

Indian Institute of Technology, Kharagpur

Department of Computer Science and Engineering

Software Engineering (CS 20006), Spring 2015-16

Assignment Management System (AMS)

Req. Spec., and Outline of Analysis, Design, and Testplan

1. A course on Software Construction in IIT wants to manage the assignments to *Students*, the submissions of assignments by *Students*, and the evaluations of the submissions through an **Assignment Management System (AMS)**. The requirement specifications for the system are as follows:

(a) Participants of the course are:

- One *Instructor*. She / he is identified by an *Employee Code*, and has *Name*, *Email* and *Mobile Number*.
- 5 or more *Teaching Assistants (TA)*. Every *TA* is identified by a *Roll No*, and has *Name*, *Email* and *Mobile Number*.
- 100 or more *Students*. Every *Student* is identified by a *Roll No*, and has *Name*, *Department*, *Hall*, *Email* and *Mobile Number*.

(b) The responsibilities of the *Instructor* include:

- *Set-up*: Design and set up the assignments, decide on the date that an assignment is to be assigned to the *Students*, and on the date of submission for the assignment. Also decide on a *Coordinating TA* for an assignment.
- *Allocation*: Allocate *Students* to *TA*. Every *Student* has one allocated *TA*.
- *Approve*: Approve / Disapprove extensions for submissions beyond the submission date and decide on the penalty.
- *Compilation*: Compile the evaluations (as performed by the *TAs*) and publish the final scores of every assignments on **AMS**.

(c) The responsibilities of a *TA* include:

- *Mentor*: Mentor and manage the *Students* allocated to the *TA*. She / he is the primary support for the allocated *Students* before she / he should approach the *Instructor*.
- *Upload*: The *Coordinating TA* of an assignment uploads an assignment as set by the *Instructor* and sets up the required *Assign* and *Submission Dates* on the **AMS**.
- *Download*: Download from the **AMS** and archive the submissions for the *Students* allocated to the *TA*. This needs to be done after the submission date in every assignment.
- *Evaluate*: Peruse the submissions, discuss with the respective *Students* for clarifications, take demonstrations (if relevant), and evaluate.
- *Report*: Report the evaluations to *Instructor*. Requests for submission date extension with full credit, on grounds of medical or personal exigencies, are also to be reported after proper authentication. Further, *TAs* are responsible for reporting Disciplinary actions (like plagiarism), if any.

(d) The responsibilities of a *Student* include:

- *Perform*: Complete every assignment within the submission date and submit to **AMS**.
- *Appeal*: Appeal to the *Instructor* for permission for special submission without penalty. Every appeal needs to go through the respective *TA* and must be authenticated by her / him.
- *Demonstrate*: Discuss and demonstrate the submission to the allocated *TA*.

(e) An *Assignment*:

- *Ownership*: Is to be completed individually by every *Student*.

- *Type*: Is one the following types:
 - *Programming Assignment*: The assignment has one *Problem* that asks to write a single program, specifies the language for coding (like C / C++ / Java), and has a single submission date.
 - *Systems Assignment*: The assignment asks to develop a System. It has one or more *Problem/s*. Each *Problem* is about a component module that builds up the System and has a separate submission date. No coding language is specified for such assignments.
- *Assign Date*: Has an *Assign Date* on which it is given to the *Students*.
- *Submission Date*: Has a *Submission Date* by which it is to be completed and submitted. For a *Systems Assignment* every constituent *Problem* has a separate *Submission Date* but a common *Assign Date*.
- *Marks*: Every *Problem* in an assignment has specified maximum marks.
- *Coordinator*: By turn a *TA* is allocated by the *Instructor* as a coordinator for an assignment. She / he manages the upload, and the dates for the assignment.

