Software Requirements Specification

for

Hound Army Checkers

Version 1.0 approved

Prepared by Samantha Coyle, Spencer Albert, Taylor Moralez, and Rowan Stone

CS3398 Fall 2018: Software Engineering, Section 001

September 12, 2018

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this document is to provide a detailed description of the requirements for the Hound Army Checker Board game. This document will cover the details on the functionality, design specifications, features, and requirements for the game. Hound Army shall provide the user a functional, high-quality checkers game. This document is version 1.0. The SRS scope will include a full description of the requirements and functional deliverables that the Hound Army shall produce.

## Document Conventions

The naming standards included within this document are as stated below. The acronym is on the left and the full-description of the acronym is following the comma. Priorities for higher-level requirements are assumed to be inherited by detailed requirements.

## Intended Audience and Reading Suggestions

This document is intended for developers so that they may follow the requirements listed within the SRS as they work on developing the product. It is intended for project managers so that they may ensure that the requirements are met. The marketing staff will utilize this document to boast its features, functionality, and efficiency to the users of the checker board game. Testers will also use this document, following it in sequential order, to ensure that all requirements are met, and product has no issues. The rest of the SRS will contain an overall description of the Hound Army Checker Board game, list the external interface requirements, system features, nonfunctional requirements, and other requirements. Additionally, there is an appendix section at the end of the requirements section to disclose a glossary and analysis models. It is organized with a general description, followed by a more detailed list of requirements. It is suggested to read this document in sequential order starting at the beginning, until the end of the document is reached so that the reader may get a good grasp of the goals, overview, and features of the game. It is recommended that all readers start at the beginning to get an understanding of the game and proceed from there. Developers, project managers, and testers should read the entire document to fully understand the product. The marketing staff does not need to understand the entirety of the requirements section but may want to read through it to ensure full comprehension of the game.

## Product Scope

Hound Army shall provide a fair checker board game utilizing JavaScript. The benefits of this product include a traditional checker board game for the entertainment and enjoyment of the user. Benefits include improving ones’ checker board game skills.

## References

“Checkers: American.” *ItsYourTurn.com - Help Page*, It's Your Turn, Inc, 2017,

[www.itsyourturn.com/t\_helptopic2030.html](http://www.itsyourturn.com/t_helptopic2030.html).

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

# Overall Description (Rowan)

## Product Perspective

Hound Army Checkers is a stand-alone application that will serve as a digital version of the American iteration of the classic board game, Checkers. Checkers is a turn-based, strategy game that requires two players. The American version of the game is played on an 8 x 8 checkerboard, with the starting number of 12 uniformly shaped game pieces per player, termed “men”. One player will use light pieces, and the other will use dark pieces. Player pieces are only allowed to occupy the dark squares on the checkerboard. The player with the dark pieces will move first. The objective of Checkers is for the player to capture all their opponent’s pieces, or to otherwise prevent them from being able to make a legal move. Legal moves consist of either “simple “moves, where the player moves their piece to an adjacent, diagonal space going towards the opponent’s side of the board; “jumps”, where the player will “jump-over” a single opponent’s piece that is on an adjacent, diagonal space to occupy the next empty space in the same direction; and “multiple jumps”, where the player performs at least two “jumps” in a row when it is valid for the player to jump over another piece after the first jump has been made, either in the same diagonal direction or a different one. When a player’s “man” reaches the row closest to their opponent on the board, known as “king’s row”, the piece is upgraded to the status of “king”. Kings can move in any diagonal direction, forward or backward.

## Product Functions

· HAC shall allow the user to reference the player manual during the game.

· HAC shall allow the user to select a difficulty mode before starting the game.

· HAC shall allow the user to select a game mode before starting the game.

· HAC shall allow the user to select between playing as the light or dark side.

· HAC shall allow the user to select an avatar picture at the start of the game.

· HAC shall allow the user to toggle music/sound on or off at will.

· HAC shall notify the user when it is their turn to move.

· HAC shall only allow each player to move their own pieces during their turn.

