. While running through Eclipse’s debug mode I was able to pinpoint the issue to the following snippet of code

\begin{verbatim}
    int selectionRowID = (int) this.getSelectionRowID();
\end{verbatim}

The specific error shown by the compiler stated that method getSelectionRowID is of an unidentified type. My initial assumption was that there might be some sort of library or API call specific to Android that I was missing or didn’t import correctly. I have experienced those kinds of issues before often caused by advances in development of specific standards for Android Java. After extensive search on Google and other online resources I have returned back to the original application’s forum and in one thread\textsuperscript{22} discovered that ListActivity.getSelectionRowId() was removed from API by Android development team and that it was recommended to use ListActivity.getSelectedItemPosition() or getSelectedItemID() instead. However, the new function may often return no selection (-1 or 0), especially when the emulator or device is in touch mode.

While trying out the new approach I’ve experienced a null pointer (both -1 and 0 for each of the suggested methods) error and found a lot of useful material online advising on how to resolve this issue. None of the methods however have really worked for me. As I have struggled to get this working, I am sure the correct solution is not too far off for I am convinced that getSelectedItemId() & getSelectedItemPosition() methods exist and are being used by other developers. Therefore, someone probably figured out how to produce an elegant solution to accommodate for exceptions and data checking.

In this particular case however, I used an alternative roundabout solution by rewriting the original method responsible for identifying a picked file. I’ve introduced getItem() method which identifies a file name selected without ever determining its ID. I was therefore forced to recode the rest of the method due to the fact that it relied on the item ID which I was never able to identify. In order to help other developers with a similar issue, I’ve posted my solution to my blog and linked to it from the support forum.

7.4.1.2 Multi-threading error: wait/notify not working as intended

\textsuperscript{21} “Android FileBrowser v2.0” Advanced Tutorial # 21 @ http://www.anddev.org/advanced-tutorials-f21/android-filebrowser-v-2-0-t101-45.html
\textsuperscript{22} “Building an Android FileBrowser” Tutorial # 67 @ http://www.anddev.org/building_an_android_filebrowser_list-based_-t67-s15.html
During development of the ViewGame module, the original design of the function allowed for a previously identified game to display one move at a time with an interval of five seconds until all of the moves are played or until the pause button is pressed. A viewer of the game would have an option of pausing the game at any time for analysis or any other reason. Such experience is customary in conventional chess viewing and I had tried to have as much of this “conventional” chess experienced be used within the chess application.

In order to achieve this experience and adhere to connectivity and reliability standards proposed for the online portion of the application, I planned to have a series of methods to download all of the game information of a previously identified game in a two dimensional array, parse this data according to specific standards/rules used within this application, and output one move at time with a five second delay between moves, and use wait/notify methods to wait for a pause button event.

Both five second delay and pause button notification require use of multithreading. In my previous programming experience I have had very limited involvement with threads and I have done some initial research on the use of wait/notify methods in connection to threads specific to Android’s implementation of Java 23.

In order to call wait(), notify(), or notifyAll() on an object you must first own the monitor of the object you wish to call the method on, so in my case within the runnable I intended to do the following:

```java
Runnable runnable = new Runnable() {
    public void run() {
        // wait(); This call wouldn't work
        synchronized (this) {
            wait(500); // This call will work – waits for 5 seconds
        }
    }
};
```

Using this simple example, I intended to call synchronized() method after every move to wait for five seconds before continuing to the next move in a loop. However, when I ran my application I saw a long delay and then all of the moves played at once. I wasn’t able to pause the working thread for 5 seconds, instead the working thread continued to run on the background while the game was on pause for 5 seconds x N number of moves.

---

23 “Sleep a thread until an event is attended in another thread” StackOverflow 2008-2011 http://stackoverflow.com/questions/121762/sleep-a-thread-until-an-event-is-attended-in-another-thread
I also experienced an issue trying to pause a particular thread through the use of the pause/play button. Creating the pause/play experience using onclicklistener was a relatively simple task; however, trying to call wait() or notify() methods from the onclicklestener proved to be difficult. According to my original design every time a pause button is pressed, a wait() function is called, while a play button is pressed a notify() would be called instead.

Once I wasn’t able to resolve both threading issues, an alternative solution had to be designed and implemented. The immediate impact on the project was that it delayed the project’s development schedule by at least six working days. I had to redesign the entire experience by giving the user more control of the view game module. Based on the alternative solution, a game viewer would have to click the “next move” button for every additional move being played on the board one move at a time.

7.4.2 What Was Lacking

There were a few nice-to-have features which were considered for this application but have been left out of this project. However, it is important to discuss them based on importance of such features to the entire project, prevalence of these features in other applications, as well as contribution that introduction of these features would bring to the overall Android development community. Therefore, the features below should be definitely introduced in the next development phase of this project.

7.4.2.1 Creation of an interactive web portal with AWT interfaces (Java applet) to support gaming capabilities.

The original web server design underwent certain scope changes during the implementation of this phase of the application development; many of its features have been excluded and the web server became responsible for responding to requests sent by the client application with no direct user access at this time. However, the next phase should include a fully functional web portal with a front-end interface including Java applet gaming interfaces. A user of the web portal should be able to utilize a full spectrum of features available within the client’s interface including being able to conduct multi-player games with users of mobile devices.

The detailed project plan, architecture design, required hardware components, and the full scope of work must be finalized prior to any scheduling during the next phase of the application development.
7.4.2.2 Secure, SSL type of connection for Client’s application.

The current implementation of online gaming modules uses simple HTTP connection requests to facilitate transfer of information between the client’s application and the web server. This implementation is sufficient to get low sensitivity game data transferred; however it may be vulnerable to various security bridges affecting operating systems of the machines running the client’s software, as well as the integrity of more sensitive data, such as live games being played in international tournaments.

In order to accommodate a secure connection between the client’s application and web server, the next phase of the software must feature https connections established using SSL certificates. There are a few acceptable solutions that could be applied. For an actual handshake to occur, you have to let Android know of your certificate. If you want to just accept no matter what, then use the pseudo-code as a starting point to get what you need with the Apache HTTP Client. Another approach is to use a self-signed SSL certificate to be accepted by a backend web service. Since a given certificate isn't signed by a CA that Android trusts by default, the software needs to add the web server's public certificate to the Android app's trusted store.

7.4.3 What Went Well

Among numerous successfully resolved technical problems throughout this project, there are a few that stand out based on the extensive time spent on resolving those issues, the importance of such issues to the entire project, and the contribution that resolution of such issues would bring to the overall Android development community. Successful resolution of the problems listed below allowed for this project to carry on with as few scope changes as possible in close proximity to the actual time line/schedule with the limited technical and human resources available. Every one of the problems listed below could have stopped the project from moving forward if it wasn’t successfully resolved.

7.4.3.1 Integration of third party modules

As previously stated, one of the key development tools that was used during this project is the ability to introduce third party solutions for specific functionalities (modules) and

---

24 “Https Connection Android” discussion on StackOverflow @ 2009-2011
http://stackoverflow.com/questions/995514/https-connection-android

25 “Android: Trusting SSL certificates” by Bob Lee, 02/2010,
http://blog.crazybob.org/2010/02/android-trusting-ssl-certificates.html
to integrate those solutions within the current body of code and architecture while changing third party code to fulfill specific goals of this project. Aside from the usual cosmetic and functional programming issues expected with this kind of integration, the author ran into code compatibility and version compatibility issues. The Android platform is a new and dynamic operating system with major changes added to every release. However, the Android’s version of the Java language also changes with every Android OS modification. Many original libraries supported by the standard Oracle's Java implementation are changed to Android Java specific solutions which are internally supposed to make the implementation mobile friendly while saving hardware and processing resources. This evolution of Android Java often makes some code rewritten for a previous version not readily compatible with later version of Android. The same issue is also relevant with re-use of third party programming solutions (modules/classes/or functions) within larger software applications.

An example of successful integration is the third party authentication module that was integrated within the application. The original implementation created by Ramesh Pauldurai26 featured login capabilities to connect to a web server using an HTTP connection. The SAX parser method is used to retrieve the XML output. The application navigates a user to the welcome page once he/she logged in successfully (with a login failed message displayed otherwise). The implementation stores the username and password in SharedPreferences after the successful login; thereafter the application navigates a user directly to the main page once the user completes the login successfully.

During integration of the third party module the following changes were introduced to the original code structure in order to make the module compatible with the rest of the application as well as to be in line with the scope of this phase of the development process.

SAX Parser functionality responsible for parsing XML input data was removed and replaced with a non-XML parsing method created from scratch. If the SAX Parser is used, once an instance of this class is obtained, XML can be parsed from a variety of input sources. These input sources are InputStreams, files, urls, and SAX InputSources. However, the properties of the XML file necessary for the SAX Parser to be used weren’t being properly set by the web server. After encountering this roadblock, I changed the implementation to call a specific server-side page (login.php) with URL variables such as username and password and to read the text displayed on the login screen. If the text is “yes”, to the system will allow for login; otherwise the system will show an error and ask the user attempt login again. Subsequently, I also created a new XML parsing method to

allow the application to parse chess games retrieved from the web server in the XML format.

Logout functionality is a major issue that was left unresolved within the original login function. I added this feature by changing the welcome.Java file and the associated XML layout. Inside the onCreate() function, I added a button that deletes username and password from the Preferences global variable. I also added a new Handler function to redirect to another activity once logout is executed.

The Welcome screen has been significantly redesigned from its original version including a complete remake of the XML layout. Instead of displaying a simple welcome message, it now presents four buttons: Start Game, Join Game, View Game, and Logout. The StartGame button calls StartGame.Java, while Join Game and View Game call JoinGame.Java and ViewGame.Java, respectively. Logout buttons clear the values of global variables for username and password and delete those from the global preferences queue, resetting the login status of the user to guest. A detailed outline of GUIs is also provided for reference within the Requirements & Specifications section above.

7.4.3.2 Android Activity Cycle.

One of the biggest development accomplishments during the implementation phase of this project was the learning of the core differences between classic Java and the Android version of JAVA. At the core of this difference is the concept of Activity. Learning how to use and implement activities is essential to any Android development.

Activities in the system are managed in an activity stack (LIFO). When a new activity is launched it becomes the top of the stack. Any previous activity will be below it and won’t come to the top until the new one exists. The application on the top of the stack has the highest priority from the operating system. While the activity that is not visible has lower priority even if a running activity requires more memory, the system can shut down that activity to free memory.

Android runs each activity in a separate process each of which hosts a separate virtual machine. Android saves metadata (state) of each activity so that when a new activity launches, it can come back to the activity when the user backtracks.

The activity can be in one of four states:

1. Active: the activity started, is running and is in the foreground.
2. Paused: the activity is running and visible but another activity (non-full sized) is running on the top or a notification is displayed. The user can see the activity but
cannot interact with it. A paused activity is fully alive (maintains state and member information) but can be killed by the system in low memory situations.

3. Stopped: the activity is running but invisible because the user has launched another activity that comes to the foreground. The activity is alive (maintains state and member information) but can be killed by the system in low memory situations.

4. Dead: either the activity is not started or it was in the pause or stop state and was terminated by the system to free some memory or by the user asking it to do so.

The Activity class provides several functions that the developer may override as needed to control the details of the Activity lifecycle. In practice, most Activities only need to override a few of these methods. The following gives a detailed description of all the methods. The way that they interact with the Activity lifecycle can become quite complex.

This figure shows the states of the activity and the methods that handle each state.
The sequence is as follows:

The activity starts, passes through `onCreate()` and `onStart()`. The activity is still not visible to the user until `onResume()` when it comes to the foreground and becomes fully running.
If another activity launches or a notification appears, the activity passes through the onPause() method. Then there are two scenarios:

1. If the system decides to kill your activity due to low memory, the activity starts the cycle again from the onCreate() method with the Bundle savedInstanceState parameter that holds data about the previous state of the activity.

