A Grand, Unified Project: Doane SuDoKu

Mark M. Meysenburg
Information Science and Technology Department
Doane College
Crete, NE 68333
mark.meysenburg@doane.edu

Abstract

In the Spring 2012 semester, several Doane College Information Science and Technology (IST) classes are working together on a unified semester project. The project involves a multi-tiered SuDoKu game system, with three major components: A "factory" computer that creates new games and determines their difficulties; a central database server to store games and player score information; and desktop and Android client applications for playing the games. The factory and client applications use a Model / View / Controller architecture. The project will result in a prototype system playable on the Doane network. Further work, in summer research projects or other classes, will migrate the database and hosting to a more robust environment, and create a Web site devoted to showing high scores, leader boards, and so on. Eventually, we will make the applications available to the public via channels such as the Android Marketplace. This paper describes the preliminary results of our ongoing project, including specifics regarding system architecture, software choices, our version control system, scheduling issues, and future plans for taking the Doane SuDoKu project "live."

1 Introduction

In the Spring 2012 semester, several Doane College Information Science and Technology (IST) classes are working together on a unified semester project. The project involves a multi-tiered SuDoKu game system, with three major components: A "factory" computer that creates new games and determines their difficulties; a central database server to store games and player score information; and desktop and Android client applications for playing the games. The factory and client applications use a Model / View / Controller architecture. The project will result in a prototype system playable on the Doane network. Further work, in summer research projects or other classes, will migrate the database and hosting to a more robust environment, and create a Web site devoted to showing high scores, leader boards, and so on. Eventually, we will make the applications available to the public via channels such as the Android Marketplace.

The three classes involved in the SuDoKu project are the author's Spring 2012 courses: IST 146 (Programming and Problem Solving II), IST 314 (Design and Analysis of Algorithms), and IST 356 (Software Engineering). Each class has unique responsibilities in the project. Specifically, IST 356 students are responsible for the detailed specifications of the client and factory applications, and for the design and coding of the controller classes for the applications. IST 314 students are responsible for the algorithms and code to create new games and determine their complexity. IST 146 students are responsible for constructing the user interfaces for the client applications and the factory, based on the specifications and user stories created by the IST 356 class. The professor is responsible for the model classes, and for the prototypes Derby databases and encapsulating classes.

The project applies to specific learning outcomes in each course, and also reinforces overall concepts. Our students learn user interface creation in IST 146; the project allows them to work on "real world" applications driven by external customers, and to learn both desktop and Android interface creation. IST 314 students obviously study algorithms; the project allows them to apply the techniques they learn to a novel problem. IST 356 students learn about software engineering topics, such as architecture and design patterns, user requirements, and specifications, through a project that will someday be available to the public at large. Through the project, students in these courses are exposed to a larger system than they have seen to this point in their college careers. They practice concepts such as pair programming and test-driven development. And, they practice developing software in a team environment, using a version control system.

The sections below describe the preliminary results of our ongoing project, including specifics regarding system architecture, software choices, our version control system, scheduling issues, and future plans for taking the Doane SuDoKu project "live."

2 System architecture

Doane SuDoKu is a multi-tiered application, where the creation of games, storage of games, and playing of games are handled by distinct systems. This separation of functionality provides a logical way to partition the project work over several courses. As shown in Figure 1, the components of the systems are:

1. Desktop and Android client applications for playing SuDoKu games;
2. A central database server for storing games, user information, and scores; and
3. A "factory" application and local database that creates new games, and determines their difficulties.

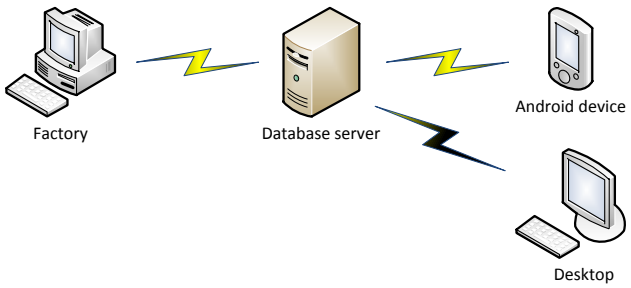

Figure 1: High-level architecture of the Doane SuDoKu architecture.