· HAC shall allow the user to move a piece, to the selected space, as long as the move is valid.

· HAC shall not move a piece for the user if the user’s chosen move for that piece is invalid.

· HAC shall provide an accurate and up-to-date score on the scoreboard.

· HAC shall remove captured pieces from the board.

· HAC shall king a player’s piece as soon as their piece reaches king’s row on their opponent’s side.

· HAC shall notify the user when the game has ended and then stop gameplay.

· HAC shall notify the user if they have won or lost the game.

## User Classes and Characteristics

There will be two user classes:

· Novice players will play the game under the ‘Easy’ difficulty setting to prevent the user from feeling intimidated by the game when starting out.

· Experienced players will play the game under the ‘Normal’ difficulty to have a stimulating challenge for a pleasant user experience.

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

## Operating Environment

The software will operate on a modern web application environment, which will be accessible on [compatible web browser name(s) here].

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

## Design and Implementation Constraints

HAC will only be available in English. The software is expected to be completed by October 22, 2018. HAC will be developed using JavaScript. HAC will not save or ask for any personal user information, therefore there will be minimal security considerations.

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

A short user manual will be included as a text document to provide users with an overview of how to operate Hound Army Checkers. A Checkers How-To-Play reference guide will be included to provide the general rules for the game.

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

A third-party AI module, [insert name here], will be used for the computer opponent that will play against the user. [List additional dependencies here]. It will be assumed that all music and/or sound effects used for the game will be under a creative commons use license. Graphics to be sourced will also need to be under a creative commons use license, or created specifically for use in HAC.

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

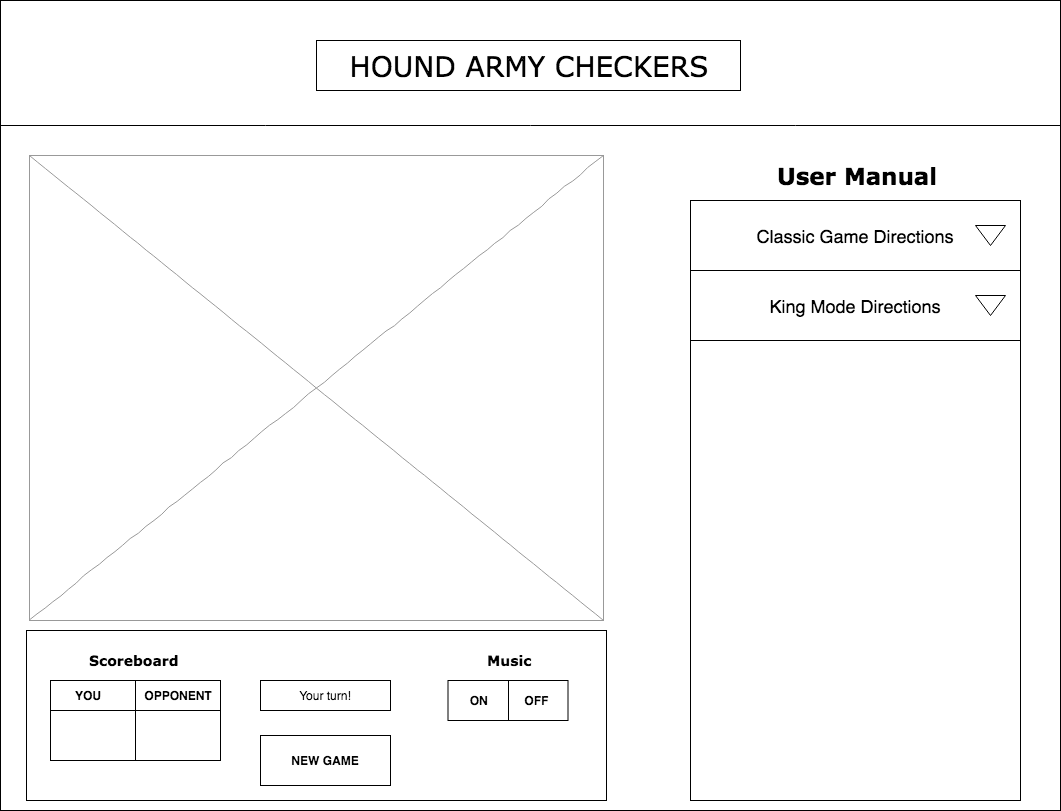
# External Interface Requirements (Taylor)