2. In order to resume a paused activity user need to close the new activity or send a notification to the paused activity in order for the activity cycle to continue from the onResume() method.

When the user is about to close the activity, the activity calls the onStop() method and then the onDestroy() method which destroys the activity.

But if another activity runs while the current one was not shut down, the activity calls the onStop() method and if it is not killed by the system it will call the onRestart() method and then the onStart() method, and continues the cycle.

**onCreate():** will be invoked in three cases:
- the activity runs for the first time and it will be invoked with a null Bundle savedInstanceState parameter.
- if the activity had been running and then was stopped by the user or destroyed by the system, then it would be invoked with Bundle savedInstanceState that holds the previous state of the activity.
- if the activity is running and you set the device to different resource settings like Portrait vs. Landscape, then the activity will be recreated.

This method creates a user interface, binds data to controls, and registers the event handlers for the controls. Then it is followed by the onStart() method.

**onStart():** invoked when the activity is first launched or brought back to the foreground. It would be followed by onResume() if the activity continues and comes to the foreground, or by onStop() if the activity is killed.

**onRestart():** invoked in case the activity has been stopped and is about to be run again. Always followed by the onStart() method.

**onResume():** invoked when the activity is about to come to the foreground and is on the top of the activity stack. It is the place where you can refresh the controls if the activity is using a service that displays some feeds or news. Always followed by the onPause() method.

**onPause():** invoked when another activity launches while the current activity is launched or when the system decides to kill the activity. In this method you have to cancel
everything you did in the onResume() method like stopping threads, stopping animations or clearing usage of resources such as camera.

This method is followed by onResume() if the activity returns back to the front or by onStop() if the activity is to be invisible.

onStop(): invoked when a new activity is about to come over the current one or the current one is to be destroyed. Always followed by onResume() if the activity comes back or onDestroy if the activity is to be killed.

onDestroy(): invoked when the activity is shutting down because the activity called finish() [terminated the activity] or because the system needs memory so it decided to kill the activity.

Conquering the Activity Life Cycle is key to successful development of Android’s Java.

7.4.3.2 Application of reverse engineering techniques

In my search on the topic of reverse engineering I outlined several prominent methods that could be used in this open source software[27]. The task of reverse engineering some important classes as it related to this project was much simpler due to the availability of a source code base. However, lack of any kind of documentation and the use of a foreign (Czech) language within comments, variables, methods, and class names made reverse engineering of this software package difficult.

Due to a lack of available resources I decided to concentrate on using the Tracing Input method to analyze parts of the existing Chess software that would be important in development of new enhancements. I identified several input points for each class that I was interested in reverse engineering. For instance, the Task class has a number of input points associated with coordinates of the board fields as well as with numerical combinations representative of chess pieces. Therefore, based on the type of input being sent to various methods of this class, I was able to derive the interworking of these methods as well as the dependencies with other methods within this class. I used the Log.d method to output command line comments in order to trace different data fields and compare their derived output with expected outputs based on my understanding of the software.

Establishing relationship and interaction patterns between various existing classes proved to be a challenging task as well. Following specific implementation guidelines set by the

---

Android version of Java, all of the interactions between classes within this software are facilitated through the use of a main activity class. This main activity class acts as a bridge between various other activities. I could clearly see an activity sending a request to the main activity class; however, tracing this request within the main class and then following data flows in reverse order was difficult to implement.

The Eclipse programming environment provided me with a number of tools to trace definitions and the use of variables, classes, and methods throughout the entire software package. Using a set of features provided by Eclipse, I was able to changes names of variables and methods at will, which helped me to translate the program and to understand data flows between methods and classes.

7.5 Project Issues

Below is a brief analysis of project issues not yet covered.

7.5.1. Top 10 issues faced by this project and their impacts.

1. Defining Project Schedule.

During the planning and requirements gathering phases of the project, it became apparent that project schedule and the period of time allocated for completion of this project would play a major role in defining the specifications and the scope of this phase of the development.

The schedule and timeline specified in the original proposal of the thesis presented to Computer Science Department at CSUN have had an impact on the development process. For instance, the Extreme Programming development paradigm was specifically chosen to make sure that a working deliverable was available after completion of every milestone/module.

2. Adjusting Project Scope.

During the requirements and design phases of this project additional features and functionality were identified. Some features, such as development of a new XML parser, were introduced during the development phase; others were left for subsequent releases due to resource and scheduling constrains. Additionally, development of automated language translation in order to port the client side application onto iPhone and PC platforms (which was a key feature of the project during initial requirement analysis) was left for subsequent phases of development due to programming limitations, lack of reliable existing technologies to handle such translation, and resource and scheduling
constrains.

The main impact of adjusting the project scope was development of a working client-side application and web server on schedule, with adherence to core software requirements. Another effect was introduction of a modularized development process to allow for addition of new features without extensive re-work of the code structure developed during this phase of the project.

3. Using an existing Open Source application as the starting point.

Using an existing Open Source application (Chess game) as a starting point in the development of a two component client/web server software helped to get the project development off the ground and allowed it to stay on schedule while completing major online and multiplayer components. It also allowed developing an object oriented and modular software structure based on the existing code structure used within the open source application.

The major setback of this approach was the fact that the open source structure of this application doesn’t support automatic translation. The application uses Android-specific Java packages (with classes such as Activities) with many of them lacking one-to-one correlation with standard Java API. Therefore, in order to translate such application to be used on a PC, many of the existing classes had to be re-coded with standard Java utilities (Java.util) and other packages. Once such adjustments are done, it is possible using some of the early adopted translation tools to recompile this code into Objective C supported source code to be used on Apple’s iOS.

4. Getting the code compiled.

A major obstacle during the early stages of the development process was compiling the original open source software used as a basis for the client’s side applications. Many of the compiler-related issues have been described earlier. However, it is important to point out because of the impact of the delay within the overall development schedule as well as introduction of Android 2.2 specific programming solutions within this application.

5. Web Server development/set-up.

Development of a simplified web server structure without a front-end interface diminished the ability to use some of the commonly available services (such as account login, and game play) directly. On the other hand, this approach required fewer development resources; supported development of online services for client based applications, and supported the timeline and scheduling parameters set for this phase of the project.
6. Using Extreme Programming (XP) techniques.

Extreme Programming allows focusing on coding and avoiding needless documentation and assets. It provides a structure that allows creation of a working product at every development milestone. It gives a frequent feeling of achievement, and generally allows production of complete code and interfaces. Therefore, XP creates working software faster, and that software tends to have fewer defects. It allows additional change of scope whenever such is needed, with minimal cost and almost no code rework. It produces reliable estimates to coordinate timelines and schedule easier.

However, this technique would only work well given enough quality assurance testing resources to test every product prototype at every development milestone. Lack of testing resources was a major challenge of this project given the fact that the sole developer is also the only QA resource on this project. Lower quality QA testing would lead to a higher likelihood of unstable and erroneous code which has a negative impact on the project and its schedule.

7. Adhering to a modularized, object oriented development approach.

Development of a modularized, object oriented code structure allowed for easier integration of third party application components which cut down on development time. However, keeping such a structure intact without any technical documentation proved to be a challenge. Following and identifying dependencies within interfaces of the application and applying those dependencies to additional interfaces and modules while benefiting from code reuse took additional time and development resources.

8. Code re-work.

Substantial rework of the original open source software was done in order to make it compatible with the new version of Android OS (Android 2.2). During this process, as was described earlier, some of the standard Java libraries were substituted by specific packages developed for Android’s version of Java. In addition to this, many of the methods used by the standard Java packages within the application were replaced. At times those replacement methods behaved differently from the ones they replaced.

The impact of such changes within the existing code was a less stable code structure with some methods (for example the getSelectedItemId() method described above) behaving unpredictably during different testing scenarios. Also, a significant schedule delay was experienced during the rework of the original code structure.


Effects and uses of third party components were widely expressed in different parts of this paper. It is however important to note that the overall impact of such modules was
lower development costs in programming of these features, unique experience in learning
the existing open source code structure as well as the programming solutions introduced
by other developers not related to this project, and significant contribution to the larger
Android community through introducing fixes for existing bugs or inefficiencies within
third party modules.

10. Additional user testing.

Software testing was a major issue during development of this project. Lack of adequate
resources, knowledge, and time for Quality Assurance often led to a simple error being
transmitted from one milestone to another which required substantial code rework in later
module integrations. As with individual phases of Extreme Programming lower quality of
QA testing lead to a higher likelihood of unstable and erroneous code which had a
negative impact on the project and its schedule. The later in the development an early
stage error is finally detected, the more costly it is to fix it.

7.5.2 Ability of the project to identify issues

Using Extreme Programming techniques allowed for early identification of potential
issues within new modules as well as integration of the existing code base. For instance,
use of a SAX Parser with authentication module has been deemed questionable and
substituted with an alternative non-XML solution. However, an XML parsing solution
was required during development of Online Game modules and prior identification of
issues within SAX Parser has allowed for timely development of another XML parsing
method which prevented unnecessary delays. Another example is identification of issues
during design phase of this project. During design of the Load Game function it has been
determined that all of the game data needs to be available within a previously saved game
file in order to properly execute loading of the saved game. However, during subsequent
analysis of the “save game” function already available within the original open source
software, it was determined that solution used within this function would not support
further reuse by the load function and cause a potential development error if left
unchanged. Detailed rework of the save function allowed elimination of unneeded
complexities during development of the load function.

7.5.3 Ability of the project to close issues.

In most development projects, existing issues are closed during quality assurance testing
where severity, probability, and impact of those issues are determined and appropriate
solutions recommended. In this project however, Extreme Programming development
environment allows closure of existing issues after every milestone, but the lack of
quality assurance assets undermines this benefit.
Therefore, in order to save time and preserve proper allocation of resources only the most critical issues effecting core functionality of the application have been closed while other issue have been mitigated to avoid system crashing and malfunctioning.

For example, during quality assurance testing of the View Game module it was determined that if a particular game contained analytic commentaries and if the length of the comments exceeds 300 characters per instance, an out-of-bounds error is displayed. To resolve this error, it was decided to skip any commentaries during this phase of the development until an alternative solution is developed.

7.5.4 Areas of the project suffered the most/least issues.

Most issues.

There are at least two areas of the project that suffered the most issues that are notable enough to point out. The issues encountered within those areas range from small technical issues to significant architectural roadblocks. What make the areas below stand out is the overall number and frequencies of the aggregate issues experienced during the design and development phases of this project.

1. Chessboard activity class

This activity is at the core of the entire project. This is where the main menu is set and it also acts as a gateway that helps other activities to communicate with each other. This allows us to achieve the modularity needed for a project of this scale. However, passing information to and from this activity while collaborating with secondary activities, which in time act like interfaces for other modules integrated within this project, caused many small to mid-size issues.

A typical activity interaction pattern is composed of the call to a “worker” class as well as a call from that class. For example the following call from the main chessboard activity to an OpenFile class

```
Intent newIntent = new Intent();
newIntent.setClass(this, OpenFile.class);
newIntent.putExtras(bundle);
startActivityForResult(newIntent, CHOOSE_FILE);
```

creates a variable of an object type Intent and passes a variable of type Bundle while halting execution of the main activity class until Intent object is returned.

On the other side of the call, OpenFile.class receives information from the main chessboard activity using the following call:
Bundle bundle = this.getIntent().getExtras();
String param1 = bundle.getString("FileName");

It returns information to the main chessboard activity via the following call

Intent newIntent = new Intent();
newIntent.putExtra("File", text);
setResult(result, newIntent);
finish();

As demonstrated in this example, every class of this application needs to originate from the main chessboard activity and results in passing information back to that activity. Following this chess board activity, the GUI generated BoardControl class is used to display any of the data processes by any of the classes on the chess board.