2.1 Client applications

For the eventual end-user, the *raison d'être* for the Doane SuDoKu project is to play SuDoKu games, and so the client applications are a vital part of the system. We are designing two client applications: one for Android devices, and another for desktops (PC / Mac / Linux). The Android version has higher priority than the desktop version; if we need to perform project triage due to time constraints, we may drop the desktop client for the time being. As of this writing, we are in the design phase for the client interfaces; they will be constructed later in the semester.

A preliminary sketch of the Android user interface is shown in Figure 2. In the sketch, the SuDoKu grid consumes most of the screen area. The game difficulty and time elapsed are shown to the left of the grid. Buttons for entering numbers into the grid appear below the grid; these are desirable since Android devices typically don't have keyboards. Buttons for undo, pause, answer / notes mode, and erase appear beneath the entry buttons. Other functionality, such as starting new games, loading / saving games, and hint mode, will be accessed through the 'Droid menu.

A preliminary sketch of the desktop UI is shown in Figure 3. The UI will contain the same functionality as the Android version, with the exception of buttons for entering number into

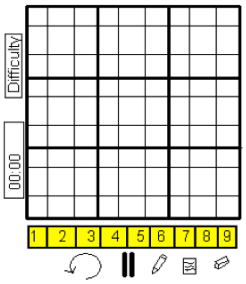

Figure 2: Preliminary sketch of Android UI.

the grid. In the desktop application, numbers will be entered into the grid by highlighting a cell with the mouse and pressing one of the keys on the keyboard.

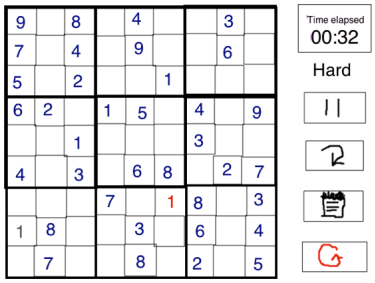

Figure 3: Preliminary sketch of desktop UI.

2.1.1 Client architecture

The client applications are being designed using the Model / View / Controller architecture. Apart from being good software engineering practice, this design allows different pieces of the applications to be developed in parallel by different Doane IST courses (see Section 3, below). The classes for the sections of the architecture are separated into model, view, and controller packages.

The model package consists primarily of a Grid class, originally written by the author several years ago for another prototype SuDoKu application. The Grid class encapsulates ideas like setting and clearing numbers, determining candidates and constraints for each of the cells, determining if the grid is solved or locked, and so on. Other representative model classes are a Game class, encapsulating the information stored about a single game in the database, and an ElapsedTime class, encapsulating the amount of time taken to solve a puzzle so far. Each game is given a unique identification number, and a difficulty level (very easy, easy, medium, hard, very hard, or expert).

The controller package will contain one key class, cleverly named Controller. The singleton Controller object will encapsulate a Grid, and communicate with a small local database as well as the central database server. The local database holds a certain number of games

of each of the difficulty levels. When the user wants to play a new game, the game is fetched from the local database. Periodically, the Controller object communicates with the central database server to fetch new games, which are copied into the local database. During game play, the Controller object handles events generated by the user interface, reflects these events on the Grid object representing the game, and updates the user interface via the observer design pattern.

The view package will contain the UI classes for the Android and desktop applications. The UI classes will not contain any of the game logic; rather, they will communicate with the controller.

2.2 Database server

The central database server serves as the public repository for games, and also stores high score information for each game and user. "High scores" are based on the elapsed time taken to play each game. Initially the server is being implemented as an Apache Derby database [3], on one of our departmental servers. As such, the server database is only available on our campus, or off campus via a VPN connection to the Doane network. Later, we migrate the database server, and perhaps the database software to a more robust, secure, and public option. Eventually, a Web client interface will access the database server, allowing high scores and leader boards to be viewed online.

2.3 Factory application

The factory application is hosted on a separate computer, whose job it is to make new SuDoKu games 24 hours a day. The application creates new games using a random process, and then classifies each game according to difficulty. The factory stores games in a local database, and periodically uploads new games to the central database server. The local database will contain all of the games ever made, thus acting as a backup for the database on the central server. The factory will be "throttled," so that it does not create too many new games. Once some threshold value of unused games is achieved on the central database, the factory will only make enough games each day to maintain that threshold value.