(f) A *Submission* :

- *Action*: Is performed for an *Assignment*, by a *Student* on a *Date*.
- *Valid*: Is *valid* if it is submitted within the *Submission Date* of the corresponding assignment. *Valid* submissions carry full credit.
- *Late*: Is *late* if it is done within 3 days from the *Submission Date* of the corresponding assignment. *Late* submissions carry 10% penalty.
- *Special*: Is *special* if it is done within 7 days from the *Submission Date* of the corresponding assignment. A *Student* needs to *appeal* for *Special* submissions to the *Instructor* through the *TA*. *Special* submissions are allowed on grounds of medical or personal exigency and carry no penalty.
- *Invalid*: Is *invalid* if it is not submitted within 3 days from the the *Submission Date* of the corresponding assignment and has not been granted extension as a *Special Submission*. *Invalid* submissions carry zero credit.

No submission is allowed before the *Assign Date* of the submission.

(g) The *Work flow* in the course is as follows:

- The *Instructor* designs an *Assignment* and mails it to the *Coordinating TA*.
- The *Coordinating TA* uploads the *Assignment* to the system and sets the corresponding *Assign* and *Submission Dates*.
- Once an *Assignment* has been set, **AMS** sends an email notification to all *Students*, *TAs* and the *Instructor* about the *Assignment*.
- The *Students* completes the *Assignment* and uploads the solution to **AMS**.
- Once the *Submission Dates* for an *Assignment* is over, **AMS** sends an email notification each to every *TA* with the respective list of completed submissions (*Students*' roll numbers are mentioned) by the *Students* allocated to the *TA*. Respective students are carbon-copied on the email. The *Instructor* is copied on every notification.
- Every *TA* downloads the respective submissions, discusses with the *Students*, checks demonstrations, and evaluates the solution.
- Once a *TA* completes her / his evaluations for an *Assignment*, **AMS** sends an email notification to the *Instructor* with the *TA* on the carbon-copy. *TA* should complete the evaluations after 4 days from the *Submission Date* and before 10 days from it.
- The *Instructor* on receipt of completion report from all the *TAs* compiles the scores for an *Assignment* and publishes on the **AMS**.
- If a *Student* misses to submit by the *Submission Date*, but submits within 3 days from that, she / he is penalized by 10%. On such submissions, **AMS** sends an email notification to the *TA* and the *TA* would similarly evaluate the submission and report to the *Instructor*.
- A *Student* may appeal for an extension on grounds of medical or personal exigency. The *TA* would scrutinize the appeal and report to the *Instructor*, if authentic. **AMS** will send an email notification for the same. As the *Instructor* approves or disapproves the appeal, accordingly the *Student* and the *TA* are notified. If the appeal is approved, the process of submission and evaluation is followed.

You have been assigned as the software engineer for the **AMS**. You are required to analyse the specifications, design the system (using UML and DP) and also prepare the test plan. Answer the following questions in this background.

- Identify the actions in **AMS** and design the Use-Case Diagrams for the actions. Identify the actors, specify their types, and mark the relationships between the actors. Show the <<include>>, <<extend>>, and generalization relationships of the use-cases. [4+4=8]
- Design Class Diagrams for *Assignments & Submissions*. Show the attributes and operations with their associated properties. Highlight specialization hierarchies, if any. [4+4=8]
- Complete the Class Diagram of **AMS** showing all other classes (in addition to Question 1b) by their respective brief Diagrams (with name and key attributes). For the entire collection of classes (that is, including *Assignments* and *Submissions*) show the associations, aggregations / compositions, generalization / specialization, and abstract / concrete etc. [6]
- Design the State-Chart Diagrams for *Assignments* and *Submissions*. [2+2=4]
- Design Sequence Diagrams for the actions in **AMS** as specified in the Work flow. [10]
- Identify and justify the use of Iterator, Singleton and Command DPs in **AMS**. [2*3=6]
- Prepare a test plan for **AMS** to perform black-box tests. Clearly mark the scenarios for Unit Testing and Integration Testing. [4+4=8]

Answers:

Part 1a: The Use-cases of **AMS** are:



Part 1b: Class diagrams for *Assignment* and *Submission*:

An Assignment can contain one or more Problems. Hence we need an auxiliary **Problem** class to define *Assignment* class.

