SLICE

Software **L**inked **I**nteractive **C**ompetitive **E**nvironment
Software Requirements Specification

Requested by:

Dr. Darren Lim
Associate Professor
Siena College
Loudonville, NY

Prepared by:

P=NP solutions
Zachary Fitzsimmons – Team Leader
Michael Pepe – Team Lieutenant
Anthony Parente – Systems Administrator
Matthew Ferritto – Webmaster
Renee Solheim – Document Analyst

October 28th, 2011

Contents

Chapter 1

SLICE Requirements Specification

1.1 Product Overview and Summary

Competitive programming contests are held at all different levels and scales, with a few teams to up to many participants and from single network sessions to across the world through the internet. Currently Dr. Darren Lim holds a programming contest at Siena College for local area high school students. In the Spring of 2013, Siena College will be holding the Consortium for Computer Science in Small Colleges Northeast (CCSCNE), which includes a programming contest. The current software is not user friendly and consists of several different parts. The goal of our software, SLICE, is to allow fast, both manual and automatic judging during a competitive programming contest and to integrate problem submission, judging, and scoreboard of a contest into a single application.

1.2 Development, Operating and Maintenance Environments

Our PC is an Optiplex 760 (name: seb2)

- Windows Vista Enterprise (32 bit) with service pack 1 installed
- An Intel Core2 Duo CPU processor (E7500 @ 2.93GHz)
- 4 GB of Memory

Our Mac is an iMac (model identifier: iMac5,1)

- Mac OS X, version 10.6.4
- An Intel Core 2 Duo Processor (667 MHz)
- 1 GB of Memory

Our server is an x86_64 PC

- Hostname: oraserv.cs.siena.edu
- CentOS 5.2 (final)
- Kernal: 2.6.18-92.el5
- Intel Xeon 2.66 GHz CPU
- 8 GB of Memory
- Java SE Runtime Environment (build 1.6.0_10-rc-b28)
- GCC Version 4.1.2 20071124 (Red Hat 4.1.2-42)
- Python 2.4.3

Our operating environment is a web based application that is only to be used within the SoS (Siena School of Science) network. The maintenance environment includes all of the hardware and software that is used to modify SLICE both in programming code and in visual appearance. This includes the computers in the Software Engineering Lab (stated above) and the software Adobe Dreamweaver, Adobe Fireworks, Adobe Photoshop, Internet Explorer, Mozilla Firefox, Google Chrome, Safari, etc

1.3 User Case Narratives

1.3.1 Participant User Case Narrative

Teams consist of several Participants. Participants will each be able to log into SLICE with a provided username and password generated by the Administrative User and provided to each team in a hardcopy sealed envelope. After a Participant logs in, the Participant will be able to view the contest scoreboard, submit problem-solutions or submit clarification-requests. For a problem-solution submission, a Participant will select one of the contest's problems as the associated problem and one of a list of programming languages for the language. For clarification-requests a Participant can make a general request or associate it with a problem. Participants receive a response from a Judge concerning the submission, stating one of the following: Correct Solution, Compiler Error, Running Time Error, Wrong Answer or Time-Limit Exceeded. Participants will also be able to receive both individual responses to clarification-requests from any Judge as well as publically broadcasted messages from any Judge or the Administrative User.

1.3.2 Team Advisor User Case Narrative

Team Advisors will be able to log into SLICE to view the problems and scoreboard. All Team Advisors use the same username and password to access SLICE and is provided to each advisor in a hardcopy sealed envelope. The username and password for the Team Advisor account will be generated by the Administrative User. In addition to viewing the problems and scoreboard, Team Advisors will receive all publically broadcasted contest clarifications.

1.3.3 Judges User Case Narrative

Judges will each have the ability to log onto SLICE with a username and password generated by the Administrative User and provided to each Judge in a hardcopy sealed envelope. Once logged in, a Judge will be able to view Participant submitted solutions that the system will compile. If a submission compiles, a Judge will then run the submitted programs using test input provided in SLICE that corresponds to the associated question. A Judge then submits a response to the team stating one of the following: Correct Solution, Compiler Error, Running Time Error, Wrong Answer or Time-Limit Exceeded. If the submission was a correct solution to the corresponding problem, a Judge with extra permissions in SLICE to update the scoreboard, the Scoring Judge, is sent the timestamp, problem number and team name to enter into the scoreboard. The Scoreboard Judge also has the ability to restrict the view of the scoreboard at a set time towards the end of the contest.

Judges will also receive clarification-requests (in general or concerning a specific problem) from Participants at any point during the contest. A Judge will be able to either respond (in general or concerning a specific problem) to a specific team or all Teams and Team Advisors by using a messaging prompt.

