SAN FRANCISCO STATE UNIVERSITY

FLORIDA ATLANTIC UNIVERSITY

SOFTWARE ENGINEERING

CSC 640/848, CEN 4010 – FALL 2008

INFINITY METRICS

AUTOMATIC COLLABORATION METRICS FOR JAVA.NET PROJECTS

REQUIREMENTS SPECIFICATION MILESTONE 2

Authors

Group 8

Marcello de Sales (msales@sfsu.edu)
Andres Ardila (aardila1@fau.edu)
Brett Fisher (bfisher@sfsu.edu)
Marilyne Mendolla (mmendoll@fau.edu)
Mateo Mejia (mmejia11@fau.edu)
Gurdeep Singh (gurdeep@sfsu.edu)

Advisors

Dr. Dragutin Petkovic (SFSU), Gary Thompson (SFSU),
Dr. Shihong Huang (FAU)

San Francisco, CA
Boca Raton, FL

Document ID

Date	09/27/2008
Responsible	Marcello de Sales
Document ID	INFINITYMETRICS_RS_20081023
Location	https://ppm-8.dev.java.net/source/browse/ppm-8/trunk/www/milestones/1/INFINITYNET_M2.doc

Revision History

Date	Version	Authors	Description
2008-09-28	0.1	Marcello de Sales	Template Creation
2008-09-28	0.2	Marcello de Sales	Modifying template according to specification (See Requirements documentation under Milestone 1 folder at the Directories and Files PPM-8 java.net)
2008-10-04	0.3	Andres Ardila & Marcello de Sales	Adding information on sections 2, 3, 4, 5, 6.
2008-10-16	1.0	Marcello de Sales	Revision with the feedback from Gary Thomson and Prof. Petkovic
2008-10-07	1.1	Andres Ardila	Final Revision for M1: Implemented feedback corrections (java.net capitalization, Use Case syntax), renaming project name (branding on images to be changed on next Milestone documentation).
2008-10-24	2.0	Marcello de Sales	Final documentation for Milestone 2
2008-10-24	2.1	Andres Ardila	Final Revision

Table of Contents

EXECUTIVE SUMMARY

Software Engineering is one of the fastest growing fields in today's global economy. This rapidly-evolving industry requires educational institutions to adopt innovative changes in their curriculums in order to provide students with the necessary competitive skills and experience required in today's job market. Precisely due to this globalization, an increasing number of companies are developing software in virtual, distributed environments that transcend national boundaries. As a consequence, educational institutions have redesigned their programs to simulate team software development in a controlled environment. This evolution in the pedagogy of Software Engineering, however, presents new challenges to instructors regarding student evaluation. It is therefore of paramount importance to provide instructors with the proper tools to assess individual and team progress, and to assess individual student participation within those teams in these new virtual classroom environments.

In light of these challenges, Infinity Metrics will provide cutting-edge and unique tools to instructors who choose the java.net development platform as part of a Software Engineering class curriculum. Infinity Metrics will allow instructors to monitor the participation of individual students in globally and/or locally distributed java.net group projects as well as the progress of the team as a whole in relation to other teams. Infinity Metrics will be an efficient, automated, and user-friendly web application that collects and reports on student participation metrics for such class settings. The application allows an instructor or set of instructors in different locations (i.e. concurrent Software Engineering courses taking place in different universities) to automatically collect participation data for a set of java.net projects and report on its members' contribution to the project at all stages of development. Because Infinity Metrics understands the need for intuitive, easy-to-use software in academia, the initial set-up will automatically generate the metrics categories available for the given projects, whilst allowing instructors to retain full control to modify these settings at any point throughout the semester.

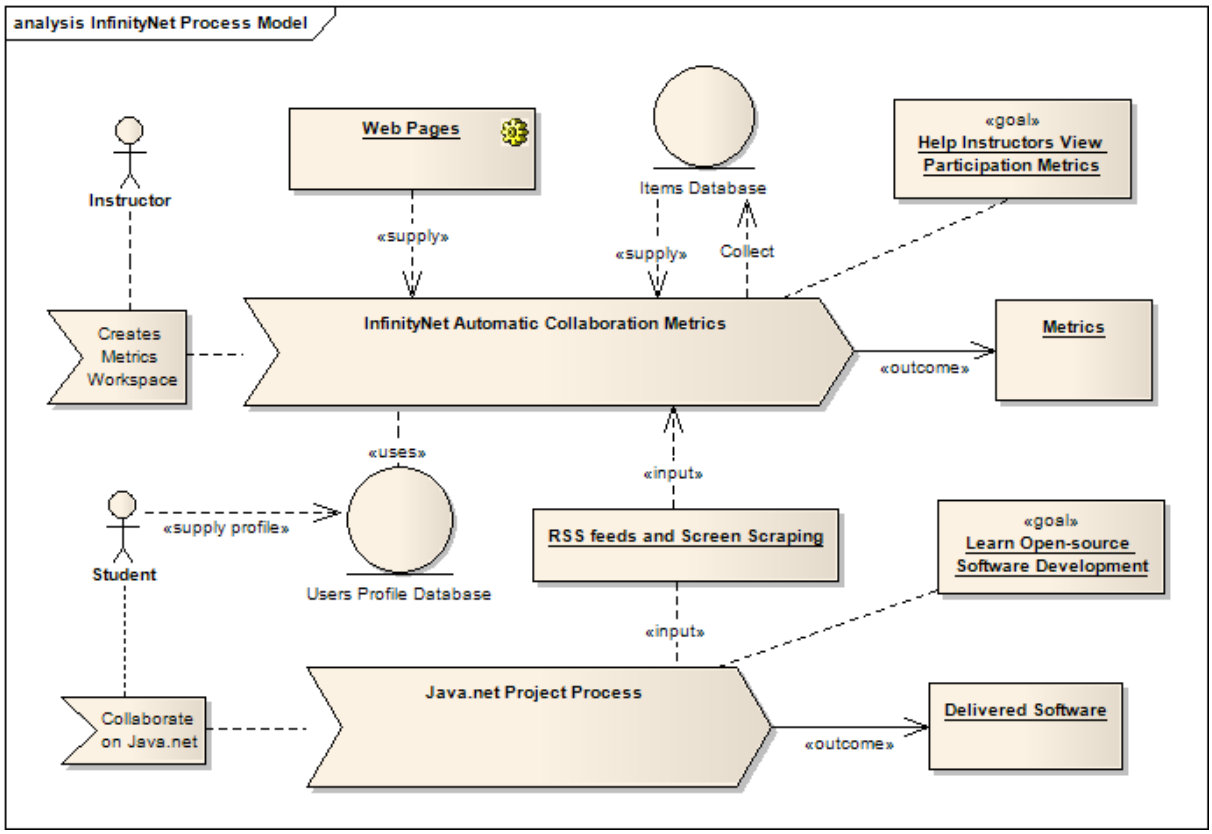
We believe that information is power. That's why our system will provide instructors with the ability to collect and view a wide ranging variety of metrics from java.net projects. It will also provide instructors with insightful information about trends and common difficulties that may arise in the development and learning processes. Our tool's seamless integration with the java.net platform, the valuable data it can extract and present (from granular to global), and its minimal maintenance overhead will allow instructors to focus their efforts in the most important aspect of their task: teaching. Infinity Metrics will thus enable and improve the quality of learning for students by providing instructors with the tools needed to be heavily involved in the progress of the class and its individuals.

1. Business Process

The business process defines the information about the overall project, gathering necessary information about the understanding of the problem that our system aims to solved. The business process is described as a high-level understanding of the system requested by Gary Thompson, and the domain vocabulary is described as the team captured both the high-level requirements and in-class information about the system.

As a matter of understanding the idea of the product, the business process model captures the main business activities, focusing on the inputs, outputs, goals and key events that drive the process. The goal of Infinity Metrics is to help instructors view participation metrics while students are learning software development on java.net.

We can describe Infinity Metrics as having two processes: one started by an instructor, and the other by students. Instructors create a configuration Workspace with the parent project in java.net from which its children projects are collected, while students provide their initial profile. As students start collaborating on java.net and, and as a result, generating RSS feeds and additional information, these metrics are collected and stored by Infinity Metrics. At this point, Infinity Metrics uses web pages to display these metrics in rich and meaningful ways to the instructor.



2. REQUIREMENTS SPECIFICATIONS

This section describes the requirements gathered from the high-level specification documentation provided by Gary Thompson. Additionally, the wiki page for the PPM project was explored. The *Formal Requirements* catalogues the categorized specification for each component to be developed. On the other hand, the *Non-Functional Requirements* section lists generalized specifications. Finally, the *Use Case Model* groups these requirements and associates them the functionalities they fulfill.

As the reader will notice, the functional and non-functional requirements are specified by means of *User Stories*, following the formal conventions and best practices of the Agile Development community, where the stories remind us of a conversation with the client.

2.1. FORMAL REQUIREMENTS

User Management		
ID	**Description**	**Priority**
US020	Users shall be able to identify themselves into the system by using their java.net username and password.	**P2**
US021	Instructors shall be able to see the students' real name on the reports.	**P1**
US039	Instructors shall be able to login into the system with my username and password created in the registration.	**P1**
US052	Instructors shall be able to view their account detail.	**P1**
US055	Users shall be able to retrieve their lost/forgotten password by email or web browser.	**P3**
US059	Users shall be able to change their profile information, but not their username.	**P3**
US060	Users shall be able to receive an email after registering on the website.	**P2**
US063	Students, who are team leaders, shall be able to identify their team members by giving their java.net usernames.	**P1**
US064	The system shall send an email to the team's dev mailing list with instructions (URL)	**P2**

Metrics Workspace		
ID	**Description**	**Priority**
US014	Instructors shall be able to manually add customized group interaction Events (disputes).	**P1**
US015	Instructors shall be able to set up a Project Metrics Configuration Workspace.	**P1**
US017	Instructors shall be able to share a Configuration with peer instructors.	**P2**
US018	Instructors shall be able to filter the students by their institution on a given Configuration.	**P2**
US019	Instructors shall be able to list the java.net username of each group member on a given Configuration.	**P2**
US022	Instructors shall be able to see each student's profile, with their Real Name, Student ID, Group Number, if he/she is a team leader, etc.	**P1**
US024	Instructors shall be able to add and modify a Configuration throughout the configuration life-cycle.	**P1**
US043	Instructors shall be able browse for projects in a given Configuration.	**P1**
US044	Instructors shall be able to browse for Event categories in a given Configuration.	**P1**
US045	Instructors shall be able to search for projects in a Configuration.	**P2**
US046	Instructors shall be able to search for categories in a Configuration.	**P2**
US053	Instructors shall be able to see the top Projects based on participation by the sum on categories or each of them.	**P2**
US058	Instructors shall be able to pause or make inactive the Metrics Workspace.	**P2**
US065	Instructors shall be able to modify which event categories are tracked for a given project in a Workspace Configutation.	**P2**
US066	Instructors shall be able to view the collection of Workspaces that he/she has created and those which have been shared with him/her	**P2**

Metrics Report		
ID	**Description**	**Priority**
US002	Instructors shall be able to track the Event of Code Repository.	**P1**
US003	Instructors shall be able to track the Event of Tracking System.	**P1**
US004	Instructors shall be able to track the Event of Emails sent to the available Mailing Lists.	**P1**
US005	Instructors shall be able to track the Event of Posts sent to Discussion Forums.	**P1**
US006	Instructors shall be able to track the Event of Documentation.	**P1**
US007	Instructors shall be able to track the Event of Disputes within teams.	**P1**
US010	Instructors shall be able to monitor events.	**P1**
US011	Instructors shall be able to view the Events by groups.	**P1**
US012	Instructors shall be able to analyze Events from the groups by their event category.	**P3**
US016	Instructors shall be able to create a Report for a Metrics Configuration.	**P1**
US025	Instructors shall be able to view Reports in a tabular format.	**P1**
US026	Instructors shall be able to view Reports in a graphical format.	**P1**
US027	Instructors shall be able to export Reports to flat text files in delimited format.	**P1**
US031	Instructors shall be able to read a Report comparing all the projects in a Configuration in a tabular way.	**P2**
US032	Instructors shall be able to read a Report for a single project in a tabular format for all the event categories.	**P1**
US033	Instructors shall be able to read a Report for a single Event category on a given Configuration Workspace.	**P2**
US034	Instructors shall be able to read the Report comparing all the projects in a Configuration in a graphical format.	**P2**
US035	Instructors shall be able to read the Report for a single project in a graphical format.	**P1**
US036	Instructors shall be able to read the Report on a single event category in graphs for the groups on a Configuration.	**P2**