This complex arrangement results in loss of data, mishandling of Intent variables, incompatible variable types, and finally a much longer and interdependent (modularized) code structure.

2. Online modules

Issues related to connection interruptions described earlier, as well as errors experienced while developing multi-threading functionalities, made online modules some of the most troublesome during development of this project. The unstable code base resulting from lack of quality assurance resources allowed for some of the perpetual issues to carry over from milestone to milestone, taking away development resources and delaying scheduled deliverables.

For example, a common “play online” module developed to support the ViewGame, JoinGame, and StartGame modules had to be designed with an extra redundancy check to account for potential connectivity issues. The same check has been built into the web server to make sure that a “no new move” status represents a true lack of a move rather than a lost internet connection on the side of the client’s application.

Least issues.

Areas with the least issues were those developed by the original creator of the basic open source software, used as the basis for this application, and carried through this project without any major rework. This applies to virtually all of the basic rules (task classes) as well as to the Artificial Intelligence module.

Since the rules of a chess game used for this project are specific to the game itself and have little influence on the design of the online collaboration module or the web component of the software, there was no need to change any of the rules as long as those
rules were correct and tasks functioned without any issues in a modularized, object oriented manner. Similarly, the enhancing Artificial Intelligence capabilities was never in the scope of this project and therefore those capabilities were left as-is without any changes. In fact, understanding the interworking of the Artificial Intelligence module without supporting documentation would have been an extremely difficult task.

7.5.5 Areas of the project that suffered the most high priority issues.

Initial software setup and code integration were by far the most important issues threatening development of the entire project. As originally provided, the open source code of this software was completely out of date so running and compiling this code on any machine proved to be a major obstacle. Successful resolution of setup-related issues allowed me to move this project forward and enter the development phase.

Online Game modules (ViewGame, StartGame, JoinGame) were a source of continuous high priority issues. Most major issues concentrated around frequent network connection timeouts and multithreading errors described earlier.
Chapter 8: CONCLUSION

This project considered the feasibility of effective and stable implementation of a paired client/web-based software package that is integrated among multiple mobile and stationary clients utilizing a mix of operating systems. The main objective was to provide a proof of concept, through a working prototype, of an effective and stable cross platform software implementation that consists of a client-based installation and a web-based platform.

The two primary goals of the research were to create functionality responsible for online real-time collaboration by multiple users and increase efficiency through the use of a web server.

The secondary goal was to create a client side application compatible with various architectures and operating systems through the use of automatic code translation.

A number of other milestones have been considered and achieved:

1. Direct contribution to the mobile development community.

2. Gaining experience with the Android development platform, setting up a software workbench and having exposure to integration of various versions of Android-based applications.

3. Demonstration of an open source development process based on the extreme programming development model.

4. Case study of the development process by defining Requirements and Specification documentation through a process of reverse engineering of the existing open source software.

The prototype, based on a chess game running on the Android OS platform, was successfully developed meeting both primary goals of the research. Online user collaboration was achieved through introduction of online games modules to the existing open source chess software. These online modules are supported by a web server which is the second component in the two-tier software prototype. The types of collaboration supported by the software are active and passive. Active collaboration involves players participating in the game, while making changes to the particular instance of the game, with ability to save and load games at will. Passive collaboration involves game viewers being able to view any existing game currently played or saved in archives on the web.
server. Any game played or saved on the server can have any number of viewers with each viewer utilizing client side application to view a chess game.

The software’s performance was enhanced by placing the resource burden of the following activities on the web server: facilitating information exchange between different game participants, providing authentication and session recognition capabilities, processing game data, offering processed data in the parsed XML format, guaranteeing reliable access to online services, and providing storage and archiving capabilities.

The secondary goal of cross platform automation was not achieved due to the fact that existing technologies behind cross platform automation and automated language translation are not well developed and it is risky to base success of the project on this factor knowing that this may not be attainable. However, various components of the prototype were coded in a modular fashion to allow for cross platform automatic translation in the future.

Furthermore, all of the key milestones outlined in the objective section of the research paper were realized during design and development phases of this project. Notably, the contribution to the Android mobile development community was evaluated using web site setup with different Android specific programming solutions introduced in the development phase of the project to monitor user base response and interactions.

Since one of the goals of the project was to make a concrete contribution to the Android development community, I maintained a blog to publicize my Android programming related problems and solutions, and it received significant traffic. Over a period of two months 153 keywords contributed to Android development related searches were used to access the site with over 1,000 unique visitors entering the site through Google and other search engines. Additionally, the site is on the first page of Google for the following terms: android login logout, android play pause button, android dynamic button, and logout android activity. The most significant results were generated as follow-up to three posts on programming solutions outlining (1) login module, (2) use of dynamic play/pause buttons, and (3) an alternative listview functionality. Therefore, the projected impact from the publishing of entire body of the source code of this development project and associated documentation online is expected to also be well received.

An additional contribution was made to the specific open source chess game used as a prototype of this project. This open source game has been enhanced with multiplayer functionality, online gaming modules, comprehensive save and load features, and a web server to support online and processing activities. In this process, 75% of the existing

28 “Android Development Discussion” @ BestWebDesignSeo Blog 2010-2011 http://www.bestwebdesignseo.com/android-development/
code base has been changed and approximately twice times of the amount of the new code added to the program.

Another major contribution factor is the outline of the open source development process in a case study format that is still rare for Android development projects due to the limited amount of publicly available development documentation specific to any one project developed for Android OS. This case study outlines all of the major requirement and specification development processes and provides insight on the use of those in relation to a specific Android project. Upon future publication of this paper online, interest by the Android community is expected to be in line with the initial feedback to a few of the android programming solutions posted on the web and explained earlier.

Lessons learned, outlined in chapter 7, provide detailed analysis of important issues encountered during development of this project including management, process, and technical challenges. Significant emphasis is also placed on exploring unique attributes of Android specific Java implementation.

In conclusion, this research demonstrates that while there are some existing challenges with code automation that could be a road block towards implementing standard solution across various architectures and operating systems there is absolutely no doubt that an implementation of a paired client/web-based software package will allow for real time collaboration across multiple devices and computing mediums as well as increase overall computing performance of some of the more resource limited computing devices such as mobile smart phones.
Chapter 9: FUTURE WORK

9.1 Recommendations for Future research.

Cross Platform Automation is the single most important issue promoting further development phases. Because of the size and importance of the change supporting cross platform automation, this enhancement will not be a simple patch within an existing version of the software but will require an almost complete redesign of the existing structure to work with more generic Java utilities and API that would be easier to port to other platforms. At this time, due to the lack of resources and knowledge base, it was decided to move “cross platform automation” into a future development queue while developing all of the other requirements (such as picking the development platform) with an eye towards cross platform automation. Since existing technologies behind cross platform automation and automated language translation are not well developed it is risky to base success of the project on this factor knowing that this may not be attainable.

However, further during the life cycle of this software the original idea to create a cross platform software package based on some existing models with the web server and multigame functionality in pursuit of creation of the two component software package could be realized through additional development phases.

9.2 Working toward unfulfilled goals.

It is clear from previous research that complex functionality of a chess program, such as strict chess rules and artificial intelligence used by chess applications cannot be simulated via simple web instructions used by Titanium and PhoneGap Solutions. A new method of cross platform automation should be developed based on the concept of reverse engineering of the Android specific development libraries to standard Java development libraries supported by Oracle.

Once the processes of substitution and reverse engineering are completed, and the software is still able to run on the Android platform with as much code as possible written in standard Java, a Java solution suitable for the PC’s architecture could be compiled. Given enough similarity between the two, porting the Android Solution to something like OpenJDK is possible. There are efforts such as Shark, Zero, and Caciocavallo that vastly ease the port process (= no ASM, simple AWT peer) especially
considering that Android is implemented on a Linux based kernel\textsuperscript{29}. In addition, both iPhone and Android-based phones have a Jazelle-compatible processor\textsuperscript{30}, and therefore the possibility of future addition of Jazelle support to OpenJDK will give developers an opportunity to choose between light resource acceleration (Jazelle) and JIT.

In terms of iPhone-specific architecture the port is also possible. Only Apple has put a clause in the iPhone license that clearly forbids VM usage. However, these restrictions may undergo further challenges in court, especially in European jurisdictions.

\textit{9.3 Real live applications.}

A number of real live applications could potentially utilize a paired client/web-based software package approach integrated among multiple mobile and stationary computing devices to promote collaboration among users in desktop publishing, engineering, social networking, or business and accounting applications supported by stationary as well as mobile computing devices. These mobile devices often take the place of stationary computers or laptops and with the advent of new technologies increasingly strive to utilize the same type of applications (Adobe Photoshop, Microsoft Word, and alike) as the traditional computers.

As the number of smaller mobile devices increases, they take up a larger proportion of the total internet traffic; and more web-based applications are introduced in order to take most of the burden away from the client’s device on to the web. The unique conditions supported by the mobile technology, especially the ability to keep uninterrupted internet connectivity for prolonged time spans support increasing demand for sophisticated web-based applications. Examples of the most popular web-based applications include Google Docs and Microsoft Office Online.


\textsuperscript{30}“Jazelle technology for Java” http://Java.epicentertech.com/Archive_Embedded/ARM_Jazelle/Jazelle.pdf

87
DEMO

Demo is available upon request. Demonstration will also be provided during the thesis defense on December 9th, 2011 at 9:00 am in room JD 4508D on the campus of California State University, Northridge.
REFERENCES: An Annotated Bibliography


The reference contains detailed description and a source code of an open source chess product programmed in C#. This program was original considered by the author as an open source based to use for this project. Later during development of the project, many functional features and techniques described in the reference have been used within the chess prototype.

Characteristics:

Does it seem like a reliable and current source? – This is a reliable source. This open source project has received a high rating, it is well document, and has a high number of positive reviews. It is current, as C# has been in existence for only a few years.

Is the research biased or objective? – Research is objective because it is supported by a working prototype.

Are the facts well documented? – Extensive documentation is provided to support further development efforts.

Is this source scholarly, popular, some of both? – Resource is popular in nature; however, there is a large body of research that has been conducted before and during development of this project.


This resource provides information on ChessBrain which is a virtual chess supercomputer using the processing power of Internet connected machines. On January 30th 2004 ChessBrain made history by becoming the first distributed network to play a game against a single human opponent. This reference was helpfully to do research and understand interactions between online computer clients within a chess game. It also provided substantial software development documentation as well as basic chess specific software logic and other related information.
Characteristics:

Does it seem like a reliable and current source? – This is a reliable and a distinguished source with a large number of researchers in participation. This source is current and represents a significant innovation in super computer design as well as real live web participation.

Is the research biased or objective? – Research is objective because it is supported by a working prototype and a significant number of participants.

Are the facts well documented? – Extensive documentation is provided to support further development efforts.

Is this source scholarly, popular, some of both? – Resource is both scholarly and popular. It has real live applications however it is also a prototype to support a concept of parallel data processing and development of an interconnected supercomputer.


This text books was used as a reference to guide me through some of the interactions between client software and web based application as well as how to setup efficient web application to support client software and process data reliably. This book uses Java (J2EE environment) as its preferred development language which made it very relevant to this project.

Specifically, some of the areas where this reference was helpfully were: understanding importance of using XML as well as the mark up of the language; two-tier client server architecture information provided by this text book has guided me through some of the architecture design conducted for this project; web application design principles; and design of MySQL database structures.

Characteristics:

Does it seem like a reliable and current source? – This is a reliable and current college level text book used by institutions across United States.