1.3.4 Administative User Case Narrative

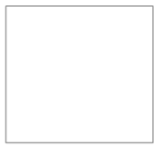
The Administrative User will be able to log into SLICE with a unique username and password. Once logged in, the Administrative User will be able to create and manage the Judge, Participant, and Team Advisor accounts. The permissions for one Judge to have Scoring Judge ability is also managed by the Administrative User. The usernames and passwords for each individual account is put into a hardcopy sealed envelope and distributed to the associated Judges, Participant or Team advisors. The Administrative User will be able to send broadcast messages to all users, or any subset i.e. all Judges, the Scoring Judge, all Participants or all Team Advisors. The Administrative User will manage the set of problems, the test input, and correct output for each contest problem. The Administrative User will also be in charge of the length of the test contest, contest start and stop times and the allowed programming languages.

1.4 UML Use Case Diagram

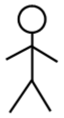
UML (Unified Modeling Language) Use Case Diagrams present a graphical overview of the functionality for a system. It uses Actors (human and non-human) to represent the users, and their goals, represented by Uses, as well as any dependencies or inheritance between them, represented by specific lines or arrows. The diagram shows what system functions are the performed by each actor.

1.4.1 Diagram Legend

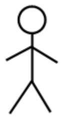
We adhere to the UML standard for our UML Use Case Diagrams, but to better explain our diagram we desribe each symbol below:



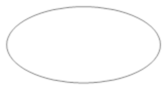
System Boundary: Where the users, human and nonhuman, located on the outside of the boundary, interact with the uses, located on the inside of the boundary, within the system



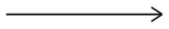
Left Side Actor: Human users which interact with the uses within the system



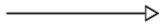
Right Side Actor: Non-Human users which interact with the uses within the system



