17-654/17-754: Analysis of Software Artifacts
Jonathan Aldrich, Nels Beckman, Kevin Bierhoff
Assignment 9 (Written): DSMs and EDSMs

Due Tuesday, April 3, 10:30am
Turn in a file named <username>-17654-A9.{txt,doc,pdf}, where username is your Andrew id, electronically through Blackboard. It should contain your answers to the tasks in this assignment.

100 points

Assignment Objectives:
- Gain familiarity with creating Dependency Structure Matrices (DSMs).
- Analyze an application design using DSMs.
- Evaluate a change scenario using EDSMs.

Dependency Structure Matrix (DSM) for Othello

The included Excel file, DSM-Program.xls, contains a DSM (DSM "A") for Othello, the example application we used in previous assignments (worksheet "DSM"). It shows dependencies between the Java files comprising the Othello code base. A non-zero entry in column I and row J means that the file in column I depends on the file in row J in some way. Dependencies between files include types that appear in the source code, such a as a field type, but also the receiver of methods that are called in the source. You should not worry about the actual numbers too much.

DSM-Program.xls uses a macro that we downloaded from www.dsmweb.org to partition the Othello DSM into strongly connected clusters (worksheet "Partitioned DSM"). As you can see, only two clusters were identified. As it is common, most—but not all—cells in the upper triangle of the partitioned DSM have the entry "0".

Note: If you enable macros in the provided Excel file then you can enter new DSMs and have it partitioned automatically. However, we will not be using this feature in this assignment.

Analyzing DSMs

Task 1 (20 points): The partitioned DSM as provided deviates from the ideal case in a few places. In particular, there are three non-zero cells in the upper triangle that are outside of identified clusters: BoardGameModel depends on AIThread, OthelloView depends on BoardGameApp and BoardGameWin. Explain where in the code of BoardGameModel and OthelloView these unusual dependencies occur. Try to understand (for yourself; you do not have to put it in writing) why these unusual dependencies inevitably appear in the DSM given the Othello code base.

Task 2 (10 points): The automatically partitioned DSM contains only two clusters. In a paragraph, explain why that is the case. Base you explanations on observations from

DSM A—do **not** argue based on the code. Comment on whether this phenomenon indicates good or bad design.

Task 3 (20 points): Surprisingly, one of the clusters contains an interface, BoardAndRules, three game engine classes, and an Othello-specific class, games.Othello. Arguably, BoardAndRules and games.Othello should not be part of the cluster: Othello-specific classes should be separate from the generic gaming engine and from game interfaces. Based on the provided DSM (do **not** argue based on the code), explain why BoardAndRules and games.Othello became part of this cluster.
Hint: Items are usually clustered together if there are circular dependencies between them. For example, Board and View are a cluster because Board depends on View and View depends on Board.

Identifying missing dependencies

Task 4 (10 points): BoardAndRules is the main interface defining a game in the given code base. If a new game were to be added then this interface would have to be implemented. It "summarizes" the other game interfaces, Board, View, Status, Move, and Rules. However, in the given DSM, BoardAndRules surprisingly does not depend on any of these interfaces. Looking at the code for BoardAndRules, explain this critical omission.
Hint: Remember that DSM A only shows dependencies that are visible in the source code.

Deriving a higher-level DSM

The provided DSM shows dependencies between Othello source files. However, this is often a too low level of granularity for analyzing a DSM. Fortunately, we can aggregate the Othello files into the following five subsystems:

1. AI
   - AIThread
2. Othello
   - All files in package edu.calpoly.csc435.othello
   - edu.calpoly.csc435.games.Othello
3. Game engine
   - BoardGameApp
   - BoardGameModel
   - BoardGameWin
   - StatusView
   - RulesDlg
4. Game interfaces
   - BoardAndRules
   - Board
   - View
   - Status
   - Move
   - Rules

5. Exceptions
   - InvalidOptionsException
   - IncrVals

Task 5 (20 points): Create a DSM ("B") that shows dependencies between the five subsystems as defined above. One way of doing so is aggregating the dependencies shown in the provided DSM A for each of these subsystems. Other ways include looking at the source files and finding dependencies by hand or writing a Crystal3 tree walker analysis. Do not worry about the exact number of dependencies; just make sure you show a dependency if and only if there is one according to DSM A.

Task 6 (10 points): How do the problems with DSM A discussed in tasks 1 and 3 appear in your DSM B?

A change task

As discussed before, the code base is designed to be extensible to new games. Thus, it should be easy to add new games. Othello is just the only game currently implemented. EDSMs are a good tool to track how changes affect the existing design.

Task 7 (10 points): Suppose that a new game, Checkers, were to be added to the provided code base. Create an EDSM ("C") that is derived from DSM B and includes an environment parameter. The environment parameter should be, "adapting the game to play checkers instead of Othello". (Recall that environment parameters appear at the top/left of an EDSM.) Show which of the existing subsystems would be affected by this change scenario by putting Xs in the column for the environmental parameter.
Hint: You may assume that a Checkers implementation would be very similar to the given Othello implementation.