Problem			Remarks
-	id	: String	Auto-generated
-	assignment	: String	Set by Create(). assignment = Assignment.id
-	submissionDate	: Date	Set by Create()
-	marks	: Int	Set by Create()
-	body	: Text	Set by Create()
-	/status	: Bool[3]	Set by Create(). status[0] = True, rest False. Changes by TRIGGER
+	Create(assignment: String, submissionDate: Date, marks: Int, body: Text, language: String = ""): Problem *		Constructor for Problem
+	Display(): void		Display all fields

The status values are derived as:

Index	Status	Remarks
0	new	The problem is created as a part of assignment that contains it
1	open	The problem is open, if today ≥ assignment.assignDate and today ≤ submissionDate
2	closed	The problem is closed, if today > submissionDate

Assignment { <i>Abstract</i> }	<i>Remarks</i>
<ul style="list-style-type: none"> – id : String – setter : String – assignDate : Date – coordinator : String – nProblems : Int – problems : String[1..nProblems] – status : Bool[4] 	Auto-generated Set by Create(). setter = Instructor.eCode Set by Create() Set by Create(). coordinator = TA.roll Set by Create(). Number of problems in the assignment Set by Create(). problem = Problem.id Set by Create(). status [0] = True, rest False. Changes by state-chart
+ <u>Create(setter: String, assignDate: Date, coordinator: TA, nProblems: Int, problems: String[]): Assignment *</u> + Display(): void + Upload(coordinator: String): void	Create factory for Assignment Display all fields Upload for assigning to students. coordinator = TA.roll

The status values are derived as:

Index	Status	Remarks
0	new	The assignment has been created by Instructor and mailed to Coordinating TA
1	uploaded	The assignment has been uploaded by Coordinating TA
2	open	The assignment is open, if at least one of the Problems is open
3	closed	The assignment is closed, if all the Problems are closed

There are two specializations – **ProgrammingAssignment** and **SystemsAssignment** – of *Assignment*. These are concrete classes.

ProgrammingAssignment Base Assignment	<i>Remarks</i>
<ul style="list-style-type: none"> – language : String 	nProblems = 1 by construction
+ <u>Create(setter: String, assignDate: Date, coordinator: TA, nProblems: Int, problems: String[], language: String): Assignment *</u>	Set by Create(). One of {C, C++, Java}
	Constructor for ProgrammingAssignment

SystemsAssignment Base Assignment	<i>Remarks</i>
+ <u>Create(setter: String, assignDate: Date, coordinator: TA, nProblems: Int, problems: String[]): Assignment *</u>	nProblems \geq 1 by construction
	Constructor for SystemsAssignment

Submission { Abstract }	Remarks
<ul style="list-style-type: none"> – id : String – student : String – problem : String – date : Date {optional} – body : Text {optional} – marks : Int – lateSubmissionAppeal : Bool – lateSubmissionReason : String {optional} – lateSubmissionForward : Bool – lateSubmissionApproval : Bool – /credit : Int – /status : Bool[8] 	<p>Auto-generated</p> <p>Set by Create(). student = Student.roll</p> <p>Set by Create(). problem = Problem.id</p> <p>The date of problem submission. Set to null by Create(), finalized by FinalizeSubmission()</p> <p>The solution as submitted. Set by Create(), edited by EditSubmission()</p> <p>Set to 0 by Create(). Set by EvalAndReport()</p> <p>Set to False by Create(). Set by AppealLateSubmission()</p> <p>Set to null by Create(). Set by AppealLateSubmission()</p> <p>Set to False by Create(). Set by EvalAndReport()</p> <p>Set to False by Create(). Set by AllowLateSubmission()</p> <p>Percentage of credit (max = 100). Set to 0 by Create(). Changes by state-chart</p> <p>Set by Create(). status[0] = True, rest False. Changes by state-chart</p>
<ul style="list-style-type: none"> + <u>Create(student: String, problem: String, body: Text = ""): Submission *</u> + Display(): void + EditSubmission(): void + FinalizeSubmission(): void + AppealLateSubmission(reason: String): void + EvalAndReport(marks: Int, lateSubmission: Bool): void + AllowLateSubmission(): void 	<p>Constructor for Submission</p> <p>Display all fields</p> <p>body of Submission edited by student</p> <p>Submission done (finalized) by student – no more edits allowed</p> <p>student appeals for late submission with reason</p> <p>TA of student reports the evaluation and / or response to appeal for late submission with reason</p> <p>Instructor allows late submission</p>