Uses: The Activities that interact with the actors outside of the system

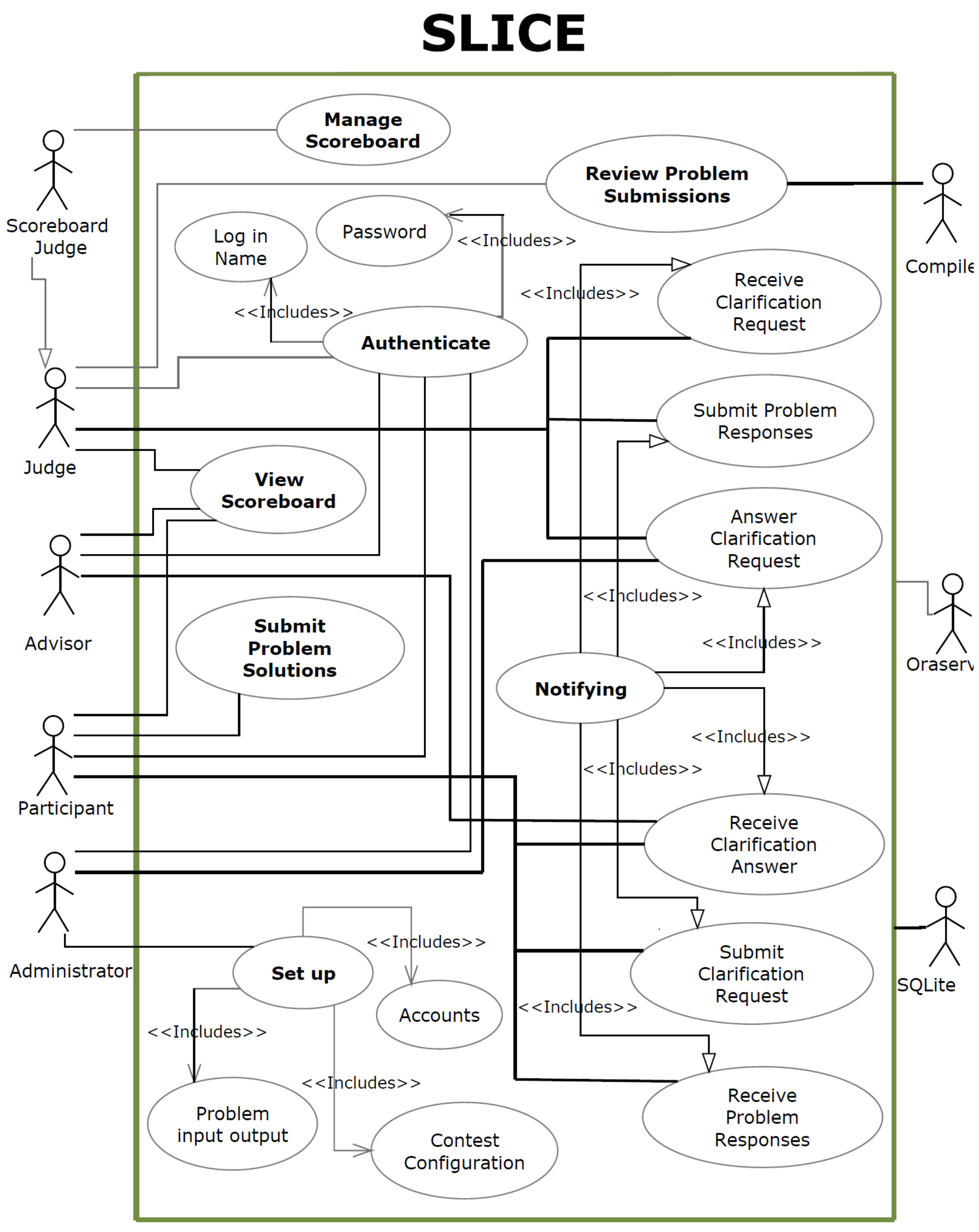


Includes Arrow: Specific items that are included within the connected



Inherits Arrow: Shows subuses of the given uses or actors which are not necessarily accessed when the parent is accessed

1.4.2 UML Use Case Diagram for SLICE

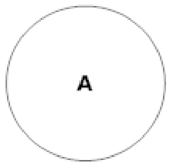


1.5 Data Flow Diagrams

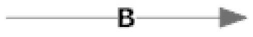
The Data Flow Diagram will show how data flows and is transformed in SLICE. The following diagrams are used as a tool to better illustrate where data is being creating, modified and going in SLICE. We will first describe the symbols used in our diagrams and how they are representing the data.
The Context Diagram is then shown, which will give a general overview of each of our data entities that interacts with the system. The Level Diagram shows more details about each process and the entities that it interacts with. Directed arrows illustrate the flow of data in this diagram. We also have a Level 1 Diagram that describes one of the major processes of the system in a greater deatil than the Level 0 Diagram and is labeled to a smaller granularity of detail.

1.5.1 Diagram Legend

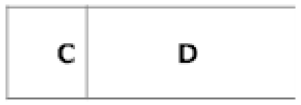
Data Flow diagrams are a means of representing a system at any level with a graphic network of symbols. Showing processes, data flows, data stores, and external entities. A key and definition for each symbol has been provided below:



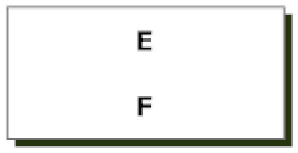
Process: The process symbol represents an activity that transforms or manipulates data. The name of the process is within the circle and represented by the letter A.



Data Flow: The data flow symbol represents the movement of data. Data flow are labeled with the description of the data that is being moved through it. The is represented by the letter B.



Data Stores: The data store symbol represents some location where data is temporarily held. The letter C represents the data store type and the letter D represents the data store name.



External Entities: External entities are sources/sinks which contribute data or information to the system or which receive data/information from it. The letter E represents the unique identifier for the entity and the letter F represents the name of the entity.

1.5.2 Context Diagram

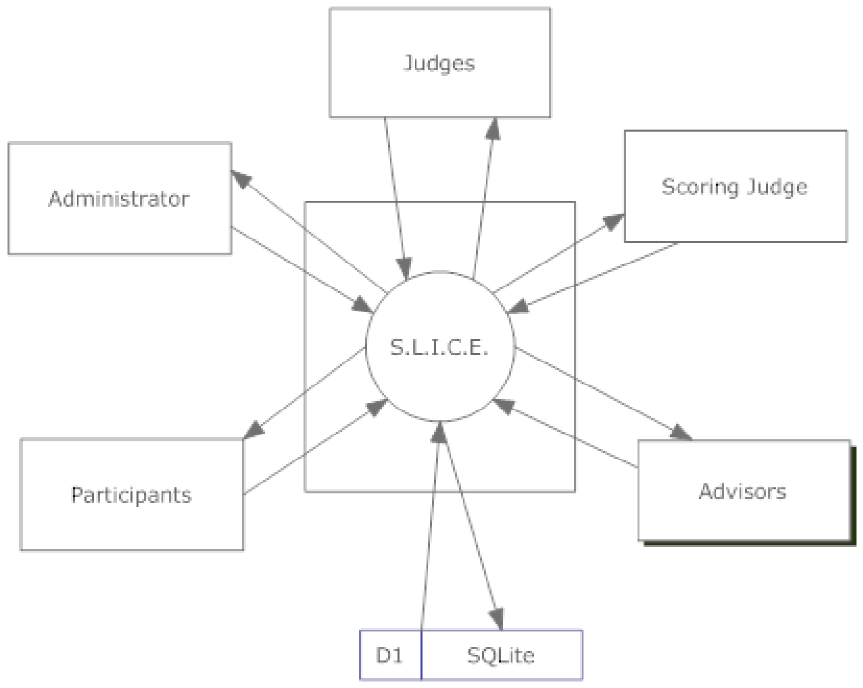


Figure 1.1: The context diagram is the highest level and represents the overall system and its interaction with its environment.