Personal Agent		
ID	**Description**	**Priority**
US047	Instructors shall be able to have their Personal Agent use their java.net credentials to parse and extract the RSS feeds from each project mailing list.	**P2**
US048	Instructors shall be able to see the students' login activities such as number of visits to java.net and the last time he logged in.	**P3**
US049	Instructors shall be able to have their personal agent to go to the parent project and load the project names for a new configuration.	**P2**
US057	Instructors shall be able to have their personal agent to verify the summary of the RSS posts on a given project has been changed.	**P2**
US062	Users shall not be able to use a different username and password from java.net.	**P1**
US067	Users shall be to collect to have their personal agent to collect his/her real name and email address from java.net	**P2**

2.2. Non-Functional Requirements

General Application		
ID	**Description**	**Priority**
US001	Instructors shall be able to access to the system using a web browser.	**P1**
US008	The system shall be able to collect events' data automatically from java.net.	**P1**
US009	The system shall be able to store **relevant parts** of the RSS feeds collected on a database.	**P1**
US013	The system shall be targeted to java.net collaborative platform.	**P1**
US028	Instructors shall be able to use the system without any additional training.	**P1**
US029	The system shall not require users to have knowledge of web application development or database systems.	**P1**
US030	The system shall be able to be accessed efficiently on high-speed Internet connections.	**P1**

US038	The system shall be able to encrypt the user's username and password.	**P1**
US040	The system shall offer users online help.	**P1**
US041	The system shall respond to queries in less than 3 seconds.	**P1**
US042	The system shall offer users a secure connection through SSL.	**P2**
US061	The system shall verify the students' identity through java.net.	**P2**

2.3. USE CASE DIAGRAMS

This section describes the use case diagrams for Infinity Metrics. It represents the fulfillment of each of the User Stories into the components as a given functionality. First, each actor is described and shown in the *Actors* UML diagram, and the following sections describe the use cases for each package listed in the *Requirements* section: Users Management is responsible for the management of users and their profiles; Metrics Workspace is the area used by the instructors to create the metrics configuration, and finally, the Personal Agent API is the set of functions that the instructor's Personal Agent will perform on his/her behalf. Finally, the Disputes Tracker is the package responsible for managing disputes in any project on a given workspace.

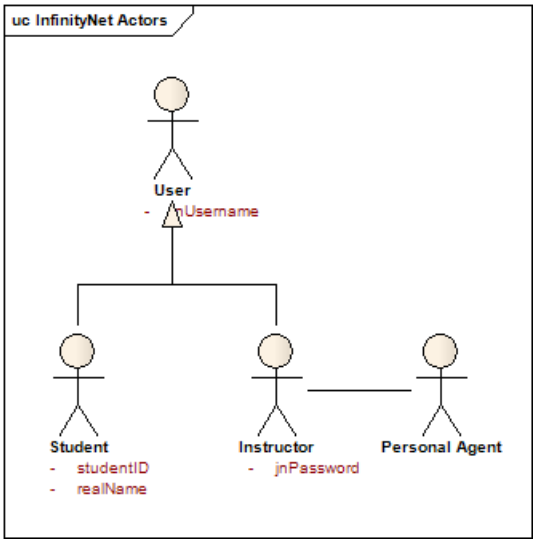
This next table summarizes the list of use cases that will be explained in the following sections.

Category	ID	Name
User Management	**UC001**	Register Student
	UC002	Register Instructor
	UC003	Login
	UC004	View Account
	UC005	Update Instructor Account
	UC006	Update Student Account
Metrics Workspace	**UC100**	Create Workspace
	UC101	View Workspace Metrics
	UC102	View Workspace Collection
	UC103	Update Workspace Configuration
	UC104	Share Workspace
	UC105	Change Workspace State
Disputes Tracker	**UC200**	Create Dispute
	UC201	Update Dispute
	UC202	Add Entries to Dispute
	UC203	Remove Entry from Dispute
Reports	**UC300**	View Individual Student Metrics Report
	UC301	View Workspace Metrics Report by Project
	UC302	View Top Performing Projects Report
	UC303	View Workspace Metrics by Event Category
	UC304	View Workspace Metrics Report by Institution
	UC305	Export Report to Delimited Text File
Personal Agent	**UC400**	Collect Events from RSS
	UC401	Collect Children Projects
	UC402	Verify user identity through java.net
	UC403	Collect Event List from Project

Before explaining the use cases, it's important to know which actors might be present on the use cases. The actors are the main entities in the system that starts the events of the system. It can be a human, or subsystem acting in favor of the system. After the Actors sub subsection, the following sections describe each of the packages of use cases.

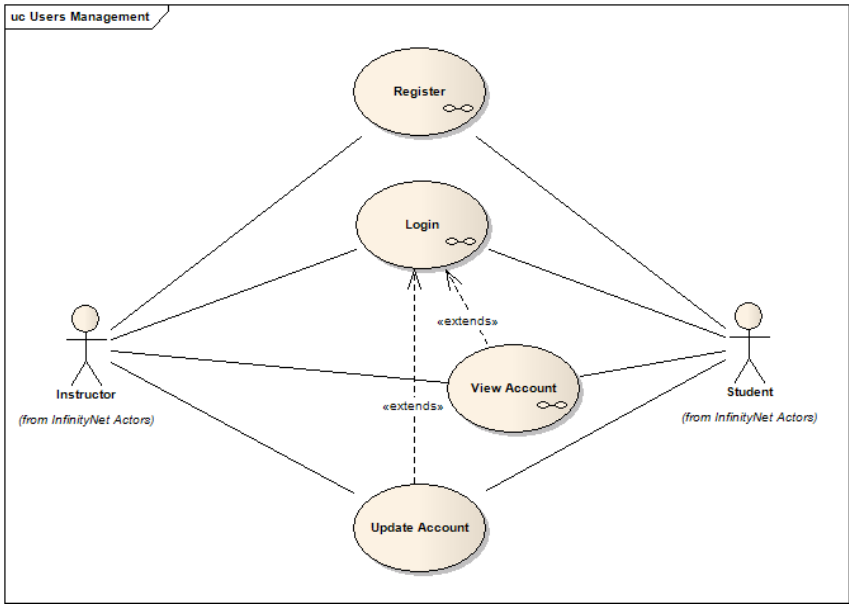
2.3.1. ACTORS

User	This is the regular user of the website. It can make login using username and password.
Instructor	Instructors are the java.net users who create the metrics. They will be providing java.net username and password to be able to navigate through the private projects.
Student	Students will just fill out forms about their personal information, institution and group.
Personal Agent	The instructor's personal agent is in the form of a web crawler that represents the instructor and virtually navigates java.net to collect information.



2.3.2. USER MANAGEMENT

The User Management package expresses to how the main actors of Infinity Metrics will use the system to create an account, register into the system and modify their information. This package is the first section of Infinity Metrics available to the customers.



UC001	Register Student
Description	A Student registers into the system.
Assumptions	The student must have a valid java.net account and email.
Precondition	The student has selected 'Student' as the user type on the registration page.
Postcondition	The student has an Infinity Metrics account.
Normal Flow	- The student enters his/her java.net username. - The student enters his/her java.net password. - The student enters his/her Real Name. - The student enters his/her email address. - The student selects his/her institution. - The student enters his/her student ID. - The student enters whether he/she is a team leader - If so, he/she enters the java.net project name. - The student submits the form. - The system validates the input. - The system verifies the student's identity against java.net. - The system sends a confirmation email to the student. - The system logs student into Infinity Metrics.
Exceptional Flow : java.net username field empty	Browser displays a message notifying student of empty field exception in java.net username field.
Exceptional Flow : java.net username already exists	Browser displays a message notifying student of duplicate username in the system and prompts for new username or to abort registration.
Exceptional Flow : java.net password field empty	Browser displays a message notifying student of empty field exception in java.net password field.
Exception Flow : Real Name field empty	Browser displays a message notifying student of empty field exception in Real Name field.
Exception Flow : Email address field empty	Browser displays a message notifying student of empty field exception in email address field.
Exceptional Flow : Invalid email address format	Browser displays a message notifying student of invalid email format exception in email address field.
Exceptional Flow : Institution empty	Browser displays a message notifying student of empty field exception in institution drop-down menu.
Exceptional Flow : Student ID empty	Browser displays a message notifying student of empty field exception in student ID field.
Exceptional Flow : Team leader selected AND project name empty	Browser displays a message notifying student of empty field exception in project name field.
Exceptional Flow : java.net identity verification failure	Browser displays a message notifying student of java.net identity verification exception and requests new credentials to retry, or abort registration process.
Benefiting Actor	Student
Requirement Traceability	US060, US061, US062, US063

UC002	Register Instructor
Description	An Instructor registers into the system.
Assumptions	The instructor must have a valid java.net account and email.
Precondition	The instructor has selected 'Instructor' as the user type on the registration page.
Postcondition	The instructor has an Infinity Metrics account.
Normal Flow	- The instructor enters his/her java.net user ID. - The instructor enters his/her java.net password. - The instructor enters the java.net project for which he/she is a project owner. - The student enters his/her Real Name. - The instructor enters his/her email address. - The instructor selects his/her institution. - The instructor clicks on the registration button. - The system validates the input. - The system verifies the instructor's identity against java.net. - The system sends a confirmation email to the instructor. - The system logs instructor into Infinity Metrics.

Exception Flow : java.net username field empty	Browser displays a message notifying instructor of empty field exception in java.net username field.
Exceptional Flow : java.net username is already registered	Browser displays a message notifying instructor of duplicate username in the system and prompts for new username or to abort registration.
Exceptional Flow : java.net password field empty	Browser displays a message notifying instructor of empty field exception in java.net password field.
Exceptional Flow : java.net project field empty	Browser displays a message notifying instructor of empty field exception in java.net project field.
Exceptional Flow : Real Name field empty	Browser displays a message notifying student of empty field exception in Real Name field.
Exception Flow : Email address field empty	Browser displays a message notifying instructor of empty field exception in email address field.
Exceptional Flow : Invalid email address format	Browser displays a message notifying instructor of invalid email format exception in email address field.
Exceptional Flow : Institution empty	Browser displays a message notifying instructor of empty field exception in institution drop-down menu.
Exceptional Flow : Institution not in list	Provide control in form for instructor to add an institution if it is not on the list
Exceptional Flow : java.net identity verification failure	Browser displays a message notifying instructor of java.net identity verification exception and requests new credentials to retry, or abort registration process.
Exceptional Flow : java.net project owners verification failure	Browser displays a message notifying instructor of java.net project ownership verification exception and requests new credentials to retry, or abort registration process.
Benefiting Actor	Instructor
Requirement Traceability	US060, US062

UC003	Login
Description	An Instructor or Student logs in into the system.
Assumptions	- The user has already registered successfully with Infinity Metrics.
Preconditions	- The user visits Infinity Metrics login page.
Postconditions	- The user is logged into the system.
Normal Flow	- The user enters his/her java.net username. - The user enters his/her java.net password. - The system verifies credentials. - The user is logged in.
Exceptional Flow : Username empty	Browser displays a message notifying user of empty field exception in java.net username field.
Exceptional Flow : Invalid username	Browser displays a message notifying user of invalid username.
Exceptional Flow : password empty	Browser displays a message notifying instructor of empty field exception in java.net password field.
Exceptional Flow : Invalid Password	Browser displays a message notifying user of invalid password exception.
Benefiting Actor	User
Requirement Traceability	US039

UC004	View Account
Description	A User views his/her account information.
Assumptions	The user has logged into the Infinity Metrics system.
Preconditions	- The user has clicked the ‘View Account’ hyperlink.
Postconditions	- The user account information is displayed.
Normal Flow	- The user clicks on ‘View Account’ hyperlink. - The system fetches user account information from database. - The system displays user account information.
Benefiting Actor	User
Dependency	UC001, UC002, UC003

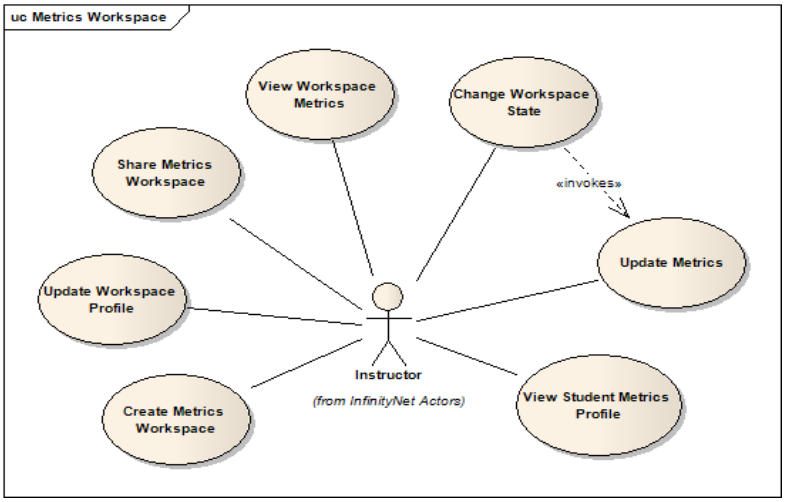
Requirement Traceability	US052

UC005	Update Instructor Account
Description	An Instructor updates his/her account information.
Assumptions	The instructor has logged into the Infinity Metrics system.
Preconditions	The instructor has clicked the 'Update Account' hyperlink.
Postconditions	The instructor account information is updated.
Normal Flow	- The instructor clicks on 'Update Account' hyperlink. - The instructor enters the updated information. - The instructor submits the form. - The system updates instructor account information in the database. - The system displays a conformation.
Benefiting Actor	Instructor
Dependency	UC002, UC003
Requirement Traceability	US059

UC006	Update Student Account
Description	A Student updates his/her account information.
Assumptions	- The user has logged into the Infinity Metrics system.
Preconditions	- The user has clicked the 'View Account' hyperlink.
Postconditions	- The user account information is updated.
Normal Flow	- The user clicks on 'Update Account' hyperlink. - The user enters the updated information. - The user submits the form by clicking the submit button. - The system updates user account information in the database. - The system displays a conformation.
Exceptional Flow : Validation fails	Browser displays a message notifying student of validation exception.
Benefiting Actor	Student
Dependency	UC001, UC003
Requirement Traceability	US059

2.3.3. METRICS WORKSPACE

Metrics Workspace package is responsible for the metrics management by the instructors. It includes regular activities such as create new Metrics Workspace, view metrics for a given Workspace, view metrics for a given project in a Workspace, and export the metrics Workspace.