The factory will judge the difficulty of newly created games by measuring which strategies are required to solve a puzzle. The strategies we are attempting to implement are described in detail on a Web site devoted to all kinds of number puzzles [11]. For example, if a SuDoKu game can be solved using only the first two strategies, it is deemed to be a "very easy" puzzle; if solving the puzzle requires 21 or more of the strategies, it is an "expert" game.

3 Class responsibilities

From the professor's point of view, the *raison d'être* for the project is tied to the goals of the author's three Spring 2012 classes: IST 146 (Programming and Problem Solving II), IST 314 (Design and Analysis of Algorithms), and IST 356 (Software Engineering). There are seven students in IST 146, seven in IST 314, and eight students in IST 356 this semester. Two students are in more than one class: one is in 314 and 356, while another is in 146 and 356.

All Doane IST courses incorporate a semester project, generally completed in teams. Sample projects are a genetic algorithm system for IST 146 and researching NP-Complete problems in IST 314. A previous version of a self-contained SuDoKu desktop application was used as the basis for IST 356 several times. IST 356 students were given a more-or-less complete version of the application, and then tasked with tracking down and fixing bugs introduced by the professor, and extending the game with new features.

This year, all three classes are working together on the same semester project, the scaled-up, multi-tiered SuDoKu application described in the preceding sections. This allows us to work on a larger project, with students in each class having responsibilities for distinct parts of the application. These responsibilities are described below.

3.1 IST 146 (CS 2)

Along with inheritance / polymorphism, exception handling, recursion, and data structures, one of the primary topics of the IST 146 class is graphical user interfaces (GUIs). We teach Java in IST 146, and so we typically teach Swing [10] interfaces first, and then use the interface-building capabilities of an IDE such as NetBeans [9]. We have never taught mobile interface development in IST 146. However, this semester, the IST 146 students are responsible for creating the Android and desktop user interfaces.

Accordingly, we are modifying the way we teach GUI development in IST 146. Students are still responsible for reading about Swing development from our text, *Java Illuminated* [1], and must take an out-of-class Blackboard quiz over the reading. However, we will not cover Swing extensively in class. Instead, we will focus on UI construction via the interface builder in the Eclipse IDE [12]. We will cover Android UI development first, then circle back and cover the same concepts applied to desktop interfaces.

The IST 146 students will create their interfaces based on specifications developed by the IST 356 students; see, for example, the initial UI sketches in Figures 2 and 3. They will write simple event-handlers that call methods of the Controller class, which is also being designed by the IST 356 students.

3.2 IST 314 (Algorithms)

The term project for IST 314 typically involves researching and reporting on an NP-Complete problem that is not covered in *Introduction to Algorithms* [5]. In addition to writing an paper on the problem and presenting it to the class, students implemented solutions to their

problems. This semester, IST 314 students are responsible for the part of the factory application that evaluates the difficulty of newly-created SuDoKu puzzles.

In the factory application, a potential SuDoKu games is first randomly created, by placing some "given" numbers into an initially blank grid. Then, a backtracking algorithm is applied to the potential game, to verify that it has one and only one solution. If the potential game is either unsolvable, or has more than one possible solution, it is discarded. However, if it has one and only one solution, the game is passed on to the next stage of the factory, to determine its difficulty level.

As stated above in section 2.3, games are graded according to difficulty based on which strategies are required to solve them. IST 314 students are each responsible for implementing at least one of the solving strategies listed on the SudokuWiki Web site [11]. Due to the limited number of students in the class, we will not end the semester with implementations of all of the solving strategies. However, we will have at least one strategy in each of the difficulty levels, which should allow the factory to begin to create games.

3.3 IST 356 (Software Engineering)

Students in the IST 356 class have several responsibilities. First, they are responsible for generating the requirements for the application. To start this process, they used index cards to create a series of user stories describing their vision of the application. Then, they created the user interface sketches that IST 146 students will use to implement the Android and desktop UIs (see Figures 2 and 3).