## User Interfaces

The user will interact with the game through a single screen, on which the user will find the game board towards the top-left of the screen. This is where the user will interact with their pieces during the game, as well as configure options for starting a new game. Below the game board area towards the bottom-left of the screen will lie a variety of information such as the current score of the game being played, and options such as a toggle for the music, as well as a ‘New Game’ button, which will restart the game, and allow the user to choose which game mode they wish to play. On the right-hand side of the screen will be a user manual menu with information about the game, including the rules for checkers, as well as the rules for King mode.

During gameplay, the user will move their pieces by first clicking on one of their pieces, then clicking on the space that the user would like to move their piece to. In the case event that a player has multiple jumps, (PLEASE FILL IN USE CASE).

Below is an approximate example of the layout of the program.



<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

There will be no additional hardware requirements besides a mouse and keyboard that is compatible with the user’s computer and operating system environment.

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

HAC will be a standalone JavaScript application, requiring only a compatible web browser to run. (Please update section with any libraries added to the program).

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

HAC will be a standalone application, with no communication with any other programs or in-between different users.

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## Game Screen (Sam)

4.1.1 Description and Priority

The overall user interface shall include the checkers board game, a text area depicting whose turn it is in the game, and a tally of the scoreboard for each player. Also, under the board will include a toggle to turn on/off music for during the game, as well as a button to click to start the game over. The interface shall have a menu option for users to select optional drop-down menus to read more in-depth information on directions and features of the game. This feature is described in further detail in section 4.1.\_\_\_\_\_. These aspects of the user interface are of high priority as they benefit the user in understanding how to play the game and exploit the different features of the game. The benefit factor is 9 as this is essential to playing the game, it is penalty factor is 0, and the risk factor is also 0.

4.1.2 Stimulus/Response Sequences

* + If the user opens the web application, then the checker board game, additional feature toggle buttons, scoreboard, text field for whose turn it is, and the user manual will appear.
  + If the user opens the web application, then an alert for the user to select the game mode shall appear. The game screen shall not let the game start without a game mode initially chosen.
  + When a player wins the checker game, an alert shall be displayed depicting the game winner.
  + If the start over game button is clicked, the checker game board is cleared and refreshed to its initial condition. The user will then be prompted for a new game mode.

4.1.3 Functional Requirements

REQ-1: If the user’s laptop/PC has visual capabilities, then the game screen shall load up properly without error.

REQ-2: The game screen shall display the checkers game board, user manual, text area to depict whose turn it is, scoreboard, music on/off toggle, and a start over button. These features will be functional.

REQ-3: The game screen user manual shall have drop down features to display more information to the user if selected.

REQ-4: The game screen shall have an alert to the user upon application open to get the game mode selection from the user.

REQ-5: The game screen shall not let the user start a game without a game mode chosen.

REQ-6: The game board itself shall be interactive and allow the user to move their game pieces to valid spaces.

REQ-7: The scoreboard shall maintain an accurate account of the number of pieces that each player has jumped over.

REQ-8: The music on/off toggle shall have the capability to play music pending the on/off mode.

REQ-9: The start over button shall start the game over if clicked. It will re-alert the user to choose the new game mode.

REQ-10: The game screen shall alert the user when the game is over who won overall.

REQ-11: More error conditions/invalid inputs TBD

## Game Engine (Group)

4.2.1Description and Priority

HAC will feature a game engine that is responsible for initializing the board state at the beginning of each game, determining the validity of any moves made by the user, making the moves for the AI opponent, and determining when a game has ended, and which player has won. This is a high priority feature, as it is responsible for the basic functionality of the game.