Is the research biased or objective? – Information presented in this text book represents opinions of the authors based on their experience and understanding of the subject matter.

Are the facts well documented? – Many examples are provided to support major ideas presented in this book.
Is this source scholarly, popular, some of both? – This is a scholarly resource intended to teach students about the subject.

*Web Engineering: The Discipline of Systematic Development of Web Applications*, authors Gerti Kappel, Birgit Prýyll, Siegfried Reich, Werner Retschitzegger; published by John Wiley & Sons 2006.

This reference describes development of requirements for web engineering projects, modeling web applications, web applications architecture, design, testing, and security. The book also demonstrates the difference between software engineering and web engineering. Web Engineering has shorter lead times compared to its software counterpart, demonstrating the rapid prototyping and agile methods of development needed in response to this.

Specifically, I have benefited from understanding activities of requirements engineering including requirements verification and validation, requirements management. Discussion on impacts of legacy code and systems was also interesting and relevant to the original open source program that I was using as the base for this project.

**Characteristics:**

Does it seem like a reliable and current source? – This is a reliable and current college level text book used by institutions across United States.

Is the research biased or objective? – Information presented in this text book represents opinions of the authors based on their experience and understanding of the subject matter.

Are the facts well documented? – Many examples are provided to support major ideas presented in this book.

Is this source scholarly, popular, some of both? – This is a scholarly resource intended to teach students about the subject.


This resource is a template for Software Test Plan Documentation. It is created by IEEE to use in development of software applications. This template has helped me to identify major software testing processes and apply them to my writing and during development of the prototype. Specifically I used this document to identify different testing approaches.
and to do more research to define relevant testing mechanisms for Extreme Programing Development model.

**Characteristics:**

Does it seem like a reliable and current source? – This is a reliable and current source provided by IEEE as a guide for software developers.

Is the research biased or objective? – This guide is an objective outline provided by a world’s largest professional association for the advancement of technology.

Are the facts well documented? – This outline does not have any facts.

Is this source scholarly, popular, some of both? – This is a scholarly resource for use as an outline for any software development project.


This reference article describes Android Activity Life Cycle which is one of the most fundamental features of Android application development. I have used this article to understand how the activity life cycle works, what it is intended to do, and what its components are. I have also learned how different activity cycles interact with each other as well as with other methods within each class.

**Characteristics:**

Does it seem like a reliable and current source? – Mina Samy has published at least four dozen articles on different aspects of Android Application Development over the past two years and has been well received by Android development community.

Is the research biased or objective? – Information presented in this article is objective and based on programming structure of Android specific implementation of Java.

Are the facts well documented? – Facts are supported by explanations and programming examples.

Is this source scholarly, popular, some of both? – This is a scholarly resource intended to teach general public about the Android development.
This reference is a presentation describing features of Dalvik Virtual Machine and how it is integrated with Android OS and supports Java development within Android. I have used this presentation for my research and to decide which language and system to choose for my prototype. Distinct features of Dalvik VM are that it can run on slow CPU, with little RAM, on an OS without swap space, and while powered by a battery. These features are representative of hardware used by a large percentage of mobile phones. These machines are representative of a sample that could use the prototype developed in this project.

**Characteristics:**

Does it seem like a reliable and current source? – This presentation has been given by Google during presentation of Android OS in 2008.

Is the research biased or objective? – This presentation is biased used to show advantages of the new operating system and associated virtual machine.

Are the facts well documented? – Facts are stated and some explanation is provided.

Is this source scholarly, popular, some of both? – This is a popular resource used to outline advantages and features of Dalvik VM.

---


This resource describes characteristics of a client-server system as well as its typical architecture. A comparison between client-server and peer-to-peer systems is made and briefly analyzed. A set of challenges associated with client-server system is also discussed. I have used this resource as a starting point in my research for principles of client-server system.

**Characteristics:**

Does it seem like a reliable and current source? – Wikipedia is community created and moderated online encyclopedia. It is constantly updated by its community members; however reliability of this resource can be questionable due to the fact that community members editing and moderating this source are of various skill levels.

Is the research biased or objective? – Research is biased in nature due to the fact that it represents opinions of individuals with various levels of expertise. However, given a
large enough number of these opinions and appropriate community moderation efforts, Wikipedia research on a particular subject matter tends to be objective over a long period of time.

Are the facts well documented? – Facts are documented and supported by references.

Is this source scholarly, popular, some of both? – This is a popular resource used as a quick reference online encyclopedia

_Larry Dignan, “Oracle vs. Google over Java: Android lawsuits may begin to pile up”_  
_Copyright @ ZDnet August 2010 <http://www.zdnet.com/blog/btl/oracle-vs-google-over-Java-android-lawsuits-may-begin-to-pile-up/38019> Retrieved 2010-08-24

This resource provides an overview of the lawsuit filed by Oracle against Google over changes made to Java programming language. This lawsuit highlights all main changes to Android’s version of Java which caused it to deviate away from standard Java implementation. Many of these changes affected this project making it impossible to port this software to a standard version of Java used by PC. I have reviewed this document to understand how the Java language is changing and what business processes have supported this change.

**Characteristics:**

Does it seem like a reliable and current source? – Source is reliable. U.S. Federal Court documents are quoted.

Is the research biased or objective? – Information is objected presented in a news report format.

Are the facts well documented? – Facts are documented and additional references are available.

Is this source scholarly, popular, some of both? – This is a popular source representing new report with analysis provided by industry experts.


This resource is an online forum for developers to share their ideas and answer questions. It provides specific programming examples frequently used and foot noted within my research. I have often used this resource to resolve issues with my prototype as well as to
find alternative solutions to programming certain features. Some of the abnormal events such as frequent loss of connectivity or issues with proxy with Android development environment have been resolved using this resource.

Characteristics:

Does it seem like a reliable and current source? – This is a current source frequently updated by developers. It is reliable based on the fact that most examples and solutions are given in code and code be tested by interested parties. Unreliable suggestions are always pointed out and often removed.

Is the research biased or objective? – Information presented in this forum is objective and based on programming solutions.

Are the facts well documented? – Many examples are provided to support ideas presented in the forum.

Is this source scholarly, popular, some of both? – This is a popular resource used to support development efforts of an online programming community.


This is another online forum used by developers to share their ideas and answer questions. It provides specific programming examples, resolves conceptual issues, and abnormal behavior of hardware/software resources. I have used this forum to gain additional understanding of an abnormal behavior experienced during the use of Android Emulator.

Characteristics:

Does it seem like a reliable and current source? – This is a current source frequently updated by developers. It is reliable based on the fact that most examples and solutions are given in code and code be tested by interested parties. Unreliable suggestions are always pointed out and often removed.

Is the research biased or objective? – Information presented in this forum is objective and based on programming solutions.

Are the facts well documented? – Many examples are provided to support ideas presented in the forum.
Is this source scholarly, popular, some of both? – This is a popular resource used to support development efforts of an online programming community.


This resource provides an example of a third party code sample integrated within the application. It helped me to create a login screen and connect with the server using the Http connection. I have used this solution to build a comprehensive authentication and registration module to connect to the Web Server.

Characteristics:

Does it seem like a reliable and current source? – This is a reliable and current source. I have tested information provided and used it in development of the application.

Is the research biased or objective? – Information presented in this example is biased and based on the opinion of the author.

Are the facts well documented? – Integration example is provided to support the goal of this article.

Is this source scholarly, popular, some of both? – This is a popular source based on a specific programming solution rather than on a research.


This resource provides comprehensive overview of software application architecture design. It covers such important aspects as key principles of software architecture, architecture patterns and styles, and design fundamentals. I have used this resource to better understand software quality attributes in order to use them in my research and adhere to them in development phase of my project.

Characteristics:
Does it seem like a reliable and current source? – This is a reliable and current college level text book sponsored by Microsoft.

Is the research biased or objective? – Information presented in this text book represents opinions of the authors based on their experience and understanding of the subject matter.

Are the facts well documented? – Many examples are provided to support major ideas presented in this book.

Is this source scholarly, popular, some of both? – This is a scholarly resource intended to teach about the subject of software architecture.
APPENDIX A – USE CASES

This appendix describes in details the development of use cases identified during gathering of requirements and development of specifications. Deriving these use cases from application requirements required identification of one recognized high level requirement such as starting a setup board (Setup Board) for each use case.

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>Setup Board</td>
</tr>
<tr>
<td>Created By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Last Updated By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Date Created:</td>
<td>02-09-11</td>
</tr>
<tr>
<td>Date Last Updated:</td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

**Actors:** Internal User of the Client System

**Description:** User setups board to start a game at a specific position

**Trigger:** User navigates to the main menu, selects submenu and clicks on board setup.

**Preconditions:**
1. User must have the software running
2. User must know which position he/she wants to setup

**Post conditions:**
1. User sets up position as desired
2. User continues the game using pre-setup position as a starting point

**Normal Flow:**
1.1 User turn the software on
1.2 Clicks on Menu
1.3 Click on an arrow to active sub-menu
1.4 Click on Setup Board
1.5 User selects appropriate pieces and starts board setup
1.6 Board is refreshed every time new pieces is moved
1.7 User is done with the setup and presses ok

**Alternative Flows:**
<table>
<thead>
<tr>
<th>Exceptions:</th>
<th>1. This is for internal play only</th>
</tr>
</thead>
</table>
| Includes: | 1. Selecting a menu choice  
2. Submenu is selected  
3. Data is return to match every new position of every piece  
4. Board is refreshed with the new position |
| Priority: | 1. Select Setup Board  
2. Define pieces and board setup  
3. Return to game mode |
| Frequency of Use: | Every time new position not saved into the database is loaded |
| Special Requirements: | 1. Main Menu Interface  
2. Sub-Menu Interface  
3. Setup Board Interface |
| Assumptions: | 4. User clicks on Setup Board to define specific position and continue with game place |
| Notes and Issues: | If the board setup process is interrupted user would have to start a new next time the software is loaded. |

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>Flip Board</td>
</tr>
<tr>
<td>Created By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Last Updated By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Date Created:</td>
<td>02-09-11</td>
</tr>
<tr>
<td>Date Last Updated:</td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actors:</th>
<th>Internal User of the Client System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>User choose to switch sides during an internal game or before a game is initiated</td>
</tr>
<tr>
<td>Trigger:</td>
<td>User navigates to the main menu and selects flip board</td>
</tr>
</tbody>
</table>
| Preconditions: | 1. User must have the software running  
2. Game must be ongoing or just started |
<table>
<thead>
<tr>
<th>3. Game must be internal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post conditions:</td>
</tr>
<tr>
<td>1. The board is flipped on the screen</td>
</tr>
<tr>
<td>2. User is now able to play for the opposite site</td>
</tr>
<tr>
<td>Normal Flow:</td>
</tr>
<tr>
<td>1.1 User turn the software on</td>
</tr>
<tr>
<td>1.2 New Game is initiated</td>
</tr>
<tr>
<td>1.3 Clicks on Menu</td>
</tr>
<tr>
<td>1.4 Select Flip Board</td>
</tr>
<tr>
<td>1.5 Board is flipped</td>
</tr>
<tr>
<td>1.6 User has now controlled of the opposing side</td>
</tr>
<tr>
<td>1.7 Game can be continued</td>
</tr>
<tr>
<td>Alternative Flows:</td>
</tr>
<tr>
<td>1.2. User is engaged in an ongoing internal game</td>
</tr>
<tr>
<td>Exceptions:</td>
</tr>
<tr>
<td>This is for internal play only</td>
</tr>
<tr>
<td>Includes:</td>
</tr>
<tr>
<td>1. Selecting a menu choice</td>
</tr>
<tr>
<td>2. Returning command to re-setup the board</td>
</tr>
<tr>
<td>3. Allow for control of the opposite site</td>
</tr>
<tr>
<td>4. Triggering of AI Module</td>
</tr>
<tr>
<td>Priority:</td>
</tr>
<tr>
<td>1. Select Flip Game</td>
</tr>
<tr>
<td>2. Flip the board</td>
</tr>
<tr>
<td>3. Continue with the game place</td>
</tr>
<tr>
<td>Frequency of Use:</td>
</tr>
<tr>
<td>Occasionally</td>
</tr>
<tr>
<td>Special Requirements:</td>
</tr>
<tr>
<td>1. Game Interface.</td>
</tr>
<tr>
<td>2. Main Menu Interface.</td>
</tr>
<tr>
<td>Assumptions:</td>
</tr>
<tr>
<td>1. User clicks on Flip Board to play the opposite color.</td>
</tr>
<tr>
<td>Notes and Issues:</td>
</tr>
</tbody>
</table>