The IST 356 students are also responsible for designing and implementing the controller class. Initially, they used the list of user stories to sketch UML sequence diagrams to understand what messages needed to be exchanged between the UI, controller, grid, and database classes. Using these, the class is now in the process of creating a class diagram for the singleton Controller class. Once this design is complete, they will begin the process of coding the controller.

3.4 Professor's responsibilities

The author is also responsible for parts of the application. In particular, the author is responsible for the Grid part of the model, and for the database classes used by the factory and client applications. These pieces are based on the previous, self-contained desktop SuDoKu application that was used for previous IST 356 course projects. The author is also available to work as a partner in pair programming sessions (see section 5, below).

4 Development tools and environment

We chose our development tools and environment to make extensive use of open-source and free tools. Specifically, we are developing the SuDoKu project in Java, using the Eclipse

IDE [12], the Android SDK [2], and the Apache Derby database [3].

For revision control, we are using the Mercurial distributed revision control system (DRCS) [7] and the MercurialEclipse plug-in [8]. The "official" repository is hosted on Bitbucket.org [4]. Bitbucket.org offers free repository hosting for educational institutions. For communication, we have a shared Google site where we post and share documents, track issues, and so on. As much as possible, we are using Google Documents for the project.

For hardware, we configured our IST lab machines – MacBooks – to synchronize with the Bitbucket.org repository at the beginning of the semester. Students are also able to configure their own machines to work on the project if they want. The factory application will run on a dedicated Windows 7 computer in the author's office. The database server is currently running on a virtualized Windows Server 2003 machine. This server is only accessible from on the Doane campus, or through a VPN connection. Later the database will be migrated to a more public configuration.

5 Development techniques

In this project, we are using a modified Extreme Programming (XP) [13] process model. For example, XP user stories were used by IST 356 students to begin thinking about the requirements for the project. Two other key XP practices we are using are pair programming and test-first development.

All of the code for the project must be completed in a pair programming environment. Students are encouraged to pair with different partners on different assignments and tasks. If the students seem to be getting in to a rut, partner-wise, the professor forces them to work with a different partner on the next assignment. Students are also encouraged to pair with students in other classes, so that it is not unusual for an IST 146 (typically a first-year student) to pair with a student in IST 314 or 356 (typically third- or fourth-year students). Students are also able to pair with the author, when schedules allow.

We are using test-first development on this project. The students must write JUnit [6] test cases before they are allowed to write any methods. The test cases are part of the code repository, along with the source code, and so everyone in the project has access to the complete test suite. This makes regression testing after refactoring easy, especially since JUnit test execution in integrated into the Eclipse IDE.

6 Pedagogic methods

Since all three of the courses involved in this project (IST 146, 314, and 356) are taught by the author, they have different meeting times. Therefore, we have scheduled several "all project" meetings to introduce the concepts, techniques, and tools we are using in the project. In each of the meetings, small project assignments were given, designed in part to

further progress on the project, but primarily to introduce tools and techniques.

In the first meeting, we covered the overall purpose of the project, it's components and architecture, the Model / View / Controller architecture, and so on. Students were assigned to read the initial project documentation created by the author, and then come up with three questions they had about the project. Their questions, and answers provided by the author, were collated and posted on the Wiki at the Bitbucket.org site for the project.

In the second meeting, we introduced the pair programming concept. Students were assigned to write a Java method to score a game of bowling. The eleven to 21 rolls in a game were passed in to the method as an ArrayList parameter. The method had to determine the score for the game and return the result.

In the third meeting, we introduced the test-first development concept. Again working in pairs, students were assigned to design tests for a palindrome-checking method, and then write the method to satisfy the tests.

In the fourth group meeting, we covered the Mercurial DRCS, its integration into Eclipse with the MercurialEclipse plugin, and the Bitbucket.org repository. For this assignment, pairs of students had to design and code an ElapsedTime class, along with corresponding JUnit tests, and commit their code to the Bitbucket.org repository. As a group, we will chose one of the submitted classes to serve as the actual time-holding object in our application.

As the semester progresses, the project team will be more self-directed. Once the various classes are designed, it will be up to the students to decide which methods to design, and when. In addition, the author is monitoring the pairings used to develop code. When we get into "ruts," with the same pairs completing assignments over and over, new pairings will be required. In this way, the pair programming methodology is more likely to spread knowledge around the group.