UC100	Create Metrics Workspace
Description	The instructor creates a new metrics workspace for set of projects already created on java.net, identifying it by a name (e.g. term name).
Assumptions	Instructor is logged in to Infinity Metrics system
Preconditions	The instructor has ownership of a java.net project.
Postconditions	The system has created a Metrics Workspace configuration linked to this instructor.
Normal Flow	- The instructor clicks on the 'Create Workspace Configuration' link. - The instructor enters the java.net project name. - The system verifies that the instructor is an owner of the given project. - The system fetches and saves the list of child projects associated with the given parent project. - The system invokes the Personal Agent to fetch and save the list of available mailing lists and discussion forums for each of the java.net children projects. - The system will set the state of the project to 'Active'. - The system displays a confirmation page listing the event categories retrieved by the Personal Agent for each child java.net project.
Exceptional Flow : java.net project field empty	Browser displays a message notifying instructor of empty field exception in java.net username field.
Exceptional Flow : java.net project name invalid	Browser displays a message notifying instructor of invalid java.net project name exception (i.e. project space does not exist) and prompts to re-enter or abort.
Exceptional Flow : java.net project ownership verification failure	Browser displays a message notifying instructor of project ownership exception and prompts to re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC002
Requirement Traceability	US015, US049

UC101	View Workspace Metrics
Description	The instructor views the metrics for the set of projects in a given workspace by choosing the name of a configuration.
Assumptions	The instructor has been linked to a Workspace configuration.
Preconditions	The instructor is logged into the Infinity Metrics system.
Postconditions	The system displays the metrics for the projects in the Workspace.
Normal Flow	- The instructor clicks on the 'View Workspace Metrics' link. - The system fetches information from the database. - The system displays the results.
Benefiting Actor	Instructor
Dependency	UC100, UC104
Requirement Traceability	US043

UC102	View Workspace Collection
Description	The instructor views the all past & present Workspaces created by him/herself or those which have been shared with him/her.
Assumptions	The instructor has been linked to a Workspace configuration.
Precondition	The instructor is logged into the Infinity Metrics system.
Postcondition	The system displays the collection of Workspaces for that instructor.
Normal Flow	- The instructor clicks on the "View Workspace Collection" link. - The system fetches the information from the database. - The system displays the results.
Benefiting Actor	Instructor
Dependency	UC100, UC104
Requirement Traceability	US066

UC103	Update Workspace Profile
Description	The instructor updates any information on a given Workspace such as adding and removing projects within a Workspace, or renaming the Workspace.
Assumptions	The instructor has been linked to a Workspace configuration.

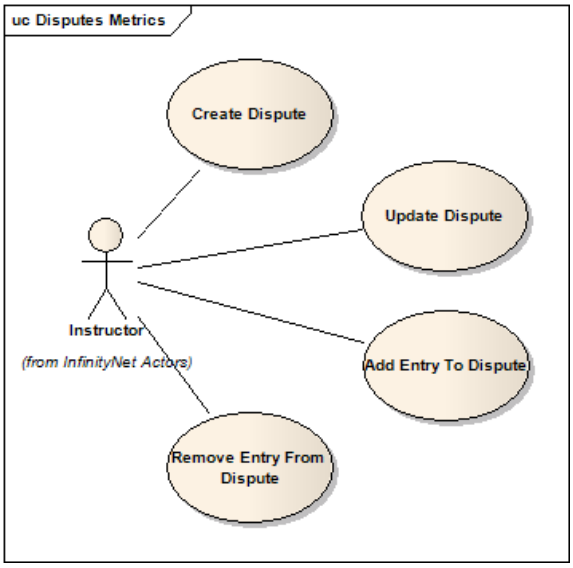
Preconditions	The instructor is logged into the Infinity Metrics system.
Postconditions	The system has updated the Workspace Profile.
Normal Flow	- The instructor clicks on the 'Update Workspace Profile' link. - The instructor selects whether to add, remove projects in a given Workspace, or rename the Workspace. - The instructor submits the form. - The system saves the updated information into the database. - A confirmation page is displayed summarizing the changes made.
Exceptional Flow : New project field empty (on 'rename')	Browser displays a message notifying instructor of empty field exception for new project name, and prompts to re-select or abort.
Exceptional Flow : Invalid new project	Browser displays a message notifying instructor of invalid java.net project (e.g. no project space in java.net) and prompts to re-select.
Exceptional Flow : java.net project ownership verification failure	Browser displays a message notifying instructor of project ownership exception and prompts to re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC100, UC104
Requirement Traceability	US024, US065

UC104	Share Metrics Workspace
Description	The instructor shares the metrics workspace with another instructor by sending him/her an invitation.
Assumptions	- The instructor sharing the Workspace has created a Workspace. - The instructor knows the java.net user id or email address of the instructor(s) with whom the Workspace is to be shared. - The instructor sharing the Workspace must be the project owner for the parent java.net project.
Preconditions	The instructor is logged into the Infinity Metrics system.
Postconditions	The Workspace is shared among instructors.
Normal Flow	- The instructor clicks on the 'Share Workspace' link. - The instructor enters the java.net username(s) or email(s) for the recipient(s). - The system verifies that the user(s) exist(s) in the Infinity Metrics users database. - The system saves the sharing configuration for the Workspace. - The system sends a confirmation email to the recipient(s). - The system displays a confirmation page to the instructor.
Exceptional Flow : Invitee not found	Browser displays a message notifying instructor that java.net username OR email address was not found in the Infinity Metrics users' database, and prompts to re-enter or abort.
Benefiting Actor	Instructors
Dependency	UC100
Requirement Traceability	US017

UC105	Change Workspace State
Description	The instructor changes the Workspace state.
Assumptions	The instructor has been linked to a Workspace configuration.
Preconditions	The instructor has logged into the Infinity Metrics system.
Postconditions	The state of the Workspace is changed.
Normal Flow	- The instructor clicks on the 'Change Workspace State' link. - The instructor selects the new state of the Workspace. - The system saves the new state in the database. - The system displays a confirmation page.
Benefiting Actor	Instructor(s)
Dependency	UC100, UC104
Requirement Traceability	US058

2.3.4. DISPUTE TRACKER

The Disputes Tracker will help project owners to include additional information about each project they are managing in a given Metrics Workspace. This can be used to flag certain activities during the development of a project such as members that are misbehaving, etc.



UC200	Create Dispute
Description	Instructor creates a new Dispute by entering a date and title for the Dispute.
Assumptions	- The Instructor has created a Workspace containing java.net projects. - A Dispute exists between team members of a project which the Instructor wishes to monitor.
Precondition	Instructor is logged in to the Infinity Metrics System
Postcondition	Dispute is added to the Dispute Tracker.
Normal Flow	- Instructor clicks on the 'Create Dispute' link. - Instructor enters title and date for Dispute (empty date defaults to current date). - Instructor enters dispute entry (blank entry defaults to no entry). - Instructor submits information to be saved in database. - System sets Dispute status to 'Active' by default. - System saves information. - The system displays a confirmation page.
Exception Flow : Dispute Title field empty	Browser displays a message notifying instructor of blank Dispute Title field and prompts to re-enter or abort.
Exception Flow : Incorrect format for date	Browser displays a message notifying instructor that date was entered in an invalid format, displays an example of the acceptable format and prompts to re-enter or abort.
Benefiting Actor	Instructor
Requirement Traceability	US007, US014

UC201	Update Dispute
Description	Instructor modifies the settings of a Dispute.
Assumptions	Instructor has already created a Dispute.
Precondition	The instructor is logged in to the Infinity Metrics system.
Postcondition	The Dispute has been updated in the Dispute Tracker.
Normal Flow	- Instructor clicks on the 'Update Dispute' link within the Dispute Tracker.

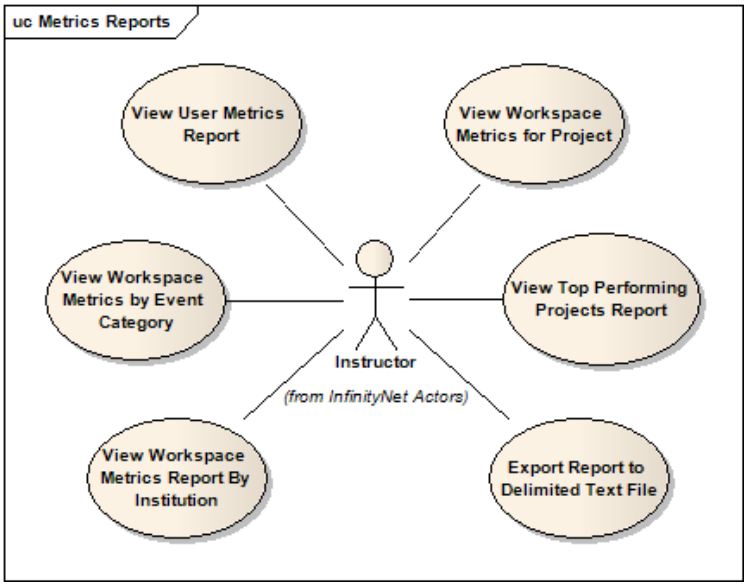
	- Instructor modifies title, date or status ('Resolved' or 'Re-open') of Dispute. - Instructor submits information to be saved in database. - If Instructor selected 'Re-open', the system sets status to 'Active', otherwise it updates the information based on the information on the form submitted. - System saves information in the database. - The system displays a confirmation page.
Exception Flow : Dispute Title field empty	Browser displays a message notifying instructor of blank Dispute Title field and prompts to re-enter or abort.
Exception Flow : Date field empty	Browser displays a message notifying instructor of blank date field and prompts to re-enter or abort.
Exception Flow : Incorrect format for date	Browser displays a message notifying instructor that date was entered in an invalid format, displays an example of the acceptable format and prompts for re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC200
Requirement Traceability	US007, US014

UC202	Add Entry to Dispute
Description	The instructor wishes to add an entry for a given Dispute.
Assumptions	The instructor has already created a Dispute.
Precondition	The instructor is logged in to the Infinity Metrics system.
Postcondition	The entry has been added to the Dispute and is saved in the database.
Normal Flow	- Instructor clicks on the 'Add Dispute Entry' link within a given Dispute. - Instructor enters the text for the entry. - Instructor submits information to be saved in database. - System saves information. - The system displays a confirmation page.
Exception Flow : Blank entry submitted	Browser displays a message notifying instructor that a blank entry was submitted, and prompts for re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC200
Requirement Traceability	US007, US014

UC203	Remove Entry from Dispute
Description	The instructor deletes an Entry from a Dispute.
Assumptions	The instructor has already created an entry in the Dispute of interest.
Precondition	The instructor is logged in to the Infinity Metrics system.
Postcondition	The selected entry has been removed from the Dispute.
Normal Flow	- Instructor clicks on the 'Remove Entry' link. - Instructor selects the entry to be removed. - Instructor submits his/her selection(s). - System removes the entry from the database. - The system displays a confirmation page.
Benefiting Actor	Instructor
Dependency	UC200
Requirement Traceability	US007, US014

2.3.5. METRICS REPORTS

The reports package only includes Use Cases for the reporting capability. This package can be seen as part of the Metrics Workspace.