<p>| Use Case ID:           | 4          |
| Use Case Name:         | Human Human|
| Created By:            | Ed Sarkisov|
| Last Updated By:       | Ed Sarkisov|
| Date Created:          | 02-09-11   |
| Date Last Updated:     | 05-11-11   |</p>
<table>
<thead>
<tr>
<th>Actors:</th>
<th>Internal User of the Client System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>User chooses to play with another user or him/herself using the same device. Essentially both sides of the game will be user controlled.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>User navigates to the main menu, then to a submenu and selects Human-human</td>
</tr>
</tbody>
</table>
| Preconditions:  | 1. User must have the software running  
                  2. Game must be ongoing or just started  
                  3. Game must be internal |
| Post conditions:| 1. AI Module is no longer active  
                         2. User is now able to make moves for both sides |
| Normal Flow:    | 1.1 User turns the software on  
                        1.2 User navigates to Human-Human through a submenu  
                        1.3 User selects Human-human menu option  
                        1.4 User is able to make moves for both sides |
| Alternative Flows: | 1.1. User is engaged in an ongoing internal game |
| Exceptions:     | This is for internal play only |
| Includes:       | 1. Selecting a menu choice  
                          2. Returning command to change control of the board  
                          3. Allow for control of the opposite site |
| Priority:       | 1. Select Human-human option  
                          2. Switch board control  
                          3. Continue with the game place |
| Frequency of Use: | Occasionally |
| Special Requirements: | 1. Game Interface.  
                                3. Main Menu Interface.  
                                4. Sub-menu Interface |
| Assumptions:    | 1. User clicks on Human-human to get control of both sides |
| Notes and Issues: | |

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>Undo Move</td>
</tr>
<tr>
<td>Actors:</td>
<td>Internal User of the Client System</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Description:</td>
<td>Used to undo a previously made move</td>
</tr>
<tr>
<td>Trigger:</td>
<td>User navigates to the main menu and selects undo move</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>1. User must have the software running</td>
</tr>
<tr>
<td></td>
<td>2. Game must be ongoing with at least one move made</td>
</tr>
<tr>
<td></td>
<td>3. Game must be internal</td>
</tr>
<tr>
<td>Post conditions:</td>
<td>1. Last move placed has been returned</td>
</tr>
<tr>
<td></td>
<td>2. User/or computer is now able to make another move instead</td>
</tr>
<tr>
<td>Normal Flow:</td>
<td>1.1 User turn the software on</td>
</tr>
<tr>
<td></td>
<td>1.2 User start a new internal game</td>
</tr>
<tr>
<td></td>
<td>1.3 After at least one move user selects undo move from main menu</td>
</tr>
<tr>
<td></td>
<td>1.4 Last move has been reverted</td>
</tr>
<tr>
<td></td>
<td>1.5 Game can be continued</td>
</tr>
<tr>
<td>Alternative Flows:</td>
<td>1.5. User can undo additional moves using Undo Move menu option</td>
</tr>
<tr>
<td>Exceptions:</td>
<td>This is for internal play only</td>
</tr>
<tr>
<td>Includes:</td>
<td>1. Selecting a menu choice</td>
</tr>
<tr>
<td></td>
<td>2. Returning command to re-setup the board</td>
</tr>
<tr>
<td>Priority:</td>
<td>1. Select Undo Move</td>
</tr>
<tr>
<td></td>
<td>2. Undo the last move made</td>
</tr>
<tr>
<td></td>
<td>3. Continue with the game place</td>
</tr>
<tr>
<td>Frequency of Use:</td>
<td>Occasionally</td>
</tr>
<tr>
<td>Special Requirements:</td>
<td>1. Game Interface.</td>
</tr>
<tr>
<td></td>
<td>2. Main Menu Interface.</td>
</tr>
<tr>
<td>Assumptions:</td>
<td>1. User clicks on Undo Move to revert last move or moves played.</td>
</tr>
<tr>
<td>Notes and Issues:</td>
<td></td>
</tr>
<tr>
<td>Use Case ID:</td>
<td>6</td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
</tr>
<tr>
<td>Use Case Name:</td>
<td>Redo Move</td>
</tr>
<tr>
<td>Created By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Date Created:</td>
<td>02-09-11</td>
</tr>
<tr>
<td>Actors:</td>
<td>Internal User of the Client System</td>
</tr>
<tr>
<td>Description:</td>
<td>Used to redo the move that was previously undone</td>
</tr>
<tr>
<td>Trigger:</td>
<td>User navigates to the main menu and redo move</td>
</tr>
</tbody>
</table>
| Preconditions: | 1. User must have the software running  
2. Game must be ongoing or just started  
3. Game must be internal  
4. Last move must be undone first |
| Post conditions: | 1. The move previously undone has been played again  
2. User is now able to play for the opposite site |
| Normal Flow: | 1.1 User turn the software on  
1.2 New Game is initiated  
1.3 At least one move is played  
1.4 User selects main menu and clicks on Undo Move  
1.5 Last move played is reverted back  
1.6 User selects main menu and clicks on Redo Move  
1.7 Previously reverted move is displayed again  
1.8 Game can be continued |
| Alternative Flows: | 1.6. Additional moves are reverted back by clicking Undo Move within the main menu screen  
1.7 User selects main menu and clicks to Redo Move  
1.8 Last move undone is reverted back |
| Exceptions: | This is for internal play only |
| Includes: | 1. Selecting a menu choice, first Undo and then Redo  
2. Returning command to re-setup the board |
| Priority: | 1. Select Undo Move  
2. Select Redo Move  
3. Continue with the game place |
| Frequency of Use: | Occasionally |
| Special Requirements: | 3. Game Interface.  
4. Main Menu Interface. |
| Assumptions: | 2. User clicks on Redo Move to play the last move undone. |
| Notes and Issues: | |

| Use Case ID: | 7 |
| Use Case Name: | Move |
| Created By: | Ed Sarkisov |
| Last Updated By: | Ed Sarkisov |
| Date Created: | 02-09-11 |
| Date Last Updated: | 05-11-11 |

| Actors: | Internal User of the Client System |
| Description: | User chooses to prompt computer to make a move instead of the user |
| Trigger: | User navigates to the main menu and selects make move |
| Preconditions: | 1. User must have the software running  
2. Game must be ongoing or just started  
3. Game must be played against a computer  
4. Game must be internal |
| Post conditions: | 1. The move is made for the user by the computer  
2. Next move is played by the opposite side |
| Normal Flow: | 1.1 User turn the software on  
1.2 New Game is initiated  
1.3 Clicks on Menu  
1.4 Select Move  
1.5 Move is made  
2.1 User turns the software on  
2.2 User makes a move |
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2.3</strong></td>
<td>Computer is scheduled to make a move</td>
</tr>
<tr>
<td><strong>2.4</strong></td>
<td>User clicks on Move menu item and prompts computer to make a move without a delay</td>
</tr>
<tr>
<td><strong>Alternative Flows:</strong></td>
<td>1.3. User makes additional moves before selecting Move option</td>
</tr>
<tr>
<td><strong>Exceptions:</strong></td>
<td>This is for internal play only</td>
</tr>
<tr>
<td><strong>Includes:</strong></td>
<td>1. Selecting a menu choice</td>
</tr>
<tr>
<td></td>
<td>2. Returning command to re-setup the board</td>
</tr>
<tr>
<td></td>
<td>3. Allow for control of the opposite site</td>
</tr>
<tr>
<td></td>
<td>4. Triggering of AI Module to make a move</td>
</tr>
<tr>
<td><strong>Priority:</strong></td>
<td>1. Select Move</td>
</tr>
<tr>
<td></td>
<td>2. Make a move</td>
</tr>
<tr>
<td></td>
<td>3. Continue with the game place</td>
</tr>
<tr>
<td><strong>Frequency of Use:</strong></td>
<td>Occasionally</td>
</tr>
<tr>
<td><strong>Special Requirements:</strong></td>
<td>1. Game Interface.</td>
</tr>
<tr>
<td></td>
<td>2. Main Menu Interface.</td>
</tr>
<tr>
<td></td>
<td>3. Trigger AI Module</td>
</tr>
<tr>
<td><strong>Assumptions:</strong></td>
<td>1. User clicks on Move to prompt computer to make a move for him or a computer.</td>
</tr>
<tr>
<td><strong>Notes and Issues:</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Use Case ID:** 8  

**Use Case Name:** Select Best Move  

**Created By:** Ed Sarkisov  
**Last Updated By:** Ed Sarkisov  
**Date Created:** 02-09-11  
**Date Last Updated:** 05-11-11  

**Actors:** Computer powered by the AI Module of the Client System  
**Description:** Computer selects a best possible move to make every time such choice is requested  
**Trigger:** Game is started against a computer
| Preconditions: | 1. User must have the software running  
|               | 2. Game must be ongoing or just started  
|               | 3. Game must be internal  
| Post conditions: | 1. Move is made by a computer  
| Normal Flow: | 1.1 User turn the software on  
|               | 1.2 New Game is initiated  
|               | 1.3 User make the first move  
|               | 1.4 Computer makes the next move  
| Alternative Flows: | 1.3. User flips the board through Flip menu option  
|               | 1.4 Computer makes the first move  
| Exceptions: | This is for internal play only  
| Includes: | 1. Selecting a menu choice  
|               | 2. Returning command to re-setup the board  
|               | 3. Triggering of AI Module to select a move to play  
| Priority: | 1. Start Game  
|           | 2. Make move by computer  
| Frequency of Use: | Every time computer generated move is made  
| Special Requirements: | 1. Game Interface.  
|               | 2. Main Menu Interface.  
|               | 3. AI Module  
| Assumptions: | 1. Computer will be prompted to make a move  
| Notes and Issues: | Computer uses an algorithm to select best move possible.  

| Use Case ID: | 9  
| Use Case Name: | Declare Winner  
| Created By: | Ed Sarkisov  
| Last Updated By: | Ed Sarkisov  
| Date Created: | 02-09-11  
| Date Last Updated: | 05-11-11  

| Actors: | Computer using AI Module during game play  

106
| Description: | Computer determines the end of the game and picks a winner based on the position of the chess pieces. |
| Trigger: | Computer analyses the game to determine if the game is over after every move. |
| Preconditions: | 1. User must have the software running  
2. Game must be ongoing  
3. Game must be internal  
4. End of game situation must be developed |
| Post conditions: | 1. Winner is declared: blacks, whites, or draw  
2. Game is finished |
| Normal Flow: | 1.1 User turns the software on  
1.2 New Game is initiated  
1.3 Move is made  
1.4 Computer checks if the game is over  
1.5 If the game is over, winner is declared |
| Alternative Flows: | 1.5 Winner is not found and draw is declared or game is continued |
| Exceptions: | This is for internal play only |
| Includes: | Triggering of AI Module after every move to determine a winner |
| Priority: | Select a winner |
| Frequency of Use: | Every Move |
| Special Requirements: | 1. Game Interface.  
2. AI Module |
| Assumptions: | Every game played should end with some sort of a result |
| Notes and Issues: | Interrupted games can’t cause exception, either the game will continue or not. |