1.5.3 Level 0 Diagram

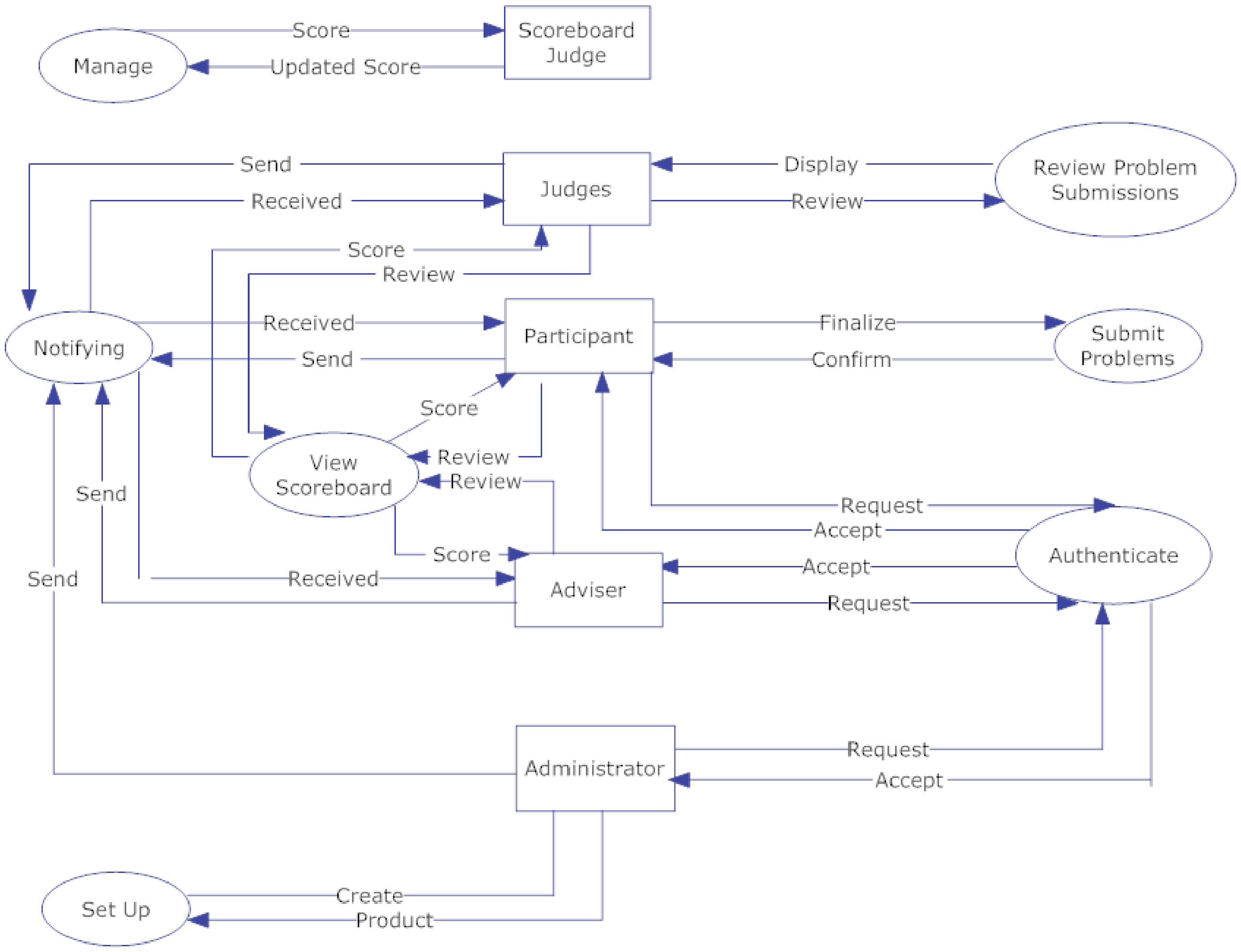


Figure 1.2: Level 0 diagrams show the major subsystems and their interactions.