UC300	View User Metrics Report
Description	The instructor wishes to view the participation metrics report for **ONE** student or a regular user by event category.
Assumptions	The instructor has created a Workspace.
Precondition	The instructor is logged in the Infinity Metrics system.
Postcondition	The system displays the student's participation metrics report by event category.
Normal Flow	- Instructor clicks on the 'View Individual Student Metrics Report' link. - Instructor chooses which student to view from a menu. - Instructor chooses which events category (ies) to view from a menu (empty selection defaults to all event categories for that particular Workspace). - Instructor submits information to the system. - System fetches information through queries. - The system displays the information.
Exception Flow : Empty student entry	Browser displays a message notifying instructor that a blank entry was submitted, and prompts for re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC100
Requirement Traceability	US010, US016, US022

UC301	View Workspace Metrics Report By Project
Description	The instructor wishes to view the participation metrics report for one or more projects for the selected event category(ies).
Assumptions	The instructor has created a Workspace and has selected which event category(ies) should be tracked.
Precondition	The instructor is logged in the Infinity Metrics system.
Postcondition	The system displays the project(s)'s participation metrics report for the given event category(ies).
Normal Flow	- Instructor clicks on the 'View Workspace Metrics Report By Project' link.

	- Instructor chooses which project(s) to view from a menu. - Instructor chooses which event category(ies) to view from a menu (empty selection defaults to all event categories for that particular Workspace). - Instructor submits information to the system. - System fetches information through queries. - The system displays the information.
Exception Flow : Empty project entry	Browser displays a message notifying instructor that a blank entry was submitted, and prompts for re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC100
Requirement Traceability	US010, US011, US016, US032, US034, US035

UC302	View Top Performing Projects Report
Description	The instructor wishes to view the top performing projects report within a given Workspace for all event categories being tracked.
Assumptions	The instructor has created a Workspace and has selected which event category(ies) should be displayed.
Precondition	The instructor is logged in the Infinity Metrics system.
Postcondition	The system displays the top performing projects report.
Normal Flow	- Instructor clicks on the 'View Top Performing Projects Report' link. - Instructor chooses the number of total projects to be viewed from a menu (blank entry defaults to all projects in Workspace). - Instructor submits information to the system. - System fetches information through queries. - The system displays the information.
Benefiting Actor	Instructor
Dependency	UC100
Requirement Traceability	US010, US016, US053

UC303	View Workspace Metrics by Event Category
Description	The instructor wishes to view the metrics within a given Workspace for individual event categories being tracked.
Assumptions	The instructor has created a Workspace and has selected which event category(ies) should be displayed.
Precondition	The instructor is logged in the Infinity Metrics system.
Postcondition	The system displays the metrics for a certain event category within a Workspace.
Normal Flow	- Instructor clicks on the 'View Workspace Metrics by Event Category' link. - Instructor chooses which event category to view metrics in from a menu (blank entry defaults to all event categories in Workspace). - Instructor submits information to the system. - System fetches information through queries. - The system displays the information.
Benefiting Actor	Instructor
Dependency	UC100
Requirement Traceability	US010, US016, US033, US036, US044

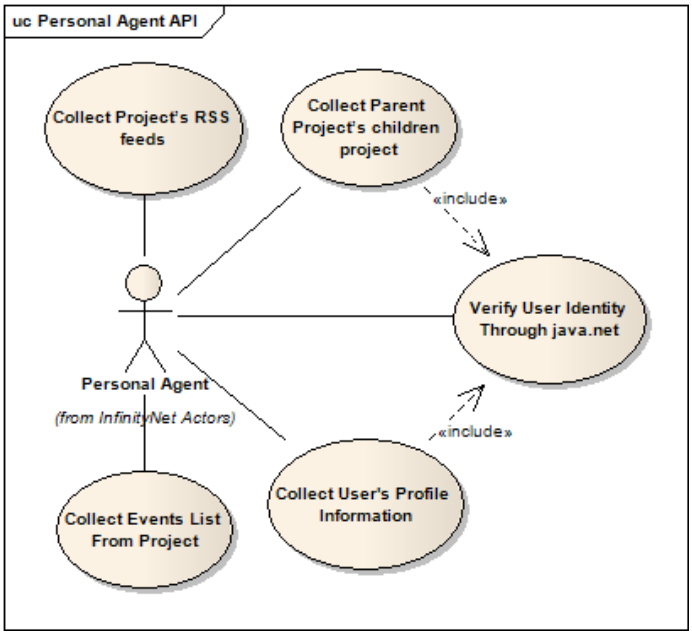
UC304	View Workspace Metrics Report By Institution
Description	The instructor wishes to view the participation metrics report in a Workspace for the selected institution.
Assumptions	The instructor has created a Workspace and has selected which event category(ies) should be viewed.
Precondition	The instructor is logged in the Infinity Metrics system.
Postcondition	The system displays the participation metrics in a Workspace for the given institution.
Normal Flow	- Instructor clicks on the 'View Workspace Metrics Report By Institution' link. - Instructor chooses which institution to view from a menu. - Instructor chooses which event category(ies) to view from a menu (empty selection defaults to all event categories for that particular Workspace).

	- Instructor submits information to the system. - System fetches information through queries. - The system displays the information.
Exception Flow : Empty institution entry	Browser displays a message notifying instructor that a blank entry was submitted, and prompts for re-enter or abort.
Benefiting Actor	Instructor
Dependency	UC100
Requirement Traceability	US010, US016, US018

UC305	Export Report to Delimited Text File
Description	The instructor wishes to export a given report to a delimited text file.
Assumptions	The instructor has created a Workspace and has selected which event category(ies) should be tracked.
Precondition	The instructor is logged in the Infinity Metrics system and is currently viewing a report in a chosen format.
Postcondition	The system exports the report to a text file.
Normal Flow	- Instructor clicks on the 'Export Report to Delimited Text File' link. - Instructor chooses name and location of output file. - System writes information to file. - The system displays the information.
Exception Flow : Output file name already exists	Browser displays a message notifying instructor that the output file name already exists, prompts for either an overwrite, re-name or abort.
Benefiting Actor	Instructor
Dependency	UC100, UC300, UC301, UC302, UC303, UC304
Requirement Traceability	US027

2.3.6. Personal Agent API

The personal agent API will be responsible for the collection of data from java.net. The personal agent will represent the Instructor on the web, and will be navigating java.net him/her, using his/her personal username and password.



For most of the functionalities that depends on java.net, personal agents can be used to ensure that the data is valid. For example, if different students can hack the system by registering their username as someone else. In order to prevent such malicious activities, the personal agent will always verify the java.net related information.

Finally, the personal agent can also collect the user's personal information during the registration process. This information might include the real name and email address used on java.net.

UC400	Collect Events From RSS
Description	The Instructor's Personal Agent collects data from the list of RSS feeds for the given Workspace.
Assumptions	The instructor has created a Workspace configuration.
Precondition	The Personal Agent is prompted by the system to go collect events from the RSS feeds for all projects in the Workspace.
Postcondition	The events collected will be saved in the database.
Normal Flow	- The Personal Agent gets the total number of events for a given category and determines if new events have been posted. - If so, the Instructor's Personal Agent collects the events from the RSS feeds in all projects belonging to the Workspace. - The events are saved to the database.
Benefiting Actor	Instructor
Dependency	UC100, UC103, UC401, UC403
Requirement Traceability	US002, US003, US004, US005, US006, US008, US009, US057

UC401	Collect Children Projects
Description	While creating a new Workspace Profile, the instructor's Personal Agent logs into the given java.net parent project and collects the names of all children projects.
Assumptions	The instructor has already created an account with Infinity Metrics.
Precondition	The Personal Agent is prompted by the system to go retrieve the list of children projects for the given parent project in the Workspace.
Postcondition	The list of children projects for a given java.net parent project will be saved in the database.
Normal Flow	- The Instructor's Personal Agent logs into java.net using the credentials for the instructor. - The Instructor's Personal Agent retrieves the list of children projects for a given java.net parent project. - The Instructor's Personal Agent saves the list into the database.
Benefiting Actor	Instructor
Dependency	UC100, UC103
Requirement Traceability	US049

UC402	Verify User Identity Through java.net
Description	While creating a new user on the system or getting updated available RSS feeds for a set of java.net projects, the Personal Agent determines whether the given java.net credentials are valid.
Assumptions	The User has provided java.net credentials to the system.
Precondition	The Personal Agent is prompted by the system to verify a user's java.net credentials
Postcondition	The system has verified the identity of the user on java.net.
Normal Flow	- The system gets the java.net credentials from the prospective user or from the database. - The system authenticates the user's credentials against java.net. - The system returns authentication results.
Benefiting Actor	User
Dependency	UC001, UC002, UC100, UC103, UC104
Requirement Traceability	US020, US061, US062

UC403	Collect Events List From Project
Description	While creating or updating a Workspace, the Instructor's Personal Agent logs into the given java.net projects and collects the event lists for all of its children projects.
Assumptions	The Instructor is creating or updating a given Workspace.
Precondition	The Instructor's Personal Agent is prompted by the system to collect the event list from a given project.

Postcondition	The event list from a given project has been saved in the database.
Normal Flow	- The Instructor's Personal Agent logs into java.net using his/her credentials. - The Instructor's Personal Agent retrieves the event list from all java.net projects for a given Workspace - The Instructor's Personal Agent constructs the RSS' URL based on the syntax rules for the given event category (i.e. "https://[PName].dev.java.net/servlets/ProjectRSS?type=message&forumID=#" for Discussion Forums, where PName is the java.net project and # is an unsigned integer representing the GUID assigned by java.net; or "https://[PName].dev.java.net/servlets/MailingListRSS?listName[MLName]", where PName is the java.net project and MLName is the mailing list (dev, issues, announce, commits, or a custom mailing list). - The Instructor's Personal Agent saves the list into the database.
Benefiting Actor	Instructor
Dependency	UC100, UC103
Requirement Traceability	US047

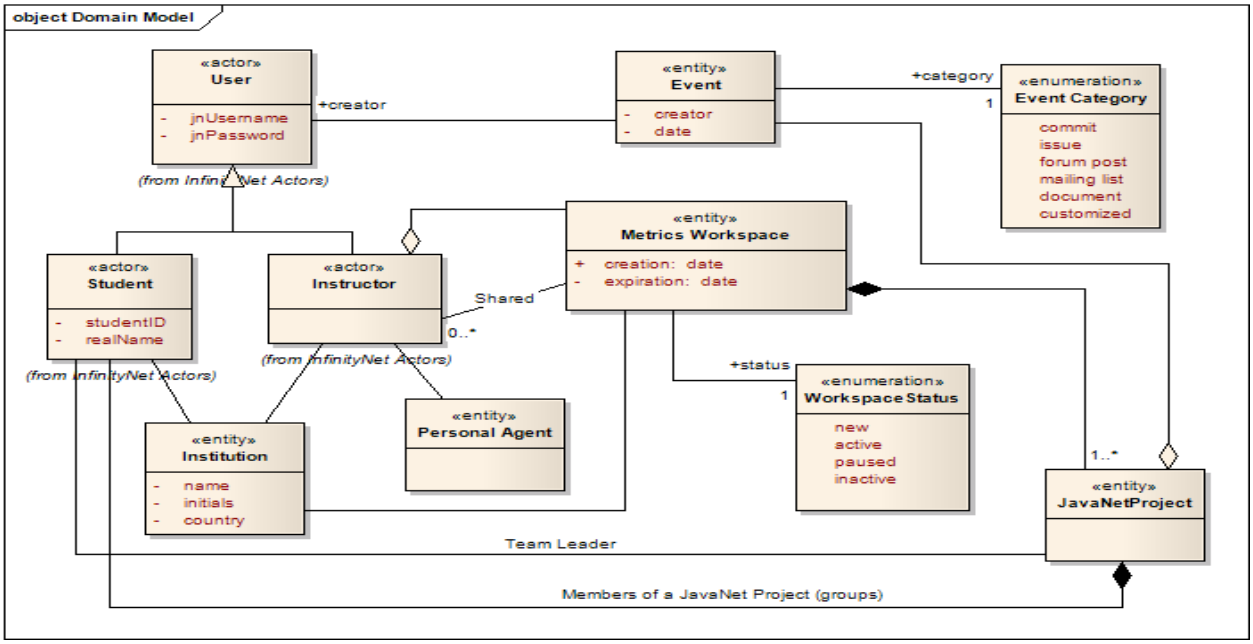
UC404	Collect User's Profile Information
Description	During the registration process, the personal agent uses the user's java.net username and password to collect his/her real name, email address and any other available personal information.
Assumptions	Any type of user is registering on the website.
Precondition	The user has an account at java.net.
Postcondition	The user's username, real name and email address is shown to the user.
Normal Flow	- The Instructor's Personal Agent logs into java.net using the credentials for the instructor - Goes to the URL of User Profile. https://www.dev.java.net/servlets/UserEdit - The agent collects the full name and email address, and verifies username.
Benefiting Actor	User
Dependency	UC100, UC103
Requirement Traceability	US067

2.4. Domain Vocabulary

The Domain Model defines a consistent, common vocabulary across a project. After analyzing the requirements and use cases, the entities and their relationships are defined below. The domain vocabulary can be thought of as the first abstraction of the high-level class diagram.