<p>| Use Case ID: | 10 |
| Use Case Name: | Quit Game |
| Created By: | Ed Sarkisov |
| Last Updated By: | Ed Sarkisov |
| Date Created: | 02-09-11 |
| Date Last Updated: | 05-11-11 |</p>
<table>
<thead>
<tr>
<th><strong>Actors:</strong></th>
<th>Internal User of the Client System and External Environment (OS applications)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>User chooses to quit game for internal or online games alike</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>User interrupts game processes by quit it through the use of a cellular phone’s Operating System’s (OS) close application button.</td>
</tr>
</tbody>
</table>
| **Preconditions:** | 1. User must have the software running  
2. User selects to quit the game  
3. Game may be internal or an online |
| **Post conditions:** | 1. Game is exited  
2. Desktop and initial menu of the OS is presented |
| **Normal Flow:** | 1.1 User turns the software on  
1.2 New Game is initiated  
1.3 Clicks on external, OS specific menu to end the current application/process  
1.4 Game is exited |
| **Alternative Flows:** | 1.3 User can power off the device which would also quit the currently running application. |
| **Exceptions:** | This is for internal and online play alike |
| **Includes:** | 1 Selecting an interrupt from an external source  
2 Returning command to quit the game/software |
<p>| <strong>Priority:</strong> | 3 Select to quit the game via an external software source |
| <strong>Frequency of Use:</strong> | Every time the game is run |
| <strong>Special Requirements:</strong> | 4 Game Interface. |
| <strong>Assumptions:</strong> | 5 User doesn’t have any other way of quitting the game through the application itself |
| <strong>Notes and Issues:</strong> | This is an area of further improvement. |</p>
<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>Open Game</td>
</tr>
<tr>
<td>Created By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Last Updated By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Date Created:</td>
<td>02-09-11</td>
</tr>
<tr>
<td>Date Last Updated:</td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actors:</th>
<th>Internal User of the Client System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>User chooses to load an existing game on the board by using Load Game option within the game menu.</td>
</tr>
<tr>
<td>Trigger:</td>
<td>User navigates to the sub menu and selects Load Game option</td>
</tr>
</tbody>
</table>
| Preconditions: | 1. User must have the software running  
2. User must have a previously saved game in the correct format to load  
3. Game must be internal |
| Post conditions: | 1. Previously saved game is loaded and displayed on the board  
2. User is now able to play this game internally |
| Normal Flow: | 1.1 User turns the software on  
1.2 New Game is initiated  
1.3 Clicks on Menu  
1.4 Select sub-menu  
1.5 Clicks on Load Game  
1.6 A file browser window is not displayed  
1.7 User selects a previously saved game with a .pgn formatted file extension  
1.8 User confirms his/her selection  
1.9 Game is loaded within the system and displayed on the board  
1.10 User can continue playing the internal game |
| Alternative Flows: | 1.7 User doesn’t have a game to load and clicks back  
1.8 User is taken back to the chess board and the original settings  
1.7 User loads a game that is not of the correct extension  
1.8 User is taken back to the chess board with the original settings  
1.7 User loads a game is data that is corrupted or not formatted correctly |
<table>
<thead>
<tr>
<th>Exceptions:</th>
<th>This is for internal play only</th>
</tr>
</thead>
</table>
| Includes: | 1. Selecting a menu choice  
2. Active Browser module  
3. Load the game into an internal memory cache  
4. Returning command to re-setup the board  
5. Allow for normal game play processes |
| Priority: | 1. Select Load Game  
2. Activate Browser Module  
3. Refresh the game screen with the new data  
4. Continue game play |
| Frequency of Use: | Occasionally, whenever previously played game is loaded |
| Special Requirements: | 1. Game Interface.  
2. Main Menu Interface.  
3. Browser Interface |
| Assumptions: | User loads previously saved game to continue with game play |
| Notes and Issues: | New error handling must be introduced in future releases |

| Use Case ID: | 12 |
| Use Case Name: | Save Game |
| Created By: | Ed Sarkisov |
| Last Updated By: | Ed Sarkisov |
| Date Created: | 02-09-11 |
| Date Last Updated: | 05-11-11 |

| Actors: | Internal User of the Client System |
| Description: | User chooses to save the game that was playing on the board |
| Trigger: | User navigates to the main menu, sub-menu, and clicks save game |
| Preconditions: | 1. User must have the software running  
2. Game must be ongoing or just started |
| Post conditions: | 1. The game has been saved as a .pgn formatted file  
2. User is now able to continue playing the game |
|------------------|--------------------------------------------------|
| Normal Flow:     | 1.1 User turns the software on  
1.2 New Game is initiated  
1.3 Game is developed throughout normal playing experience  
1.4 User selects Main Menu  
1.5 User clicks on “more” to access sub-menu  
1.6 User selects Save Game option within sub-menu  
1.7 Save PGN screen is displayed with a number of fields to be completed  
1.8 User clicks “Save” button after completing all of the entries  
1.9 Message is displayed letting user know that the game is Saved  
1.10 User clicks back on the external menu to continue playing the game  
1.11 Game is now ready to be continued. |
| Alternative Flows: | 1.8. User clicks “Save” without assigning a name to the game  
1.9. A window with an error “Save error” is displayed  
1.10. User can click back and continue playing the game |
| Exceptions:      | This is for internal play only |
| Includes:        | 1. Selecting a menu choice module  
2. Activating Save game module  
3. Game play module |
| Priority:        | 1. Select Save Game  
2. Save game to a file  
3. Continue with the game place |
| Frequency of Use: | Occasionally, every time the game is saved |
| Special Requirements: | 1. Game Interface.  
2. Main Menu Interface.  
3. Save Game Interface |
| Assumptions:     | 2 User clicks on Save Game to save the game for future play. |
| Notes and Issues: | Define user friendly error messaging screens and options. |

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>Registration</td>
</tr>
<tr>
<td>Actors:</td>
<td>Internal User of the Client System and the Web Server</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Description:</td>
<td>User attempts to obtain access to the online game modules by registering with web server</td>
</tr>
<tr>
<td>Trigger:</td>
<td>User navigates to the main menu and selects Online games</td>
</tr>
</tbody>
</table>
| Preconditions: | 1. User must have the software running  
2. User must be connected to the internet through the devise that is running the client’s installation of the software |
| Post conditions: | 1. User obtains username and password to be able to play online through the client installation on the user machine. |
| Normal Flow: | 1.1 User turns the software on  
1.2 Clicks on Menu  
1.3 Select Online Games  
1.4 Login/Authentication screen is displayed  
1.5 User clicks on “register now” link  
1.6 Online Registration Screen is displayed  
1.7 User completes all of the necessary information and submits to the web server  
1.8 Web Server return “registered” message  
1.9 User is able to use the login information to access online games through login/authentication screen |
| Alternative Flows: | 1.8. Web Server returns an error “User name already exists within the system”  
1.9 User would need to select a different username and re-submit it to the web server until it is successful in processing. |
| Exceptions: | This is a registration/data screen only. No game play is actually activated |
| Includes: | 1. Selecting a menu choice  
2. Online module  
3. Registration module |
| Priority: | 1. Select Online Game  
2. Access Registration screen  
3. Submit successfully to the web server |
<table>
<thead>
<tr>
<th><strong>Frequency of Use:</strong></th>
<th>One time only</th>
</tr>
</thead>
</table>
| **Special Requirements:** | 1. Game Interface.  
2. Login Interface  
3. Registration Interface |
| **Assumptions:** | User clicks on Online Games to be able to play online. |
| **Notes and Issues:** | Design error screens that are more user friendly during the next release. |

<table>
<thead>
<tr>
<th><strong>Use Case ID:</strong></th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use Case Name:</strong></td>
<td>Login/Authentication</td>
</tr>
<tr>
<td><strong>Created By:</strong></td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td><strong>Last Updated By:</strong></td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td><strong>Date Created:</strong></td>
<td>02-09-11</td>
</tr>
<tr>
<td><strong>Date Last Updated:</strong></td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Actors:</strong></th>
<th>Internal User of the Client System and Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>User chooses Online Game module and needs to login within the web server before being able to play online</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>User navigates to the main menu and selects Online Games from the sub-menu</td>
</tr>
</tbody>
</table>
| **Preconditions:** | 1. User must have the software running  
2. User must navigate to Online Games  
3. User may not be already logged in. |
| **Post conditions:** | 1. User successfully logs in to the web server  
2. A number of online options: View Online Game, Play Online and etc. are displayed. |
| **Normal Flow:** | 1.1 User turns the software on.  
1.2 User navigates to Online Games through the sub-menu.  
1.3 Login/Authentication screen is displayed.  
1.4 User enters username and password.  
1.5 Server returns confirmation.  
1.6 User is presented with an Online Games: Options screen.  
1.7 User can choose different Online Games options thereafter. |
## Alternative Flows:
1.5. Server returns an error message
1.6. User is prompted to try to login again.

## Exceptions:
This is for online game play only

## Includes:
1. Selecting a menu choice module
2. Selecting login module
3. Authentication and Session storage module
4. Online Games: Options module

## Priority:
1. Select Online Game option
2. Logging to the web server
3. Get authenticated
4. View Online Options once logged in.

## Frequency of Use:
At least once if the cookies and sessions are not cleared occasionally.

## Special Requirements:
1. Game Interface.
2. Main Menu Interface.
3. Login Interface
4. Authorization Interface
5. Online Game options interface.

## Assumptions:
1. User clicks on Online Game in order to play online

## Notes and Issues:

### Use Case ID: 15

**Use Case Name:** ViewGame

<table>
<thead>
<tr>
<th>Created By</th>
<th>Last Updated By</th>
<th>Date Created</th>
<th>Date Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ed Sarkisov</td>
<td>Ed Sarkisov</td>
<td>02-09-11</td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

**Actors:** Internal User of the Client System, Web Server, and Online users

**Description:** User choose to view a game currently played online or an archived game stored online

**Trigger:** User navigates to the main menu, selects online games and chooses View Game
### Preconditions:
1. User must have the software running
2. Devices must be connected online
3. Web Server must have games to view

### Post conditions:
1. Online game is selected to view
2. Appropriate view game screen is loaded

### Normal Flow:
1.1 User turns the software on
1.2 User Navigates to Online Games
1.3 Once logged in user selects view game
1.4 View Game option interface is displayed with the available games
1.5 User selects a game among the list of displayed games
1.6 User is prompted to confirm the selection
1.7 Game is loaded within client’s internal database
1.8 Game is displayed one move at a time based on user control
1.9 If the game is lived, controls are disabled in real time

### Alternative Flows:
1.9 For archived games controls are available throughout the game

### Exceptions:
This is for online game play only

### Includes:
1. Selecting a menu choice
2. Returning command to re-setup the board
3. View Online Game Module
4. Authentication Module

### Priority:
1. Choose Online Games
2. Login to the web server
3. Choose a game to view
4. Display the Game

### Frequency of Use:
Occasionally

### Special Requirements:
1. Online Games Interface
2. Login Interface
3. View Game Interface: Choose Game
4. View Game Interface: Chess Board

### Assumptions:
User clicks on Online Games to play or view games originated or stored online

### Notes and Issues:

### Use Case ID: 16
### Use Case Name: StartGame

---

115
<table>
<thead>
<tr>
<th><strong>Actors:</strong></th>
<th>Internal User of the Client System and a web server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>User chooses to initiate an online game and waits until other participants join the game.</td>
</tr>
<tr>
<td><strong>Trigger:</strong></td>
<td>User navigates to the online games through sub-menu, logs in to the web server, and starts an online game</td>
</tr>
</tbody>
</table>
| **Preconditions:** | 1. User must have the software running  
2. Device must be connected to the internet  
3. User must have an authorized access to the web server |
| **Post conditions:** | 1. User created an online game and waiting for other users to join it |
| **Normal Flow:** | 1.1 User turns the software on  
1.2 User navigates to Online Games through sub-menu  
1.3 User navigates to Login Interface and inputs login data  
1.4 User selects StartGame from the Online Games interface  
1.5 User completes necessary fields within New Online Game profile  
1.6 User now navigates to a new online game screen |
| **Alternative Flows:** | |
| **Exceptions:** | This is for online play only |
| **Includes:** | 1. Selecting a menu choice  
2. Login to the web server  
3. Completing the Start New Game form  
4. Registering the new game with the web server |
| **Priority:** | 1. Select Online Games  
2. Login to web server  
3. Click on StartGame  
4. Complete New Game Online Profile |
| **Frequency of Use:** | Occasionally |
| **Special Requirements:** | 1. Game Interface.  
2. Main Menu Interface.  
3. Login/Authentication Interface  
4. Online Games Interface |
5. New Game Online Profile Interface  
6. Online Game Play Interface  

Assumptions: User clicks on Start Game in order to start a new online game.