1.5.4 Level 1 Diagram

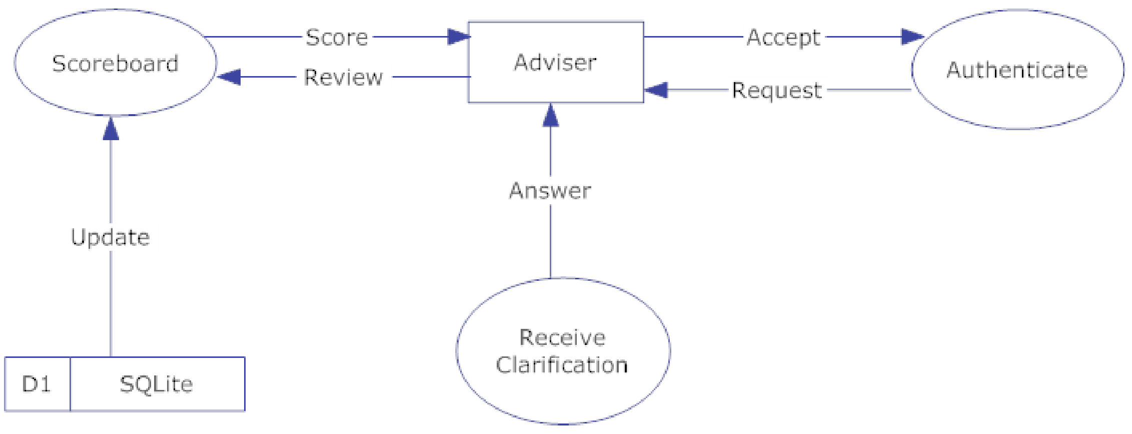


Figure 1.3: Level 1 diagrams show the processes that make up each of the major subsystems. This diagram shows the processes for the Team Advisor.

1.6 Administrative User View Prototype Used for Feature Discovery

The prototype below represents the Administrator screen. After verification of a password and username, the administrator will be able to set the number of teams, the number of problems, the number of judges, the number of languages, the start time, and the end time.