User	Derived from the Actors, a user has the username and password from java.net. It will be used by the specialized users Instructor and Student for identity verification.
Student	Students play a secondary role on Infinity Metrics where he/she just fills out the participation form with his/her personal information. Also, he can be identified as a team leader in any java.net project and is always tied to a given institution.
Instructor	Instructors are aggregated by their Metrics Workspaces for different terms on a given institution. They also can participate on the ownership of a given Workspace.
Personal Agent	The instructor's personal agent is the internet crawler that uses the professor's username and password to navigate through java.net on his/her behalf in order to get metrics information and find out more about the projects.
Metrics Workspace	The Metrics Workspace is composed by a set of java.net projects names and it can assume different states during the metrics workspace life-cycle.
java.net Project	The java.net project space name that will be tracked. It is composed of a set of java.net usernames in the form of the student names.

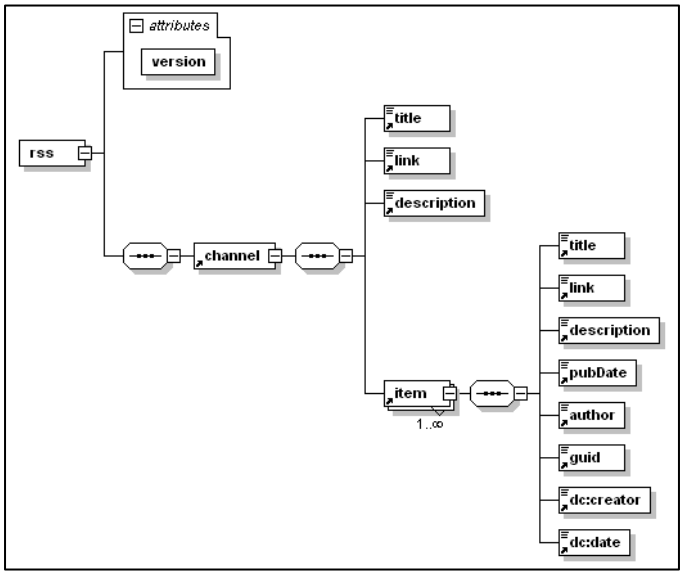
Event	Each item on a java.net project RSS feed is called an Event. Each event is part an event category such as the commit message, an issue, and a topic posted in a discussion forum or an email sent to a given mailing list. Finally, it contains a reference to the creator of the item, who can be a student or a professor. It can also refer to a given entry by the instructor. This information is associated with the java.net username used in the RSS 'creator' tag.
Event Category	This is an enumeration of the types of events, or from which category the RSS item originated. It will be associated with each event and will also include the customized values of events such as disputes on the team.
Workspace Status	It is associated with the Metrics Workspace. Active is defined by default when the instructor creates the workspace environment. In that way, the instructor can tell his Personal Agent to collect metrics online.
Dispute	Any situation that may arise among members of the same team throughout the semester.
Dispute Tracker	The Dispute Tracker enables an instructor to track Disputes within a team. It allows Instructors to add entries about Disputes, and catalogues them for monitoring.
Dispute Entry	Dispute Entries are entered by the Instructor to track significant events of a particular Dispute in a team. An Instructor can add one or more entries for every instance of a Dispute, which will be catalogued in the Dispute Tracker.



2.5. METRICS DATA COLLECTION

The most basic data to be used is the RSS feeds from the mailing lists and discussion forums. Those are the main source of data to generate the metrics. For this case, here is the XML Schema definition of the RSS feeds that are generated by java.net.

Note that an RSS feed is composed by an instance of channel, and each channel is composed by a collection of items. By the implementation of java.net RSS feeds, the channel element will always contain at least one item, showing that there are no items available.



In order to improve performance during data collection, the personal agent will be trained to collect just meaningful information from the RSS feeds. That includes unique identification of the event category, its description, location and the collection of items, in our case the collection of events generated on each of the categories. For this reason, it's relevant to describe where each of the data will come from and map each of them into the Infinity Metrics. The next table describes the meaningful data extract from some of the xml elements from the RSS feeds.

The personal agent can extract the entire data or decide extracting only a subset of characters or the element value, depending on its values. There are possibly 2 different types of RSS feeds on java.net: One for mailing lists and another for discussion forums. For each of them, different URL structure is constructed to acquire the data to be collected. They can be described as follows:

- **RSS feeds for Mailing Lists**

`https://{_PROJECT_NAME_}.dev.java.net/servlets/MailingListRSS?listName={_EVENT_CATEGORY_}`

  - **`{_PROJECT_NAME_}`** is the name of the project;
  - **`{_EVENT_CATEGORY_}`** is the identification of the event category.
    - **"announces", "cvs", "dev", "users", "issues", "commits"** are usually the default values for a new java.net project installation.

- **RSS feeds for Discussion Forums:**

`https://{_PROJECT_NAME_}.dev.java.net/servlets/ProjectRSS?type=message&forumID={_FORUM_ID_}`

  - **`{_PROJECT_NAME_}`** is the name of the project;
  - **`{_FORUM_ID_}`** is the positive integer value that identifies the forum.

Basically, the pairs **`{_PROJECT_NAME_}`, `{_EVENT_CATEGORY_}`** and **`{_PROJECT_NAME_}`, `{_FORUM_ID_}`** and will be used to uniquely identify each channel type of a project. As one can notice, both URLs differ on the name of the server-side component call and parameters name. There is NO need whatsoever in collecting those values, since they can be re-generated on the fly.

When it comes to the identification of each event, the Agent has to be capable of extracting the identifiers of each of them. As described below, they contain the same variables from the channel URL structure, plus their unique identification for each of them.

- **Mailing Lists Event**

```
https://{_PROJECT_NAME_}.dev.java.net/servlets/ReadMsg?list={_EVENT_CATEGORY_}&msgNo={_EVENT_ID_}
```

- **RSS feeds for Discussion Forums:**

```
https://{_PROJECT_NAME_}.dev.java.net/servlets/ProjectForumMessageView?
forumID={_FORUM_ID_}&messageID={_EVENT_ID_}
```

  - **{_EVENT_ID_}** is the unique identification of the event.

In this way, the following table describes the parsing strategies and what meaningful data is being collected.

RSS Element (XPath)	Parsing Strategy	Infinity Metrics Data Binding
/rss/channel/title	Complete Data	The title for each event channel. This is used to visually identify each event category. It will be used just for the user interface.
/rss/channel/link	Partial Data	The agent extracts first the project name to make sure the data is valid from the URL being used. Depending from where the RSS was generated from, there are different channel identifiers to be captured: • **Mailing lists**: must extract the value of the HTTP parameter "**listName**"; • **Discussion Forums**: must extract the value of the HTTP parameter "**forumId**".
/rss/channel/item[x]/link	Partial Data	Each event is identified by a unique identification. • **Mailing lists**: must extract the value of the HTTP parameter "**listName**"; • **Discussion Forums**: must extract the value of the HTTP parameter "**messageID"**
/rss/channel/item[x]/pubDate	Complete Data	This is the event's publication date. It will be used for filtering events by its creation time.
/rss/channel/item[x]/author	Complete Data, with match	Different mailing lists can contain different values for the author of the event. Here is the list of the possible values, categorized by the types of RSS feeds: • **Issues, forums**: the regular username; • **users, dev**: the complete full name; Must match registered user's full name. • **commits**: the email address. Must match registered user's java.net email address.

Some URLs from java.net gives the binding values the variables described above. The idea is to bind these variables with the values from the User's Profile java.net URL, which contains the username, email address and possibly his/her full name, depending on the configuration of java.net. In this case, the system should collect this information directly from the user, asking to copy the same name as java.net.

3. User Interfaces

As user interfaces goes, Infinity Metrics will provide the regular registration forms and login forms as any other web application, containing their own set of required field for the type of user: instructor or students. However, the most important aspect of the application is the instructor's metrics workspace.

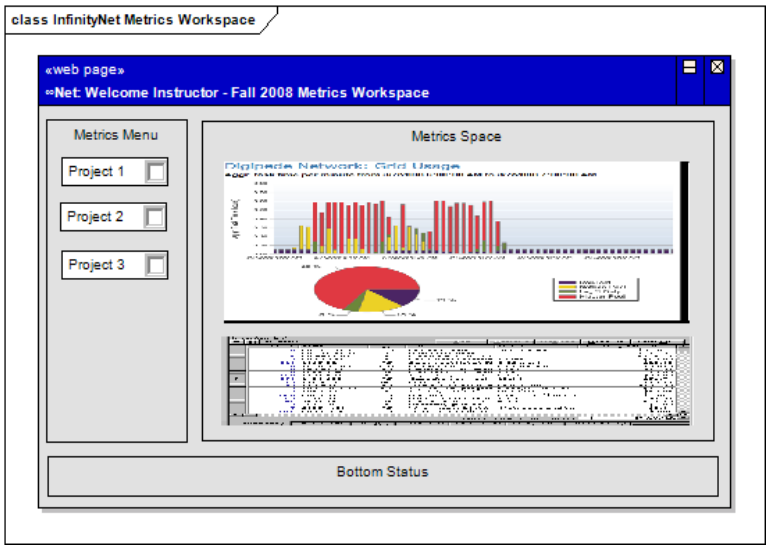
3.1. Mock-ups

3.1.1. Metrics Workspace

The Metrics workspace consists of the top bar, a left-side navigation menu and the main area in the center of the screen.

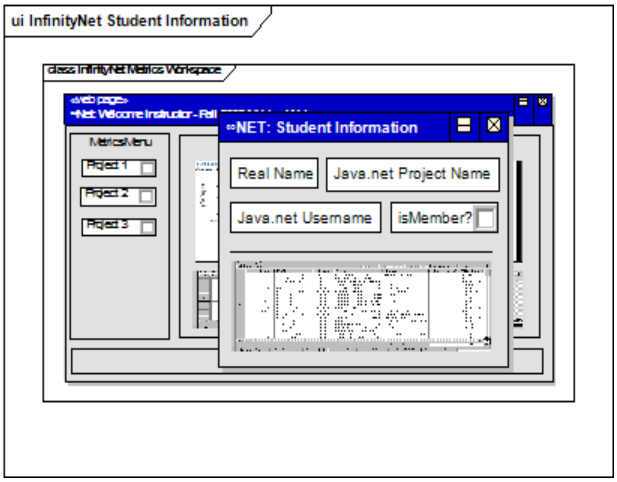
The top bar contains the instructor's name, the name of the current workspace being viewed and the name of other instructors with whom the Workspace is shared.

The left-side navigation menu lists projects contained in the current Metrics Workspace, as well as general related functional items such as edit list, download events data file, search student, etc.



Finally, the center are contains the metrics in tabular and the graphical representation, from the aggregated events data for the general set of projects. When one of the projects is clicked from the left-side navigation, the tabular data and graphical representation are changed accordingly.