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.2.2Stimulus/Response Sequences

* + If the user opens the application or selects to start a new game, then HAC shall display the checker game board with the proper checker game pieces so that the users may start the game. HAC shall send an alert to the user to select a game mode—either regular or king mode and select the game piece color they would like to play as.
  + If the user selects to move a game piece, then HAC shall allow the user to click where they want to place their piece. If the user decides to move the game piece to an invalid space, then HAC shall notify the user of the invalid move.
  + Once the user has completed their turn with a valid move, HAC shall choose a valid move for the AI opponent.
  + If the user gets a piece to the opponents end of the board, HAC shall make that piece a king
  + When the game is over, and one player has no more game pieces to play or if they can’t make a move, then HAC shall notify the user who has won the game.
  + If the user can jump over multiple opponent pieces, then HAC shall enforce the user to jump as far as possible until there are no more available jumps for that piece.

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.2.3Functional Requirements

REQ-1: If the user clicks on a game piece that is their own, then the game board shall allow the user to move the game piece to a valid space.

REQ-2: If the user’s turn is up, then HAC shall have the AI opponent complete their turn.

REQ-3: If the game is over, then HAC shall notify the user who has won the game.

REQ-4: If the user tries to move the game piece to a non-valid space, then HAC shall notify the user that the move is invalid and shall not move any pieces.

REQ-5: If the user tries to move a game piece that is not their color, then HAC shall notify the user that they are the other color game pieces and no pieces shall be moved.

REQ-6: If the user starts a new game, then HAC will initialize the board with all the pieces (regular or kings, depending on which game mode the user has chosen), and HAC will reset the score of both the player and AI opponent.

REQ-7: If the user selects a game piece, then HAC shall allow the user to click where they want to place the piece, if it is a valid move.

REQ-8: HAC shall keep track of the number of game pieces that a player has left to determine if the game is won/lost if one player has no more pieces, or if they can no longer make a move.

REQ-9: When a piece gets to the end of the board, HAC shall make that piece a king.

REQ-10: HAC shall keep track of the number of available jumps for the game piece that the user selects to ensure that the user makes a valid move and moves their chosen piece as far as possible.

REQ-11: HAC shall keep track of a game piece status (king or regular), so that pieces that are already kings cannot become kings again.

REQ-#: Other error conditions TBD

## System Sound (Rowan)

4.3.1Description and Priority (set below to paragraph format for consistency)

Priority: Low

Benefit: Provide a more enjoyable gameplay experience for the user as well as additional gameplay feedback that users will find satisfying and more immersive.

The game shall provide the player with the option to toggle music on or off during the game. The game shall also provide audio feedback when a piece has been moved on the board.

4.3.2Stimulus/Response Sequences

* If the user clicks the sound button once while HAC is running, the game shall play sounds when pieces are moved.
* If the user clicks the sound button a second time while HAC is running, the game shall stop playing sound effects when pieces are moved.
* If the user clicks the music button once while HAC is running, the game shall play music during the game.
* If the user clicks the music button a second time while HAC is running, the game shall stop playing music.

4.3.3Functional Requirements (Revise)

REQ-1: If the user’s computer has a functional sound card, then the sound and music toggle options shall be available for the user to use.

REQ-2: Sound effects and music will need to be acquired in order to enable this feature.

## In-Game Manual (Sam)

4.4.1Description and Priority

The in-game user manual shall be displayed on the right side of the game screen. It shall display to the user a drop-down menu option of different features and rules in detail so that they can have a better understanding of some of game and its additional features. This is of medium priority as it is not essential to have, but it will aid in the user experience and understanding of the application. It is given a score of 7 for benefit to the user, 0 for penalty as it does not harm the experience of the user, and it will be a risk factor of 0.

4.4.2Stimulus/Response Sequences

* + If a drop down on the in-game manual is clicked, more information on chosen subject shall appear.
  + If the drop-down arrow on the in-game manual is clicked a second time, the additional information shall hide again until re-clicked.
  + If multiple drop-downs are selected to open, then they shall all be available to open.