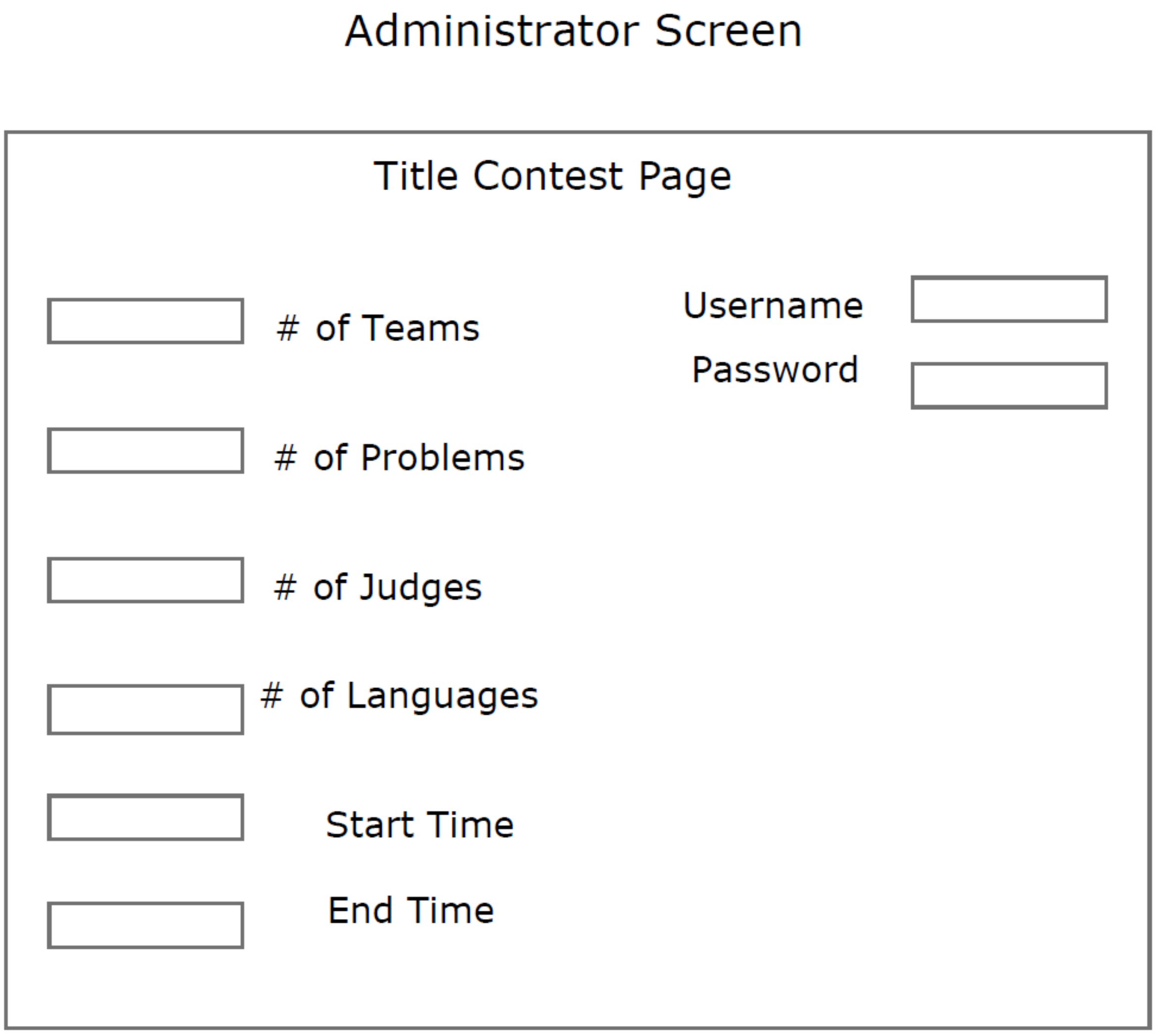


Figure 1.4: This diagram was used during client meetings with our client Dr. Darren Lim for both Administrative User functional requirements and overall contest configuration requirements discovery.

1.7 Functional Requirements Inventory

For a general functional requirement, SLICE will be able to run on all popular browsers including, Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

Below is a list of functional requirements for each human user that interacts with the system.

1.7.1 Participant User

- Will be able to log onto SLICE with a provided username and password while the contest is active
  - An incorrect username and/or password or invalid time period will result in an appropiate error message
- Will be able to log off of SLICE while the contest is active
- Will be automatically logged off of SLICE if the contest becomes inactive
- Will be able to view the contest scoreboard while active
- Will be able to submit problem submission sourcecode
  - Will be able to associate a contest problem with their problem submission
- Will be able to participate in messaging capabilities with the Judges
  - Will be able to send clarification-requests
    - Will be able to associate a problem or send general clarification-requests
  - Will receive results from problem submissions
- Will be able to receive publically broadcasted messages

1.7.2 Team Advisor User

- Will be able to log onto SLICE with a provided username and password
  - An incorrect username and/or password will result in an appropiate error message
- Will be able to view the contest scoreboard while active
- Will be able to receive all publically broadcasted messages

1.7.3 Judge User

- Will be able to log onto SLICE with a provided username and password
  - An incorrect username and/or password will result in an appropiate error message
- Will be able to log off of SLICE at any point
- Will be able to view the contest scoreboard at any point
- Will be able to review Participant problem submissions
  - Willl be able to compile problem submission sourcecode on a central server
  - Will receive compiile-time feedback
  - Will be able to execute problem submissions with provided test input
  - Will be able to compare the output to provided test output
- Will be able to participate in messaging capabilities with the Participants
  - Will be able to answer both general and contest problem specific clarification-requests
  - Will be able to give results to the Participant concerning problem submissions
- Will be able to send publically broadcasted messages
- A Judge will extra permissions called the Scoreboard Judge will be able to update the contest scoreboard
- The Scoreboard Judge will be able to recieve data consisting of correct submissions with corresponding, problem number, timestamp and Participant

1.7.4 Administative User

- Will be able to log onto SLICE with a provided username and password
  - An incorrect username and/or password will result in an appropiate error message
- Will be able to log off of SLICE at any point
- Will be able to participate in messaging capabilities will all users or any subset of users

- Will be able to complete the intial contest set-up
  - Will be able to create all other user accounts (Participants, Team Advisors, Judges)
  - Will be able to grant one Judge User, Scoreboard Judge permissions
  - Will be able to list contest problems and associate test input and output
  - Will be able to set contest start and stop times
  - Will be able to set the practice time for the contest
  - Will be able to choose the list of allowed programming languages
  - Will be able to set the time that the contest scoreboard will be active during the contest

1.8 Non-Functional Requirements

The non-functional requirement inventory is a list of non-functional system requirements. This list is composed of requirements that specify how the system should be. As more information is gathered about the project, this list may be subject to change:

- The system will be easily maintained.
- The system will be stable.
- The system will be viewable on multiple browsers.
- The system will run efficiently.
- The system will be user friendly.

1.9 Exception Handling

SLICE will be designed as a secure system. All users must authenticate to have access to SLICE and all inputs will be sanitized before being used by the system. Participants may only submit sourcecode of problem-solutions and they are compiled by a Judge, on a central server under a restricted account environment seperate from the contest problem's test input and output. There are also time limits set on both compile-time and execution time of Participant problem-solutions to avoid denial of service attacks on SLICE.

