SE 3XA3: Module Guide
Ultimate Tic Tac Toe

Team 3, Ultimate Tic Tac Toe
Kunal Shah - shahk24
Pareek Ravi - ravip2

November 17, 2016

Contents

List of Tables

List of Figures

Abstract

Ultimate Tic Tac Toe is a variation on the classic game of Tic Tac Toe. It is simply multiple games of Tic Tac Toe running simultaneously to make a classic game that often ends in a draw have an exciting ending.

The primary purpose of the Module Guide (MG) is to describe, justify, and contextualize the module decomposition of the system. The document also provides a conceptual view of each module and its context in the broader system. Before the MG is written, the Software Requirements Specification (SRS) document should be completed. The SRS describes requirements for the software system that the MG should relate to the implemented design. By completing the SRS beforehand, it is more likely that the selected system design will comply with all compulsory requirements and the chosen module decomposition is appropriately structured.

1 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (**?**). We advocate a decomposition based on the principle of information hiding (**?**). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC (SC not defined) - CM, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules laid out by **?**, as follows:

- System details that are likely to change independently should be the secrets of separate modules.

- Each data structure is used in only one module.

- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

Table 1: **Revision History** Try adding clearpage or something to make this go into correct place - CM

Date	Version	Notes
November 8	0.0	Initial set up
November 11	0.2	introduction
November 11	0.4	updated all sections
November 13	1.0	Rev0 Submission

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (**?**). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.

- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.

- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section 4 specifies the connections between the software requirements and the modules. Section 5 gives a detailed description of the modules. Section 6 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 7 describes the use relation between modules.

2 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 2.1, and unlikely changes are listed in Section 2.2.

2.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The hardware on which the software is running.

AC2: Draw / tie logic handling

AC3: Online muli-player support

AC4: Game user interface layout (responsiveness based on device screen size)

AC5: The language in which the game rules is presented.

2.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input devices (touch and mouse)

UC2: Win and loss logic

UC3: Data structures holding game state information

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: User Interface Module

M2 : Move module

M3: Player turn module

M4: Active region module

M5: Win check module

Is your Win Check Module not satisfying your requirements? Does your active region not satisfy your NF requirements? SD modules are used for non-requirement bases software. Think holding the board in storage or memory, sorting... etc. - CM

4 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

Level 1	Level 2
Hardware-Hiding Module	User Interface Module
Behaviour-Hiding Module	Player turn module Move module
Software Decision Module	Active region module Win check module

Table 2: Module Hierarchy

5 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by **?**. The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. Also indicate if the module will be implemented specifically for the software.

5.1 Hardware Hiding Modules

5.1.1 User Interface Module (M1)

Secrets: The data structure and algorithm used to implement the Ultimate Tic Tac Toe game.

Services: This module provides the interface between the script and User. So, the script can use it to display outputs and to accept inputs from User.

Implemented By: HTML user interface

For above, perhaps talking about the clicking, hovering and displaying of the game. I understand the intent, it just isn't clear - CM

5.2 Behaviour-Hiding Module

5.2.1 Player Turn Module (M3)

Secrets: current player state

Services: Includes programs that provide externally visible player symbol the system as specified in the software requirements specification (SRS) documents. This module

4

indicates the player which square the next player will make their move based on the previous player's move

Implemented By: Player Turn Module

Couldn't understand Services, try rewording and being more clear - CM

5.2.2 Move Module (M2)

Secrets: The format and structure of the input data.

Services: This module is responsible for translating user input and sending to the game logic data structure.

Implemented By: Move Module

Really like this one! Perfect scope and descriptions. Other than implemented by, try saying which app would do it. This makes more sense when considering multi-app programs where separation of work is more essential - CM

5.3 Software Decision Module

5.3.1 Active Region Module (M4)

Secrets: Next Active region

Services: This module is responsible for updating the board to the new active region based on previous player interactions after receiving information from the move module

Implemented By: Active Region Module

5.3.2 Win Check Module (M5)

Secrets: Current game state

Services: This module is responsible for taking care of checking if the player has won the game, and updating the board after receiving information from the move module. This means if player has won or drawn an inner or game board.

Implemented By: Win Check Module

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
FR1	M1
FR2	M1, M2, M4
FR3	M4, M5
FR4	M1, M5
FR5	M5
FR6	M1, M3

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M5
AC3	M3
AC4	M1
AC5	M1

Table 4: Trace Between Anticipated Changes and Modules

7 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. **?** said of two modules A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the each level of the hierarchy offers a testable and usable subset of the system.
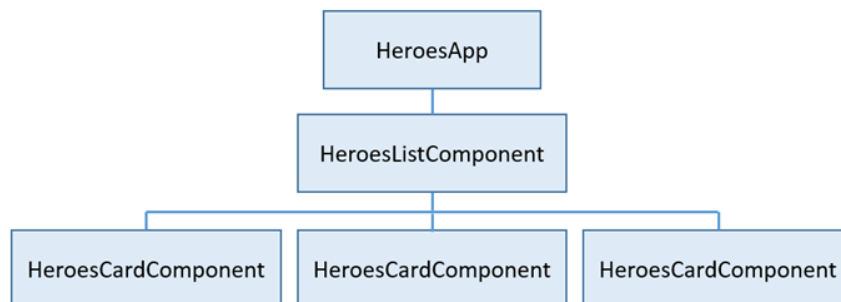
Figure 1: Use hierarchy among modules

PDF wasn't included in any commits. I don't really like it though. I don't see how the Win Check Model uses the UI. Perhaps more detail on the structure or rework it. I also threw in a sample module hierarchy so that the TeX compiles - CM