Notes and Issues:

<table>
<thead>
<tr>
<th>Use Case ID:</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Name:</td>
<td>JoinGame</td>
</tr>
<tr>
<td>Created By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Last Updated By:</td>
<td>Ed Sarkisov</td>
</tr>
<tr>
<td>Date Created:</td>
<td>02-09-11</td>
</tr>
<tr>
<td>Date Last Updated:</td>
<td>05-11-11</td>
</tr>
</tbody>
</table>

| Actors: | Internal User of the Client System, external online user, and web server |
| Description: | User chooses to join currently available online game. |
| Trigger: | User navigates to the Online Game through the sub-menu, identifies him/herself and joins a currently available online game. |
| Preconditions: | 1. Device must have active internet connection  
2. User must be registered with the web server  
3. Game must be started by a different online user  
4. Game must be available to join |
| Post conditions: | Internal User joins a game successfully |
| Normal Flow: | 1.1 User turns the software on  
1.2 User selects Online Games from the sub-menu  
1.3 User logs in to the web server through the Login Interface  
1.4 User selects Join Game option  
1.5 User pick a game to join in the Available Games interface  
1.6 User selects a game a confirms game selection  
1.7 User starts playing an online game |
| Alternative Flows: | 1.5. No games are available in the Available Games Interface  
1.6. User is prompted to come back to join a game later |
<table>
<thead>
<tr>
<th>Exceptions:</th>
<th>This is for online play only and only if other online games are available</th>
</tr>
</thead>
</table>
| Includes:  | 1. Selecting a menu choice  
             2. Selection Login/Authentication module  
             3. Joining an online game  
             4. Triggering an online play module |
| Priority:  | 1. Select Online Games  
             2. Select Join Game  
             3. Find a game to join |
| Frequency of Use: | Every time a user wants to play an online game |
| Special Requirements: | 1. Game Interface.  
                           2. Main Menu Interface.  
                           3. Login Interface  
                           4. Online Games Interface  
                           5. Join Game (Available Games) Interface |
| Assumptions: | User clicks on Join Game to play against an opponent online |
| Notes and Issues: | Instead of prompting user to return, also refresh the screen and search for new games to join every couple of seconds. |

| Use Case ID: | 18 |
| Use Case Name: | PlayOnline |
| Created By: | Ed Sarkisov |
| Last Updated By: | Ed Sarkisov |
| Date Created: | 02-09-11 |
| Date Last Updated: | 05-11-11 |

| Actors: | Internal User, external online user, and a web server |
| Description: | User chooses to play an online game by starting a game or joining a game |
| Trigger: | User navigates to the online games through sub-menu and starts a new game or joins a game started by a different user |
| Preconditions: | 1. User must have the software running  
                           2. Device must be connected to the internet |
<table>
<thead>
<tr>
<th>3. User must have an online account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post conditions: User is able to play online with the opposite side</td>
</tr>
<tr>
<td>Normal Flow:</td>
</tr>
<tr>
<td>1.1 User turns the software on</td>
</tr>
<tr>
<td>1.2 User selects online games sub-menu option</td>
</tr>
<tr>
<td>1.3 User logs in to the web server</td>
</tr>
<tr>
<td>1.4 User presented with possible online options</td>
</tr>
<tr>
<td>2.1 User selects to Join a game online</td>
</tr>
<tr>
<td>2.2 User selects specific game to join</td>
</tr>
<tr>
<td>2.3 User confirm his/her selection and joins an online game</td>
</tr>
<tr>
<td>3.1 User is presented with a standard board setup and with the name of the opponent and as well as with the opponents rating and ability to send opponent messages</td>
</tr>
<tr>
<td>3.2 User makes a move on the board</td>
</tr>
<tr>
<td>3.3 Online player playing for the opposite side responds in turn</td>
</tr>
<tr>
<td>3.4 Game is continued in the same fashion</td>
</tr>
<tr>
<td>Alternative Flows:</td>
</tr>
<tr>
<td>2.1 User selects to start a game online</td>
</tr>
<tr>
<td>2.2 User completes the start form</td>
</tr>
<tr>
<td>2.3 User awaits for an online user to join a game</td>
</tr>
<tr>
<td>Exceptions: This is for online game only</td>
</tr>
<tr>
<td>Includes: 1. Selecting a menu choice</td>
</tr>
<tr>
<td>2. Selecting Online Games</td>
</tr>
<tr>
<td>3. Selecting to Start or Join Game</td>
</tr>
<tr>
<td>4. Triggering of Play Online Game module</td>
</tr>
<tr>
<td>Priority: 1. Select Online Game</td>
</tr>
<tr>
<td>2. Select Start Game/Join Game</td>
</tr>
<tr>
<td>3. Play Online Game</td>
</tr>
<tr>
<td>Frequency of Use: Every time an online game is played</td>
</tr>
<tr>
<td>Special Requirements: 1. Game Interface.</td>
</tr>
<tr>
<td>2. Main Menu Interface.</td>
</tr>
<tr>
<td>3. Online Game Interface</td>
</tr>
<tr>
<td>4. Login Interface</td>
</tr>
<tr>
<td>5. Join Game or Start Game Interface</td>
</tr>
<tr>
<td>Assumptions: User clicks on Online Games to play a game online against a live opponent.</td>
</tr>
<tr>
<td>Notes and Issues: Figure out how to determine if the game is over</td>
</tr>
</tbody>
</table>
APPENDIX B – Interfaces

The following user interfaces are used within this system. These interfaces work together to represent the playing environment in chess. The relationship between any of this interface and associated classes has been derived from the source code provided by the original developer. All of the interactions within these interfaces are defined via the associated modules while basic rules are set within the respective classes.

### 2. Menu and submenu

<table>
<thead>
<tr>
<th>Interface Identity</th>
<th>Main menu interface and submenu interface define menu options which represent different functions within the software.</th>
</tr>
</thead>
</table>

**Resource Provided**

| a. Resource syntax | Program Name: AktivitaSachovnice module  
Parameter Type: Input using mouse  
Parameter Info: Button  
Exception: None |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b. Resource semantics</td>
<td>Any buttons within menu and sub-menu interface identified by “more” button</td>
</tr>
<tr>
<td>c. Resource usage restrictions</td>
<td>Typing and other types of input</td>
</tr>
</tbody>
</table>

**Locally defined data types**  
Button graphic type

**Exceptions**  
None

**Variability provided**  
None

**Quality attribute characteristics**  
This is limited by the capabilities of the system and the device in terms of processing speed and number of requests processed.

**Elements Requirements**  
Every button needs to match the elements within the
Rational & Design Issues

1. What data is returned under normal conditions?
   Interface corresponding to a specific button.

2. What data is returned if the system is down?
   Standard system error. Software is closed or restarted.

3. What data should be returned?
   Interface corresponding to a specific button.

Implementation Notes

Introduce Help option describing different menu items within the next product release. Outside of the scope of this development cycle.

Usage guide

Menu items are self-explanatory but the help option will be provided within the next product release.

3. Save Game

Interface Identity

Save Game interface used to assign parameters of the save game and generate saving function.

Resource Provided

a. Resource syntax
   Program Name: SaveGame module
   Parameter Type: String, Chars, and system generated select inputs
   Parameter Info: Value input String type
   Exception: Strings with special characters

b. Resource semantics
   Any keyword without special characters

c. Resource usage restrictions
   Special Characters

Locally defined data types
   A string used to generated function name for a file

Exceptions
   Special characters not applicable for file names
<table>
<thead>
<tr>
<th>Variability provided</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality attribute characteristics</td>
<td>This is limited by the capabilities of the system and the device in terms of processing speed and number of requests processed.</td>
</tr>
<tr>
<td>Elements Requirements</td>
<td>Input fields are first validated for appropriateness before generating the file name and the file. Other fields correspond to specific behavior introduced during file generation as well as the content of the file itself.</td>
</tr>
<tr>
<td>Rational &amp; Design Issues</td>
<td>1. What data is returned under normal conditions? A file is generated within sdcard folder and a successful action message is displayed on screen 2. What data is returned if the system is down? Standard system error. Software is closed or restarted.</td>
</tr>
<tr>
<td>Implementation Notes</td>
<td>Create additional input validation within subsequent development phases. Outside of the scope of this project.</td>
</tr>
<tr>
<td>Usage guide</td>
<td>None</td>
</tr>
</tbody>
</table>

### 4. Browse File

#### Interface Identity
Browse File interface is used by the Load Game function to browse for available files.

#### Resource Provided

<table>
<thead>
<tr>
<th>a. Resource syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Name: AndroidFileBrowser module Parameter Type: Input using mouse Parameter Info: String which represents a file name Exception: Strings with special characters</td>
</tr>
<tr>
<td>b. Resource semantics</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>c. Resource usage restrictions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Locally defined data types</th>
<th>A string matching file name standards</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Bad string with special characters, doesn’t meet file name standards, system errors</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Variability provided</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Quality attribute characteristics</th>
<th>This is limited by the capabilities of the system in terms of processing speed and number of requests processed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Elements Requirements</th>
<th>Files system is checked against the filename string. Strings are first validated for data correctness.</th>
</tr>
</thead>
</table>

| Rational & Design Issues | 1. What data is returned under normal conditions? File with the selected name is loaded into the system  
2. What data is returned if the database is down? Standard system error. No such file is found. |
|--------------------------|-------------------------------------------------------------------------------------------------|

<table>
<thead>
<tr>
<th>Implementation Notes</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Usage guide</th>
<th>None</th>
</tr>
</thead>
</table>

## 5. Login Screen

<table>
<thead>
<tr>
<th>Interface Identity</th>
<th>Login Screen interface is used to login to the web server.</th>
</tr>
</thead>
</table>

| Resource Provided | Program Name: Login module  
Parameter Type: Username and password |
|--------------------|--------------------------------------------------------------------------------|

123
| Parameter Info | username and character protected password  
| Exception: username string with special characters |
| b. Resource semantics | Any username without special characters |
| c. Resource usage restrictions | Special Characters |
| Locally defined data types | A string matching regular expressions |
| Exceptions | Bad username with special characters, system errors |
| Variability provided | None |
| Quality attribute characteristics | This is limited by the capabilities of the system in terms of processing speed and number of requests processed. |
| Elements Requirements | Online Content Databases is checked against the username. Password is validated against pre-set criteria. Usernames are first validated for data correctness. |
| Rational & Design Issues | 1. What data is returned under normal conditions? Welcome, portal screen is displayed.  
2. What data is returned if the database is down? “Cannot login error” |
| Implementation Notes | Set password criteria (check what is set now). Check username against online database. This is outside of the scope of this project and is set for the next release. |
| Usage guide | None |

6. Registration Screen

| Interface Identity | Registration Screen interface used to register access with |
the web server.