1.10 Early Subsets and Implementation Priorities

While every effort will be made to complete all the functional requirements of SLICE, the following have been defined as implementation priorities:

- The ability of administrators to manage all other users
- The ability for participants to submit source code in at least the Java programming language
- The ability for judges to review the output of automatic judging
- The ability to have a secure log in function
- The ability for SLICE to be scalable to up to 30 Participants
- The ability for the system to be updated and improved in the future

1.11 Foreseeable Modifications and Enhancements

Future modifications to SLICE may include the following:

- Allow for contests to be held simultaneously at multiple sites
- Allow for automated problem-submission responses without explicit Judge interaction
- Allow for strict security features to prevent cheating and system downtime

1.12 Testing Requirements

SLICE will be tested on the four major browsers, including Internet Explorer, Mozilla Firefox, Google Chrome, and Safari. However, testing may not be limited to those browsers. Each functional requirement will be tested on its own. If each functional requirement works on its own, then tests will be conducted on the system as a whole. The testing process will be described in greater detail in both the forthcoming Preliminary Design and Detailed Design documents. The results of the tests will be presented in the Acceptance Test document, at which point P=NP Solutions will determine if all of the functional requirements were met efficiently.

1.13 Acceptance Criteria

The acceptance criteria for the system will be defined by the functional requirements inventory, listed previously in this document. Additionally, the non-functional requirements inventory, listed previously, will be used. The functional requirements, by definition, include the abilities of the system and their ability to be tested. Non-functional requirements, on the other hand, define how the system should behave, and are not able to be tested. After the completion and testing of SLICE, $P = NP_{solutions}$ will determine which functional requirements were fulfilled.

1.14 Design Hints & Guidelines

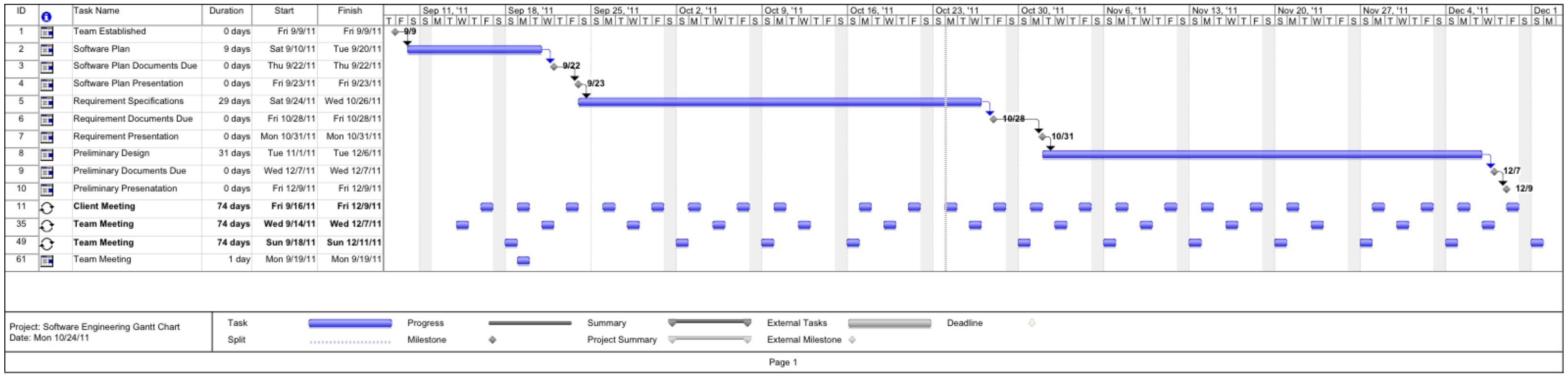
As SLICE is still in a very early stage of development, there are little to no hints and guidelines that can be described at this time. As the development of the project progresses, guidelines will be determined by both the client, Dr. Lim, and $P = NP_{solutions}$.

Appendix A

Glossary of Terms and Gantt Chart

- Actor: An entity in UML Use Case Diagrams and UML Activity Diagrams. It represents the human and non-human external entities that interact with the system
- Activity Diagram: A diagram based on the Unified Model Language(UML). This represents the processes that comprise a certain activity within the system.
- Data Flow Diagrams: Used to show how data is moved and processed within a system. There are various levels, each providing more detail then the next.
- Data Flows: A component of a Data Flow Diagram that represents the movement of data.
- Data Stores: A component of a Data Flow Diagram that represents a location in which information or data is stored.
- External Entities: A component of a Data Flow Diagram that represents any human or non-human user of a Software System.
- Functional Requirements Inventory: Define what the system will be able to do and what is testable about the system.
- Hardware: The physical parts of a computer, such as the hard drive and the CPU.
- Microsoft Internet Explorer a web-browser developed by Microsoft
- Non-Functional Requirements: Specifies how a product is supposed the be in relationship to the functional requirements.
- Process: A component of a Data Flow Diagram that represents an activity that transforms or manipulates data.
- UML: Unified Modeling Language is the industry-standard language for the specification. Visualization, construction, and documentation of the components of software systems.

- Use Case Diagram: Represents the high-level functions of the system. Also depicts how actors interact with each of those functions.
- User Case Narrative: an explanation of the functions and abilities users have for a specific Software System.