3.1.2. Student information for Instructors

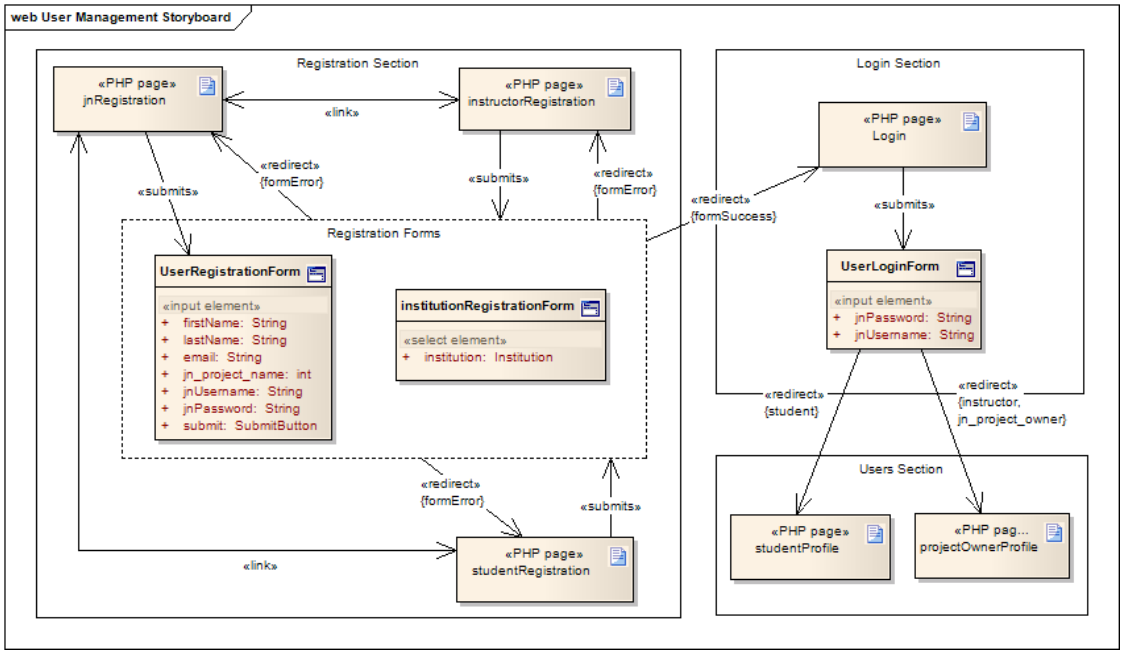


When the instructor clicks on a given student, he/she is shown the students' participation metrics in the current workspace metrics opened. Note that the information displayed shows his participation in the project only. It shows his list of team members and which one is the team leader. Also, it shows statistical information about the overall performance in the current project on the workspace for all the configured event categories such as commits to repository, participation on issues and posts on discussion forums.

3.2. STORYBOARDS

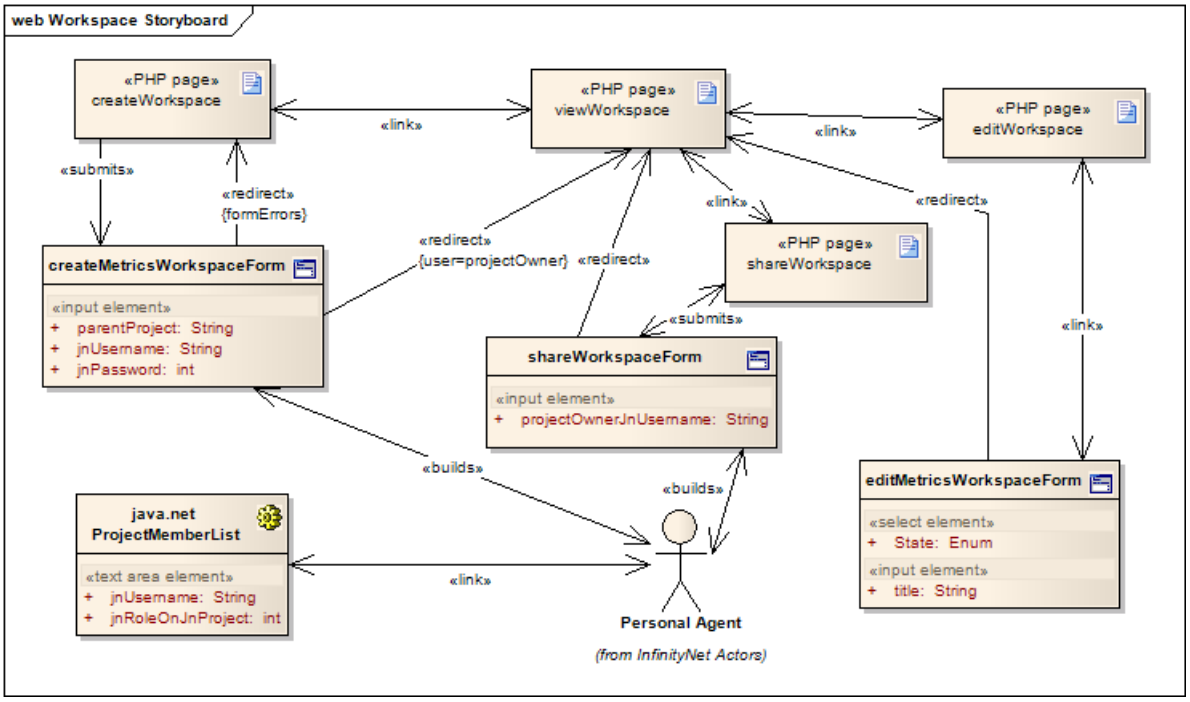
3.2.1. USER MANAGEMENT STORYBOARD

Users will be taken to the User's registration page. This page will be asking regular users for registering at the jnRegistration page. If they happen to be from an institution, then the instructors and students have a separate registration process. After a successful registration process, the user is taken to the login section, where they can use the username and password as like the one used on java.net. After a success login into Infinity Metrics, then the user's profile is shown for a team member or a project owner.



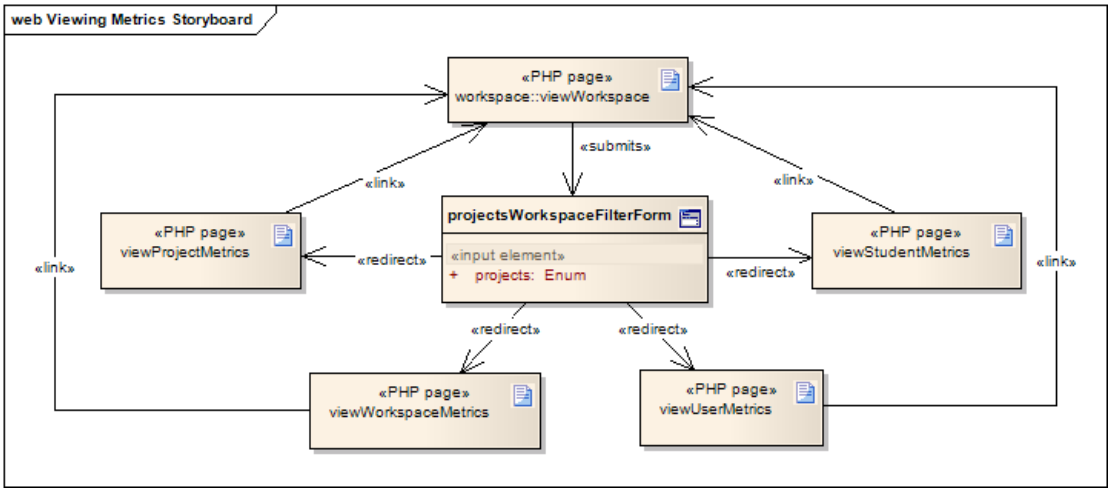
3.2.2. METRICS WORKSPACE STORYBOARD

The metrics workspace is used by project owners to create, edit and update workspaces. It includes the boards of metrics reports. It will use the Personal Agent to verify the project's ownership, and when the user wants to share the workspace metrics.



3.2.3. METRICS REPORTS STORYBOARD

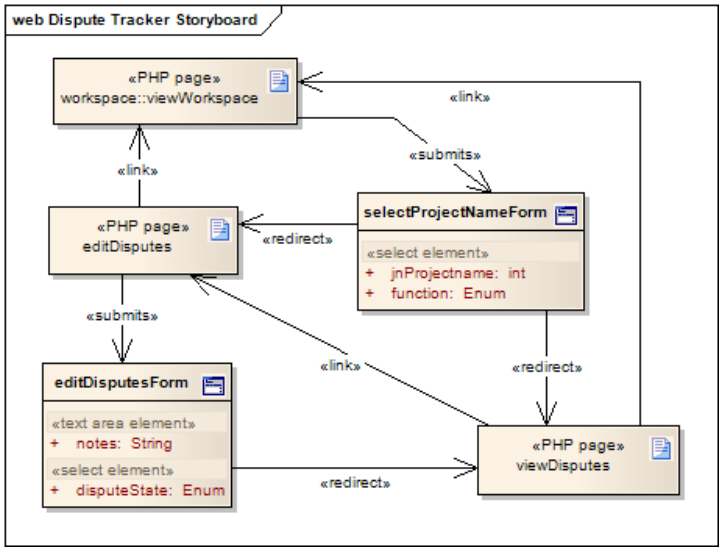
The metrics will give options to the user to browse the projects list, and from each of them it will be possible to view more reports such as for a given project member. For institutions, that would be in the context of a project, its members and the events. For each of the metrics visualization pages, the project owner can go back to the first page to view his/her workspace.



3.2.4. DISPUTE TRACKER STORYBOARD

The metrics for the dispute tracker is used to control additional information about a project. The project owners can add and update event disputes.

From a given Metrics Workspace, the project owner can select one of the existing functionalities: view disputes or edit disputes. Viewing disputes will just list the list of disputes. The edit disputes updates the information on the metrics, as well as it shows that a given dispute has been finished.

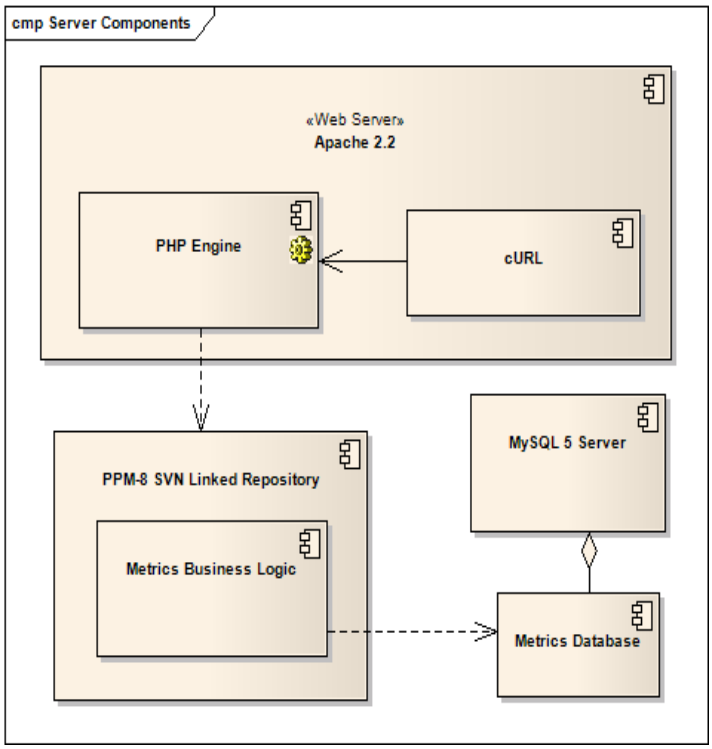


4. High-level System Architecture

The system is intended to be developed on the Internet. Therefore, main client to the application is a web browser and the server contains technologies to publish server-side web pages, integrated with a database system.

4.1. Web Server and Web technology

The font-end of the Infinity Metrics implementation consists of an extensive User Interface, constructed mainly in PHP and perhaps Ajax. Infinity Metrics will be redesigning and expanding on the core functionality of the StatsWidgets Project, and in doing so, will work to bring this functionality up-to-date with current technology by implementing a well-constructed, web-based user interface along with its back-end crawler. In order to offer most of the features planned, the server will feature an Apache Web Server, along with additional PHP modules. The most important additional module is called cURL, which can be used as an HTTP client.



As for the PHP source-code, it will be deployed to the SFSU T-2000 server using Subversion infrastructure during the development process. In this way, a given revision **must** be tested and selected to be deployed. This revision includes the core components for the Metrics Business logic, with the intent to make use of additional PHP frameworks.