### Resource Provided

d. **Resource syntax**  
   Program Name: OnlineRegistration module  
   Parameter Type: Strings  
   Parameter Info: username, password, email, and name  
   Exception: String with special characters for email and password

e. **Resource semantics**  
   Any password and email string without special characters

f. **Resource usage restrictions**  
   Special Characters for email and password

### Locally defined data types

A string matching regular expressions

### Exceptions

Bad email and password with special characters, system errors

### Variability provided

None

### Quality attribute characteristics

This is limited by the capabilities of the system in terms of processing speed and number of requests processed.

### Elements Requirements

Online Content Databases is checked against the username and email. All of the input strings are first validated for data correctness.

### Rational & Design Issues

1. What data is returned under normal conditions?  
   Registration successful message is returned with a link to the login page.
2. What data is returned if the database is down?  
   Standard error page is returned stating that request has been denied and a reason stated

### Implementation Notes

Implementation enhancement.

### Usage guide

None
7. Welcome Screen

<table>
<thead>
<tr>
<th>Interface Identity</th>
<th>Welcome Screen interface used to display available options after login.</th>
</tr>
</thead>
</table>

**Resource Provided**

<table>
<thead>
<tr>
<th>a. Resource syntax</th>
<th>Program Name: Login module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Type: Buttons with input selected using mouse like device.</td>
<td>Parameter Info: Options presented after successful login</td>
</tr>
<tr>
<td>Exception: None</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>b. Resource semantics</th>
<th>Only buttons represented using GUI within the welcome screen.</th>
</tr>
</thead>
<tbody>
<tr>
<td>c. Resource usage restrictions</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Locally defined data types</th>
<th>A button represented by a set of local GUI libraries.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Variability provided</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Quality attribute characteristics</th>
<th>This is limited by the capabilities of the system in terms of processing speed and number of requests processed.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Elements Requirements</th>
<th>Every button needs to match the elements within the module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Rational &amp; Design Issues</th>
<th>1. What data is returned under normal conditions? Interface (screen) of the option selected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. What data is returned if the database is down? Standard system error. Software is closed or restarted.</td>
</tr>
</tbody>
</table>
### 8. View Game Selection Screen

<table>
<thead>
<tr>
<th>Interface Identity</th>
<th>View Game Selection Screen interface used to display available games to view from online portal.</th>
</tr>
</thead>
</table>

#### Resource Provided

**a. Resource syntax**
- Program Name: ViewGame module
- Parameter Type: Input using mouse
- Parameter Info: Buttons through the GUI
- Exception: None

**b. Resource semantics**
- Only buttons represented using GUI within the welcome screen.

**c. Resource usage restrictions**
- None

**Locally defined data types**
- A button represented by a set of local GUI libraries.

**Exceptions**
- None

**Variability provided**
- None

**Quality attribute characteristics**
- This is limited by the capabilities of the system in terms of processing speed and number of requests processed.

**Elements Requirements**
- Every button needs to match the elements within the module

**Rational & Design Issues**
- 1. What data is returned under normal conditions?
  - Game ID of a specific game archived only or connected to a web server
2. What data is returned if the database is down?  
Standard error page is returned stating that connection to the web server or a specific game is not found.

<table>
<thead>
<tr>
<th>Implementation Notes</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage guide</td>
<td>None</td>
</tr>
</tbody>
</table>

## 9. Game View Screen

<table>
<thead>
<tr>
<th>Interface Identity</th>
<th>Game View Screen interface used to view previously selected online game.</th>
</tr>
</thead>
</table>

### Resource Provided

- **a. Resource syntax**  
  Program Name: ViewGame module  
  Parameter Type: Input using a mouse  
  Parameter Info: “Next Move” button  
  Exception: None

- **b. Resource semantics**  
  A button that plays the next move in the queue of moves

- **c. Resource usage restrictions**  
  None

### Locally defined data types

- Button defined by a GUI

### Exceptions

- None

### Variability provided

- None

### Quality attribute characteristics

- This is limited by the capabilities of the system in terms of processing speed and number of requests processed.

### Elements Requirements

- Online Content Databases is checked against the game id and game information is retrieved and stored in the array.
One the “move next” button is clicked every new move is retrieved from the array.

| Rational & Design Issues | 1. What data is returned under normal conditions? Move is generated on the board.  
2. What data is returned if the database is down? Standard error page is returned. System is shut down or restarted. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation Notes</td>
<td>None</td>
</tr>
<tr>
<td>Usage guide</td>
<td>None</td>
</tr>
</tbody>
</table>

10. Join Game Screen

**Interface Identity**

Join Game Screen interface used to display available online games to join.

**Resource Provided**

a. **Resource syntax**  
Program Name: JoinGame module  
Parameter Type: Input using a mouse  
Parameter Info: checkbox accompanied by a name of the user who started the game in a link like format  
Exception: None

b. **Resource semantics**  
A checkbox that represents a game to join

c. **Resource usage restrictions**  
Special Characters

**Locally defined data types**

checkbox defined by a GUI accompanied by a link

**Exceptions**

None

**Variability provided**

None

**Quality attribute characteristics**

This is limited by the capabilities of the system in terms of processing speed and number of requests processed.
### Elements Requirements
Online Content Databases is checked against the requested id to establish connection between the owner of the game and the client system. New moves are first validated for data correctness and then posted to the web server.

### Rational & Design Issues
1. What data is returned under normal conditions?
   Chess board is displayed and the online game play mode is activated.
2. What data is returned if the database is down?
   Standard error page is returned stating that request has been denied.

### Implementation Notes
None

### Usage guide
None

---

### 11. Logout Screen

#### Interface Identity
Logout Screen interface used to displays a message after user is logged out.

#### Resource Provided

<table>
<thead>
<tr>
<th>a. Resource syntax</th>
<th>Program Name: Login module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Type: String</td>
<td>Parameter Info: Plain text</td>
</tr>
<tr>
<td>Exception: None</td>
<td></td>
</tr>
</tbody>
</table>

| b. Resource semantics               | Predefined string                           |
| c. Resource usage restrictions      | None                                        |

<p>| Locally defined data types          | A predefined string of text                 |
| Exceptions                          | None                                        |</p>
<table>
<thead>
<tr>
<th>Variability provided</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality attribute characteristics</td>
<td>No processing is required. This system is required to be functioning at the time the interface is displayed.</td>
</tr>
<tr>
<td>Elements Requirements</td>
<td>System is logged out and session is expired.</td>
</tr>
</tbody>
</table>
| Rational & Design Issues | 1. What data is returned under normal conditions? A predefined text massage stating that user is successfully logged out. 
2. What data is returned if the database is down? Standard error page is returned. System is shut down or restarted |
| Implementation Notes | None |
| Usage guide | Simple screen, no user input is required. |

### 12. Start Game Screen

**Interface Identity**
Start Game Screen interface used to display a form that defines a game to be started.

**Resource Provided**

| a. Resource syntax | Program Name: StartGame module  
Parameter Type: Strings  
Parameter Info: game name, color, and other fields  
Exception: String with special characters for name |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b. Resource semantics</td>
<td>Any keyword without special characters</td>
</tr>
<tr>
<td>c. Resource usage restrictions</td>
<td>Special Characters</td>
</tr>
</tbody>
</table>

**Locally defined data types**
A string matching regular expressions

**Exceptions**
Bad game name with special characters, system errors
Variability provided
None

Quality attribute characteristics
This is limited by the capabilities of the online database and system in terms of processing speed and number of requests processed.

Elements Requirements
Online Content Databases is modified with the game information for the new game. Data is first checked internally.

Rational & Design Issues
1. What data is returned under normal conditions?
   Game board is displayed and the user await for an online player to join the game
2. What data is returned if the database is down?
   Standard error stating that the game could not be started due to a particular issue (for example: internet connection)

Implementation Notes
None

Usage guide
None

13. Settings Screen

Interface Identity
Settings Screen interface used to display game level and language.

Resource Provided

a. Resource syntax
   Program Name: BoardControl module
   Parameter Type: String
   Parameter Info: Game level and other preferences, all input fields
   Exception: Strings with special characters
<table>
<thead>
<tr>
<th><strong>b. Resource semantics</strong></th>
<th>Any keyword without special characters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>c. Resource usage restrictions</strong></td>
<td>Special Characters</td>
</tr>
<tr>
<td><strong>Locally defined data types</strong></td>
<td>A string matching regular expressions</td>
</tr>
<tr>
<td><strong>Exceptions</strong></td>
<td>Bad input term with special characters, system errors. This is for internal game only</td>
</tr>
<tr>
<td><strong>Variability provided</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Quality attribute characteristics</strong></td>
<td>This is limited by the capabilities of the system in terms of processing speed and number of requests processed.</td>
</tr>
<tr>
<td><strong>Elements Requirements</strong></td>
<td>Input terms are first validated for data correctness.</td>
</tr>
</tbody>
</table>
| **Rational & Design Issues** | 1. What data is returned under normal conditions? The game will be reset with the new settings/
| | 2. What data is returned if the database is down? Standard system error. System is either restarted or shut down. |
| **Implementation Notes** | None |
| **Usage guide** | None |

14. About Screen

**Interface Identity** | About Screen interface used to display game credits and information about the game |

**Resource Provided**

<p>| <strong>a. Resource syntax</strong> | Program Name: Menu module Parameter Type: String Parameter Info: Plain text Exception: None |</p>
<table>
<thead>
<tr>
<th></th>
<th>Predefined text</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>b. Resource semantics</td>
<td>Predefined text</td>
<td>None</td>
</tr>
<tr>
<td>c. Resource usage restrictions</td>
<td>None</td>
<td>A predefined string of text</td>
</tr>
<tr>
<td>Locally defined data types</td>
<td>A predefined string of text</td>
<td></td>
</tr>
<tr>
<td>Exceptions</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Variability provided</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Quality attribute characteristics</td>
<td>No processing is required. This system is required to be functioning at the time the interface is displayed.</td>
<td></td>
</tr>
<tr>
<td>Elements Requirements</td>
<td>Predefined text</td>
<td></td>
</tr>
<tr>
<td>Rational &amp; Design Issues</td>
<td>1. What data is returned under normal conditions?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>System output predefined information about the system and the authors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. What data is returned if the database is down?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Standard system error. System is either restarted or shut down.</td>
<td></td>
</tr>
<tr>
<td>Implementation Notes</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Usage guide</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Parameter Info: Button represented by a GUI</td>
<td>Exception: any keywords</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>b. Resource semantics</td>
<td>Buttons defined by a GUI</td>
<td></td>
</tr>
<tr>
<td>c. Resource usage restrictions</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Locally defined data types</td>
<td>Button</td>
<td></td>
</tr>
<tr>
<td>Exceptions</td>
<td>This is for internal game only.</td>
<td></td>
</tr>
<tr>
<td>Variability provided</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Quality attribute characteristics</td>
<td>This is limited by the capabilities of the system in terms of processing speed and number of requests processed.</td>
<td></td>
</tr>
<tr>
<td>Elements Requirements</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
| Rational & Design Issues                  | 1. What data is returned under normal conditions? Board is re-shuffled according to user instructions  
2. What data is returned if the database is down? Standard system error. System is either restarted or shut down. |
| Implementation Notes                      | None                    |
| Usage guide                               | None                    |