ID	Task Name	Duration	Start	Finish
1	Team Established	0 days	Fri 9/9/11	Fri 9/9/11
2	Software Plan	9 days	Sat 9/10/11	Tue 9/20/11
3	Software Plan Documents Due	0 days	Thu 9/22/11	Thu 9/22/11
4	Software Plan Presentation	0 days	Fri 9/23/11	Fri 9/23/11
5	Requirement Specifications	29 days	Sat 9/24/11	Wed 10/26/11
6	Requirement Documents Due	0 days	Fri 10/28/11	Fri 10/28/11
7	Requirement Presentation	0 days	Mon 10/31/11	Mon 10/31/11
8	Preliminary Design	31 days	Tue 11/1/11	Tue 12/6/11
9	Preliminary Documents Due	0 days	Wed 12/7/11	Wed 12/7/11
10	Preliminary Presenatation	0 days	Fri 12/9/11	Fri 12/9/11
11	**Client Meeting**	**74 days**	**Fri 9/16/11**	**Fri 12/9/11**
35	**Team Meeting**	**74 days**	**Wed 9/14/11**	**Wed 12/7/11**
49	**Team Meeting**	**74 days**	**Sun 9/18/11**	**Sun 12/11/11**
61	Team Meeting	1 day	Mon 9/19/11	Mon 9/19/11

Project: Software Engineering Gantt Chart
Date: Mon 10/24/11

Page 1

Appendix B

Annotated Bibliograpy

B.1 PC2 (Programming Contest Control) System

PC2 is a software system that is designed to support programming contests in different computing environments. PC2 allows participants to submit programs over a network to judges. The judges can recompile the submitted program, execute it, view the source code and/or execution results, and send a response back to the team. The system also supports an "automated judging" mode where judging is performed by software rather than by human judges. The system automatically timestamps and archives submitted runs, maintains and displays current contest standings in a variety of ways, and allows the judges to retrieve and reexecute archived runs. It also provides a mechanism for contestants to submit clarification requests and queries to the judges, and for the judges to reply to queries and to issue broadcast bulletins to teams. In addition, PC2 supports contests being held simultaneously at multiple sites by automatically transmitting contest standing information between sites and generating a single contest-wide standings scoreboard at each remote site. A wide variety of configurable options allow the contest administrator to tailor the system to specific contest operations. For example, the number of teams, problems, and languages in the contest; the scoring method being applied; which problems are handled by which judges; whether teams are automatically notified of the result of a submission; and the frequency of automatic scoreboard updates are all configurable. There are also mechanisms provided for editing the internal scoring database, and for recovering from various types of soft and hard errors. The system is designed to allow teams to use any language development tool which can be invoked from a command line and generates an executable file. PC2 was developed at California State University, Sacramento (CSUS). The most recent version, V9, is written in Java (using Eclipse) and is intended to run on any Java 1.5 (or greater) platform, including Windows (98/ME/2000/XP/Vista), Mac OS X (10.4+) and a variety of Unix-based systems including Solaris, Linux, and FreeBSD.
Source: http://www.ecs.csus.edu/pc2/pc2desc.html

B.2 WACS (Wide Area Contest System)

WACS is a software system that is similar to PC2, but is designed to negate the apparent weaknesses of PC2. Most prominently, PC2 is limited to the English language, and while it can be used in different centers, it cannot be used behind firewalls. WACS was developed primarily to allow teams from different places to participate in different contests. Other contest types, such as math, physics, Arabic, and history, can be easily plugged into the system, allowing participants from different cities and countries to participate in different contests. However, WAS is implemented with a focus on open-source programming languages and technologies such as XHTML, CSS, PHP, Apache, MySQL, and Ajax. Since WACS should be installed on a server (running Linux or Windows) and without any component on client machines, will reduce the burden for educational institutions or any participant without IT experts. All the configurations files, applications and servers (web, database) reside on the main server, which should be in the institution who hosts the competition. The only requirement a client needs in order to connect to this system is a web-browser. WACS was developed at the Khalifa University of Science, Technology, and Research in Sharjah (United Arab Emirates)
Sources:

- **http://www.alzaytoonah.edu.jo/Faculties/Science/Conferances/ICIT09/PaperList/Papers/Software%20Engineering/511.pdf**

- **http://www.intechopen.com/source/pdfs/8858/InTechSmart_web_based_programming_contests_management_tool.pdf**

B.3 Midas

Midas is a Programming Competition System, and is offered as a (nearly) zero configuration networking alternative to PC2. While PC2 s implementation relies on Javas RMI (Remote Method Invocation), Midas relies on a lightweight alternative written in .NET. Midas was developed at Universidad de los Andes in Bogota (Columbia).
Source: http://code.google.com/p/midas-judge/