4.2. Database Server

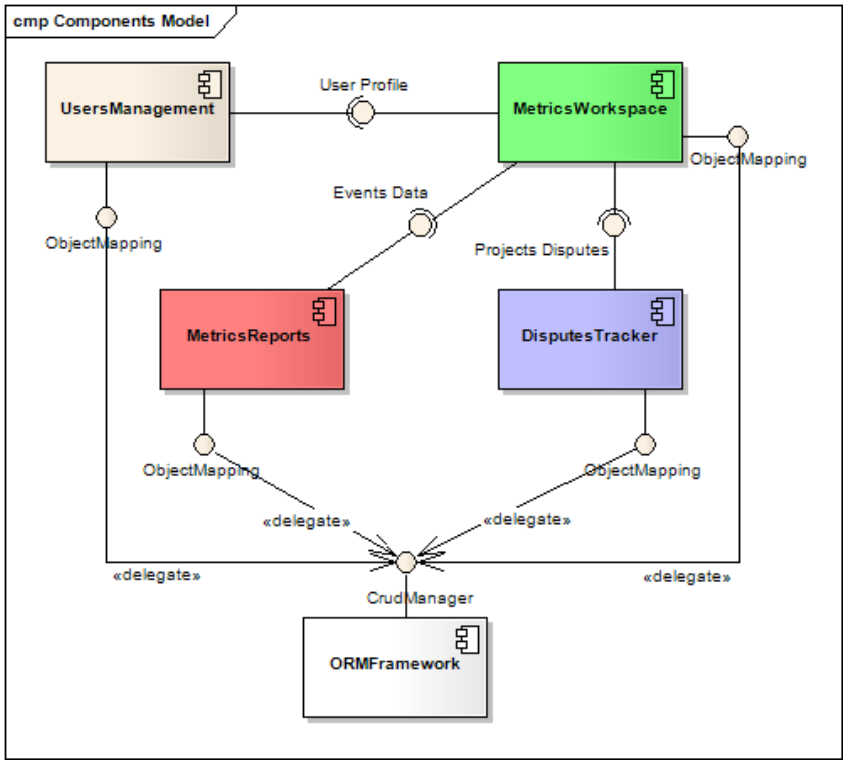
In order to overcome the short-comings of RSS feeds where older data drops as new data is acquired, the back-bone of Infinity Metrics system will contain a MySQL database for information storage. In this way, the metrics data will be categorized in a meaningful way to be later retrieved. The MySQL server consists of the version 5, and will use advanced stored procedures and triggers during implementation.

At the end of a given semester, or upon project completion, the data that was collected and stored will not be discarded. This will allow instructors not only to maintain access over the project life-cycle and eventual project completion, but into the future, where analysis and comparisons can be made between different Workspace Configurations in order to infer trends or the effects of changes in the structure of the course by student participation and progress in the projects.

5. ARCHITECTURAL DESIGN

Infinity Metrics is designed using an extended version of the Model-View-Controller (MVC) architectural layers design-pattern. It includes the Data layer, among the traditional ones, responsible for the persistent space for the objects of the Model layer, which is a mapping between the business classes from the Model to the relational database. In this way, we call it DMVC to explicitly describe the use of an Object-Relational Mapping (ORM) framework, which will be evaluated and selected during Milestone 3.

We will take the approach of separating the Use Cases packages into separate component, being responsible for the given functionality in as a well-contained collection of complete functionalities. Each component is then modeled using the DMVC layers introduced above and mixed together in the architectural diagram.



Component	Description
Users Management	It's responsible for user management in general. It encapsulates the types of users, their profiles, and the personal agent for each type of users. It exposes the User Profile interface to the Metrics Workspace Profile.
Metrics Workspace	The main interface for project owners is placed into the Metrics Workspace. It will be responsible for managing the project owner's metrics into groups of projects. In this way, it is composed by the Metrics Reports and Disputes Tracker components, which has its own responsibilities.
Metrics Reports	Responsible for the generation of reports. It will be where the RSS data feeds are encapsulated.
Disputes Tracker	Disputes are generated for any project on a given workspace, whenever it is necessary.
ORM Framework	The Object-Relational Model framework is responsible for managing the mappings of objects of all components to the data layer, or the Database. This component is one to be selected from the available ones on the PHP Community.

The next subsections describe each layer that is used on each of the components. After each of the subsections, the business classes that make part of the components were highlighted with the same color for better identification. The complete Architectural Diagram is located in the end of the subsections.

5.1. MODEL

The model is composed four main components, implemented as PHP classes: Users, Metrics Workspace, Participation Metrics and Dispute Tracker. Users are subdivided into Instructor and Student users. They contain the user's information, including their java.net username & password, their real name, email address (and in the case of Students, the Student ID). Users are members of an Institution, which is in itself a class containing the Institution's name, abbreviation, city, province and country. Instructors can create Metrics Workspaces, which include a title and description as well as a state (e.g. 'Active', 'Paused', 'Inactive'). Workspaces contain java.net Projects, composed of a parent and children projects. The Workspace can be shared with other Instructors. The list of Event Categories available in each of the children projects is collected from java.net and allows the system to validate project ownership for Instructor registration and team leaders. These Event Categories, in turn, populate the list of RSS feeds that the Instructor's Personal Agent will visit in order to collect Participation Metrics. In addition, the instructor may add Disputes, which in turn contain Entries that will allow him/her to track individual events within a team pertaining to a Dispute.

5.2. VIEW

The presentation of model is based on the Participation Metrics collected by the Personal Agent, which can be viewed through the Reports in several PHP pages in either graphical or tabular presentation. These views include Participation Metrics by group (viewProjectMetrics), by student (viewStudentMetrics), by Event Category (viewWorkspaceMetrics) and by top performing project, or exported (exportMetrics). Views are also provided for user accounts (studentProfile and projectOwnerProfile). In addition, a View is provided for the set of Workpspaces that have been created and/or shared with an Instructor (viewWorkspace). Lastly, views are also available for the Dispute Tracker on which the Instructor can track entries related to a given Dispute (viewDisputes).

5.3. CONTROLLER

The Controllers implements the functionality of the Use cases. Basically, for each package of the use cases, a controller is given the responsibility to implement each functionality. The controller MUST list all the methods, and those that are not implemented must through the exception to the client.

As it will be shown on the diagram, each component will have a mapping of 1-to-1 with the controllers. For example, the Use Cases package Users Management will have its UserController, implementing all the use cases mapped into method calls. They will extensively depend on the model and will select the appropriate view after its execution.

5.4. DATA

The Data Persistence will consist of an ORM framework that will provide a transparent data storage mechanism for the state of the application. It will map the PHP persistent classes of the Model layer, and will implement the Create-Retrieve-Update-Date operations, also known as CRUD operations. Below is a table that summarizes the frameworks to be evaluated during the first iteration of the milestone 3.

Basically, such framework should be able to speed the development process, avoid having to develop the same configuration for the same usual code. Given the Data Mode, the code ORM framework should offer a simple code generation tool that can generate the Object Code and the CRUD Views. In this way, developers will just update and choose which of them are necessary. The image below depicts these ideas from the framework QCodo.

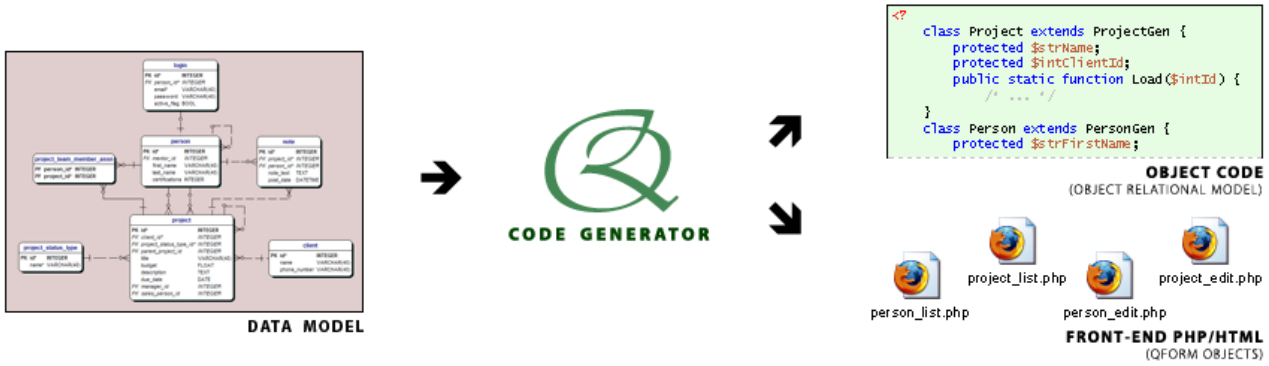


Image referenced from http://www.qcodo.com/images/codegen.png

Although the use of ORM helps development, it should give support to the execution of SQL in case of very complex objects graph. That should be one of the main features supported by the framework. For features like lazy loading and cascade, it should be provide performance and avoid multiple SQL requests. The maximum allowed should be not more than 4 SQL commands executed per batch.

- **Doctrine**: It uses descriptors to describe the tables and its columns in a very clean way. It supports CRUD, but might need integration with other frameworks such as PEAR: http://www.doctrine-project.org/documentation/cookbook/1_0/?one-page

- **PHP Object Generator**: It is the easiest way to generate the ORM in PHP, given that there's no need for other installations, includes, etc. Online generation.
  http://www.phpobjectgenerator.com/

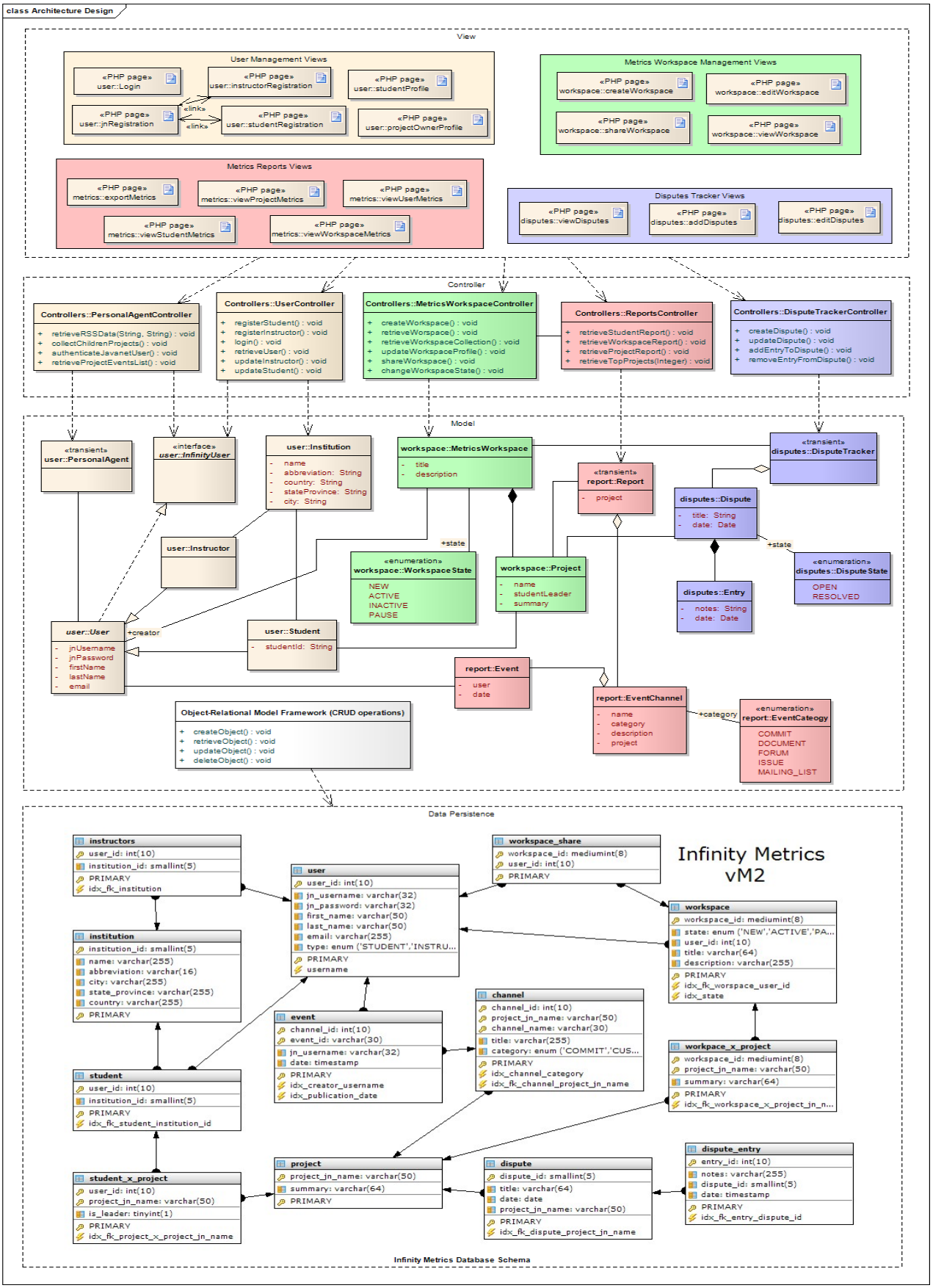
- **Propel**: Maybe the most powerful ORM implementation in PHP, features all the properties of an ORM;
  http://propel.phpdb.org

- **QCodo**: Generate classes and CRUD pages.
  http://www.qcodo.com

- **Xyster**: From ZendFramework, and features ORM mapping and dependency injection.
  http://xyster.devweblog.org/documentation/guide/xyster.orm.html