4.4.3Functional Requirements

REQ-1: If the game screen is opened, all drop-downs in the in-game manual shall be kept closed until opened by user.

REQ-2: If the game screen in-game manual drop-down button is clicked, the appropriate information shall be displayed to the user.

REQ-#: More error conditions/invalid inputs TBD.

## Scoreboard (Spencer)

4.5.1Description and Priority

The game screen shall display a scoreboard underneath the gameboard, with the display showing the user how many pieces they have taken from the opponent, and how many pieces the opponent has taken from the user. This feature is of medium priority.

4.5.2Stimulus/Response Sequences

* + If a user starts a new game, then scoreboard will be reset to 0.
  + If a user takes an opponent's piece, or if a user has a piece taken, then scoreboard will be updated.
  + If a game ends, then the scoreboard will be reset to 0.

4.5.3Functional Requirements

REQ-1: When a checker piece is taken off the board, the scoreboard shall be updated correctly to whoever took the piece (user, or opponent) before the next turn takes place.

REQ-2: When a game ends, or a new game begins the scoreboard shall zero out.

## King Mode (Taylor)

4.6.1Description and Priority

The game shall allow the user to start a new game where all the player’s and opponent’s pieces start of as kings. The game mode will then continue as normal, either until the game ends or the player starts a new game. This is a medium priority feature, as it adds significant additional features to the game, but is not integral to the core functionality of the program.

4.6.2Stimulus/Response Sequences

* If the user starts a new game (either on startup of the game, when a game is finished, or when the user clicks the ‘New Game’ button), then the user will be prompted with a dialogue box that will ask whether to start a regular game or a King Mode game.
* The game mode will not be altered until the user starts a new game.
* Once the user selects a new King Mode game, the game board will reset to the starting position, the score will be reset, and all the pieces on the board will be kings.
* The game will continue as normal, until either the game is finished, or the user selects the ‘New Game’ button, in which the user will be prompted once again whether they wish to start a new regular checkers game, or a new King Mode game.

4.6.3Functional Requirements

REQ-1: The game screen will have an alert that asks the user to select either a regular checkers game, or a King Mode game.

REQ-2: Upon selection of a new King Mode game, the game board will be populated with all the pieces as kings, and the score reset.

REQ-3: Error conditions TBD

## Avatar Picture Selection (Rowan)

4.7.1Description and Priority

Priority: Low

Benefits: This feature will allow for a more customizable user experience for the player.

The game shall provide the player with the option to choose out of a predetermined selection of image icons to customize their gameplay experience.

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.7.2Stimulus/Response Sequences

o If the user chooses to select an avatar, then the avatar selection menu shall open. The menu shall show a list of images available to choose from.

o If the user clicks on an image in the list, the menu shall close, and the selected image will be set as the user’s avatar picture.

o If the user does not select an image, no avatar will be set.

o If the user clicks outside of the avatar selection menu, then the menu shall close, and no avatar image shall be set.

4.7.3Functional Requirements (Revise)

REQ-1: Collection of appropriately sized images that are in a browser compatible format.

# Other Nonfunctional Requirements (Spencer)

## Performance Requirements

Hound Army Checkers shall not crash or freeze for the user. Beyond that, there have not been any specific performance requirements set. The overall goal of Hound Army Checkers shall be to get the program running smoothly for the user, without any bugs or other user related problems.

## Safety Requirements

There are no safety requirements for this project.

## Security Requirements

The user will not be inputting any sensitive information, and none will be kept by Hound Army Checkers, so there are no security requirements.

## Software Quality Attributes

Hound Army Checkers shall run well, without bugs, and give the user an experience in comparison to other online checker games with similar features.

## Business Rules

There are no business rules for Hound Army Checkers.

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

|  |  |
| --- | --- |
| HAC | Hound Army Checkers |
| AI | Artificial Intelligence |
|  |  |

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>