7 Preliminary results

As of this writing, we are early in the project, only one-half of the way through the semester. Accordingly, we do not have much code beyond the model package classes that were leveraged off the previous desktop-only SuDoKu application. However, we have had time for good design work and some code for the project.

The IST 356 students have user stories describing the features of the applications, initial designs for the Android and desktop interfaces, sequence diagrams showing the communications between the objects in the client applications, and the beginnings of a design for the controller class. In the near future they will complete the controller class design and begin the implementation of the class.

The IST 314 students have been introduced to the overall algorithm used to create new games, and have selected the solving strategies they will implement for the project. Due to the difficulty of these algorithms, the implemented solving strategies will probably be the last parts of the system to be integrated.

The IST 146 students will cover GUIs in the near future. Once they have an introduction to creating Android and desktop interfaces, they will create UIs based on the sketches created by the IST 356 students. From the three or four interfaces created by the class, the IST 356 class will chose one to be the official "face" of each application. If time becomes an issue, priority will be given to the Android interface and application over the desktop version.

Pedagogically, we have seen students pair across the classes. In addition, we have seen the pair programming technique spread knowledge through the group. This becomes evident when students unable to attend the group meetings pair with someone who was there, thus learning the techniques "second hand."

8 Future plans

The goal for the project is to have at least prototype versions of each of the components of the application up and running by the end of the semester. We will likely have an insufficient number of solving strategies complete by the end of the term, but we should have enough to begin creating games with the factory application.

In the future, either through work by the author, through summer or honors research projects, or through further classes, we will add more capability to the system in various ways. First, we will add more solving strategies, so we can more reliably judge the difficulty of created games. We will need to move the database to a more public and secure database server, and perhaps change the server software from Derby to a more robust solution. We would also like to add a Web interface to database server, showing leader boards and high scores.

The eventual dream is to offer application for sale in the Android marketplace, with any proceeds going in to a scholarship fund for Doane IST students. Once up and running, maintenance and / or improvement projects can be taken on by future classes, as well.

9 Conclusion

Although this project is still in an early phase, we are very excited about how it is proceeding. It is valuable for the students to work on a larger project, with a larger group of students than they normally would. Working on a real-world product that could eventually be available through the Android marketplace helps students be more motivated towards the project. The project helps first-year students get to know students with more experience, and exposes them to techniques and tools that they normally would not encounter

until later in their careers.

Hopefully this application will be something we can use in our IST curriculum for many years to come.

References

[1] ANDERSON, J., AND FRANCESCHI, H. J. *Java Illuminated: An Active Learning Approach*, third ed. Jones & Bartlett Learning, 2012.

[2] ANDROID OPEN SOURCE PROJECT. Android SDK — Android Developers. http://developer.android.com/sdk/index.html, March 2012.

[3] APACHE SOFTWARE FOUNDATION. Apache Derby: Downloads. http://db.apache.org/derby/derby_downloads.html, March 2012.

[4] ATLASSIAN. Free source code hosting – Bitbucket. https://bitbucket.org/, March 2012.

[5] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms*, third ed. MIT Press, 2009.

[6] JUNIT.ORG COMMUNITY. Welcome to JUnit.org! http://www.junit.org/, March 2012.

[7] MERCURIAL COMMUNITY. Mercurial SCM. http://mercurial.selenic.com/, March 2012.

[8] MERCURIAL ECLIPSE COMMUNITY. Welcome to MercurialEclipse. http://www.javaforge.com/project/HGE, March 2012.

[9] ORACLE. NetBeans IDE Download. http://netbeans.org/downloads/index.html, March 2012.

[10] ORACLE. The Java Tutorials: Trail: Creating a GUI with JFC / Swing. http://docs.oracle.com/javase/tutorial/uiswing/, March 2012.

[11] STUART, A. Sudoku solver by Andrew Stuart. https://mail.google.com/mail/?shva=1#inbox, March 2012.

[12] THE ECLIPSE FOUNDATION. Eclipse - The Eclipse Foundation open source community website. http://eclipse.org/, March 2012.

[13] WELLS, D. Extreme Programming: A gentle introduction. http://www.extremeprogramming.org/, March 2012.