- **SilverStripe**: It has a content management system and the ORM framework.
  http://www.silverstripe.com/showcase

- **CakePHP, Synfony, Jelix, Phpontrax, ZoopFramework** are the selections for Rapid Application development with a complete MVC stack, evaluated depending on team's performance.


class Architecture Design
View
User Management Views
«PHP page» user::Login
«PHP page» user::instructorRegistration
«PHP page» user::studentProfile
«PHP page» user::jnRegistration
«PHP page» user::studentRegistration
«PHP page» user::projectOwnerProfile
«link»
Metrics Workspace Management Views
«PHP page» workspace::createWorkspace
«PHP page» workspace::editWorkspace
«PHP page» workspace::shareWorkspace
«PHP page» workspace::viewWorkspace
Metrics Reports Views
«PHP page» metrics::exportMetrics
«PHP page» metrics::viewProjectMetrics
«PHP page» metrics::viewUserMetrics
«PHP page» metrics::viewStudentMetrics
«PHP page» metrics::viewWorkspaceMetrics
Disputes Tracker Views
«PHP page» disputes::viewDisputes
«PHP page» disputes::addDisputes
«PHP page» disputes::editDisputes
Controller
Controllers::PersonalAgentController
+ retrieveRSSData(String, String) : void
+ collectChildrenProjects() : void
+ authenticateJavanetUser() : void
+ retrieveProjectEventsList() : void
Controllers::UserController
+ registerStudent() : void
+ registerInstructor() : void
+ login() : void
+ retrieveUser() : void
+ updateInstructor() : void
+ updateStudent() : void
Controllers::MetricsWorkspaceController
+ createWorkspace() : void
+ retrieveWorspace() : void
+ retrieveWorkspaceCollection() : void
+ updateWorkspaceProfile() : void
+ shareWorkspace() : void
+ changeWorkspaceState() : void
Controllers::ReportsController
+ retrieveStudentReport() : void
+ retrieveWorkspaceReport() : void
+ retrieveProjectReport() : void
+ retrieveTopProjects(Integer) : void
Controllers::DisputeTrackerController
+ createDispute() : void
+ updateDispute() : void
+ addEntryToDispute() : void
+ removeEntryFromDispute() : void
Model
«transient» user::PersonalAgent
«interface» user::InfinityUser
user::Institution
- name
- abbreviation: String
- country: String
- stateProvince: String
- city: String
workspace::MetricsWorkspace
- title
- description
«transient» report::Report
- project
«transient» disputes::DisputeTracker
disputes::Dispute
- title: String
- date: Date
user::Instructor
«enumeration» workspace::WorkspaceState
NEW
ACTIVE
INACTIVE
PAUSE
workspace::Project
- name
- studentLeader
- summary
«enumeration» disputes::DisputeState
OPEN
RESOLVED
disputes::Entry
- notes: String
- date: Date
user::User
- jnUsername
- jnPassword
- firstName
- lastName
- email
+creator
user::Student
- studentId: String
report::Event
- user
- date
report::EventChannel
- name
- category
- description
- project
«enumeration» report::EventCateogy
COMMIT
DOCUMENT
FORUM
ISSUE
MAILING_LIST
Object-Relational Model Framework (CRUD operations)
+ createObject() : void
+ retrieveObject() : void
+ updateObject() : void
+ deleteObject() : void
Data Persistence
Infinity Metrics vM2
instructors
user_id: int(10)
institution_id: smallint(5)
PRIMARY
idx_fk_institution
user
user_id: int(10)
jn_username: varchar(32)
jn_password: varchar(32)
first_name: varchar(50)
last_name: varchar(50)
email: varchar(255)
type: enum ('STUDENT','INSTRU...
PRIMARY
username
workspace_share
workspace_id: mediumint(8)
user_id: int(10)
PRIMARY
workspace
workspace_id: mediumint(8)
state: enum ('NEW','ACTIVE','PA...
user_id: int(10)
title: varchar(64)
description: varchar(255)
PRIMARY
idx_fk_worspace_user_id
idx_state
institution
institution_id: smallint(5)
name: varchar(255)
abbreviation: varchar(16)
city: varchar(255)
state_province: varchar(255)
country: varchar(255)
PRIMARY
event
channel_id: int(10)
event_id: varchar(30)
jn_username: varchar(32)
date: timestamp
PRIMARY
idx_creator_username
idx_publication_date
channel
channel_id: int(10)
project_jn_name: varchar(50)
channel_name: varchar(30)
title: varchar(255)
category: enum ('COMMIT','CUS...
PRIMARY
idx_channel_category
idx_fk_channel_project_jn_name
workpace_x_project
workspace_id: mediumint(8)
project_jn_name: varchar(50)
summary: varchar(64)
PRIMARY
idx_fk_workspace_x_project_jn_n...
student
user_id: int(10)
institution_id: smallint(5)
PRIMARY
idx_fk_student_institution_id
project
project_jn_name: varchar(50)
summary: varchar(64)
PRIMARY
dispute
dispute_id: smallint(5)
title: varchar(64)
date: date
project_jn_name: varchar(50)
PRIMARY
idx_fk_dispute_project_jn_name
dispute_entry
entry_id: int(10)
notes: varchar(255)
dispute_id: smallint(5)
date: timestamp
PRIMARY
idx_fk_entry_dispute_id
student_x_project
user_id: int(10)
project_jn_name: varchar(50)
is_leader: tinyint(1)
PRIMARY
idx_fk_project_x_project_jn_name
Infinity Metrics Database Schema

6. COMPETITIVE ANALYSIS

Despite its potential wide-ranging applications, research on the automation of collecting participation metrics of java.net projects by the community at large has yielded little significant results. In fact, GlassFish, one of the largest and most widely used projects on java.net[1] has no automated participation metrics system[2]. While StatsWidgets provides the limited functionality of outputting delimited text files of a given RSS feed (or a set of them) from public and/or private projects, this implementation provides little usability for the intended audience of our system because it would require instructors to set up, configure and maintain a database to handle the output files generated by StatsWidgets. Since the requirements explicitly state that users should have no such prior knowledge of database systems, StatsWidgets proves inadequate in this regard. In addition, not all metrics generated by a project are traceable using RSS, and so the need for a tool that will collect metrics on these non-RSS participation metrics is essential. We aim to automate not only the retrieval of the RSS feeds, but the storage in a database that will enable presentation in rich and meaningful ways to the instructor.

Notwithstanding StatsWidgets' differing intended audience, the similarity of its core functionality makes it a competitor to our system. Following is a side-by-side architectural and functional comparison of the two systems.

Competitive Analysis – Financial and Technical					
Product	**Open-Source**	**Cost**	**Platform**	**Persistence**	**Modules**
∞Metrics	Yes	Free	PHP 5.2	MySQL	PHP, cURL
Stats Widgets	Yes	Free	JDK 2	Text Files	Kosuke Scraper
Project Metrics	No	Free	~	~	~

Competitive Analysis – Functional Requirements			
Functionality	**∞Metrics**	**Stats Widgets**	**Project Metrics**
Provides Metrics for Public projects	✓	✓	✓
Provides Metrics for Private Projects	✓	✓	
Store Metrics in a Database System	✓		✓
Provides Web Interface for Users	✓		
Provides Tabular Data to Users	✓	✓	✓
Provides Graphical Data to Users	✓		✓
Export Delimited Data	✓	✓	✓
Saves Configuration	✓		

[1] http://community.java.net/projects/top.csp

[2] http://wiki.glassfish.java.net/Wiki.jsp?page=CommunityStatistics

7. RISK MANAGEMENT

The risk management is an important tool to identify possible problems and how to mitigate them. It is used by the entire team, especially by the project leaders. The Risk Analysis subsection describes each of the identified risks for this project, defining its scope and type. Along with that, the subsection Risk Monitoring describes probabilities of such risks to take place, indicators of that show when the risk has emerged the strategies on how to mitigate each of them.

7.1. RISK ANALYSIS

Risk	Risk Type	Risk Scope	Description
Team Member Skill Set	People	Project/Product	Team members do not possess the programming skill set to adequately complete the project.
Scheduling Difficulties	Organizational	Project	Time zone differences, schedule overload and conflicting schedules among team members will occur
Legal Liabilities	Business Environment	Business	The Terms of Use in java.net do not permit automated or scripted access to the site.
Underestimating Project Size	Estimation	Project/Product	The team will underestimate the size of the project during the planning stage
Requirement Changes	Requirements Risk	Project/Product	The requirements will change after the requirement planning phase
Development IDE Underperformance	Tools	Project	The tools used to develop the software will not perform as expected (i.e. NetBeans 6.5 is a Beta release)
java.net updates its software platform	Technology	Project/Product	java.net updates the version of CEE to a new version that changes the layout of the HTML at the sections where the personal agent uses.
Team Member Drop-out	People	Project	Team members will withdraw from the class and thus out of the project
Team Member Lack of Participation	People	Project	Team members will not participate sufficiently to fulfill the project's requirements.
Deployment Platform Malfunction	Technology	Project	The deployment environment (T2000) server will malfunction and render our system inoperable or inaccessible.
Support software underperformance	Tools	Project	The distributed virtual development environment will not perform as expected (i.e. Virtual Machine)
Underestimating Development Time of Requirements	Estimation	Project/Product	The time needed to develop and implement the requirement specifications will exceed the length of the semester.
Group Dynamics Conflicts or Failure	People	Project	The team will reach an impasse or a irreconcilable differences which will hinder the progress of the project.
CASE Tool Underperformance	Tools	Project	The code generated by CASE tools is insufficient or inaccurate

7.2. RISK MONITORING

Risk	Probability	Effects	Potential Indicators	Mitigation Strategy
Team Member Skill Set	High	Serious	- Constant build breaks, inability to meet; - Implementation deadlines	- Have team members follow tutorials and training sessions to enhance skill set.
Scheduling Difficulties	High	Serious	- Missed meetings - Poor communication, - poor performance and missed deadlines in project & other activities	- Increase communication; - Allow for flexibility in case of conflict; - Provide time management tools for team members to fulfill all demands on time effectively and efficiently.
Legal Liabilities	Low	Armageddon	Legal suits are filed, cease & desist orders, etc.	- Research legal Terms of Use regarding use & access policies on java.net.
Underestimating Project Size	Moderate	Serious	Issues queue accumulate, and team seems unable to grasp scope of project design.	- Be realistic about time allowance for development; - Use good planning strategies.
Requirement Changes	Moderate	Serious	- Multiple revisions to requirements due to customer requests; - re-writing previously developed components to accommodate new requirements.	- Perform requirement validation; - Verification techniques along with traceability.
Development IDE Underperformance	Moderate	Serious	Constant application crashes, loss of unsaved code.	- Have version control in case of file corruption; - Stay up to date on NetBeans released/bug fixes and periodically download last build of IDE.
Technology Change	Low	Armageddon	Personal Agent unable to retrieve RSS feeds, project sets, or metrics	Stay up to date with java.net forums and news.
Team Member Drop-out	Moderate	Tolerable	Team members no longer in class	Build strong team cohesiveness.
Team Member Lack of Participation	Moderate	Tolerable	Lack of participation, missed meetings, lack of communication	Encourage team members to contribute to the best of their ability and allow flexibility in contribution levels.
Deployment Platform Malfunction	Low	Armageddon	System is inoperable or offline.	Have a set of policies & procedures with contact information for technical support team responsible for maintaining the T2000 server.
Support software underperformance	Moderate	Serious	Inability to use virtual environment.	Test environment thoroughly and stabilize environment before distributing to team.
Underestimating Development Time of Requirements	Moderate	Serious	End of semester with no complete implementation	Carefully plan execution time for each requirement. Prioritize requirements. Keep time constrains in mind.
Group Dynamics Conflicts or Failure	Low	Serious	Team disputes and communication ceases.	Provide a positive communication environment, morale-boosting techniques and conflict resolution.
CASE Tool Underperformance	Low	Insignificant	Code breaks and behaves unexpectedly.	Test thoroughly code generated by CASE tools and allow sufficient time for re-writes.

8. RESOURCES ALLOCATION

Since the team will be following an Agile/Scrum process, the team will be cross-functional, with all the members being developers, and the team leaders as product owner. More detail is presented in the following table, as well as the roles each of us will be playing in the team.

Team Member	Role
Marcello de Sales	Team Leader, Scrum Master, Product Owner, Cross-functional Developer
Andres Ardila	Product Owner, Cross-functional Developer
Brett Fisher	Cross-functional Developer
Gurdeep Singh	Cross-functional Developer
Marilyne Mendolla	Cross-functional Developer
Mateo Mejia	Cross-functional Developer