The status and credit values are derived as:

Index	Status	Remarks	Credit
0	draft	The submission has been created but not submitted	0
1	appealed	lateSubmissionAppeal lateSubmissionForward lateSubmissionApproval	0
2	forwarded	True False False	0
3	approved	True True False	0
4	valid	True True True	0
5	late	The submission has been done and date ≤ problem.submissionDate	100
6	special	The submission has been done, date > problem.submissionDate, date ≤ problem.submissionDate + 3, and lateSubmissionApproval = False	90
7	invalid	The submission has been done, date > problem.submissionDate, date ≤ problem.submissionDate + 7, and lateSubmissionApproval = True	100
		The submission has been done, date > problem.submissionDate + 3 and lateSubmissionApproval = False	0

Part 1c: The Class diagram for **AMS**:



Part 1d: The **Problem** has the following states that change according to the state-chart:



Present State	Action	Next State	Actor	Status			Method & Condition
				new	open	closed	
null	Create	New	Instructor	True	False	False	Create() & today < assignment.assignDate
New	TRIGGER	Open	System	False	True	False	today ≥ assignment.assignDate & today ≤ submissionDate
Open	TRIGGER	Closed	System	False	False	True	today > submissionDate

The **Assignment** has the following states that change according to the state-chart:



Present State	Action	Next State	Actor	Status				Method & Condition
				new	uploaded	open	closed	
null	Create	New	Instructor	True	False	False	False	Create()
New	Upload	Uploaded	TA	False	True	False	False	Upload()
Uploaded	TRIGGER	Open	System	False	False	True	False	At least one problem is in open state
Open	TRIGGER	Closed	System	False	False	False	True	All problems are in closed state

The *Submission* has the following states that change according to the state-chart:



Condition Name	Condition Expression
<i>Cond 1</i>	$\text{date} \leq \text{problem.submissionDate}$
<i>Cond 2</i>	$\text{date} > \text{problem.submissionDate} \ \& \ \text{date} \leq \text{problem.submissionDate} + 3$
<i>Cond 3</i>	$\text{date} > \text{problem.submissionDate} + 3$
<i>Cond 4</i>	$\text{date} > \text{problem.submissionDate} \ \& \ \text{date} \leq \text{problem.submissionDate} + 7$
<i>Cond 5</i>	$\text{date} > \text{problem.submissionDate} + 7$

Present	Action	Next	Actor	Status								Method & Condition
				draft	appealed	forwarded	approved	valid	late	special	invalid	
State		State										
Create												
null	Create	New	Student	T	F	F	F	F	F	F	F	Create()
Appeal for Extension												
New	Appeal	Appealed	Student	T	T	F	F	F	F	F	F	AppealLateSubmission()
Appealed	Forward	Forwarded	TA	T	T	T	F	F	F	F	F	EvalAndReport(True)
Appealed	Forward	Appealed	TA	T	T	F	F	F	F	F	F	EvalAndReport(False)
Forwarded	Approve	Approved	Instructor	T	T	T	T	F	F	F	F	AllowLateSubmission(True)
Forwarded	Approve	Forwarded	Instructor	T	T	T	F	F	F	F	F	AllowLateSubmission(False)
Edit												
New	Edit	New	Student	T	F	F	F	F	F	F	F	EditSubmission()
Appealed	Edit	Appealed	Student	T	T	F	F	F	F	F	F	EditSubmission()
Forwarded	Edit	Forwarded	Student	T	T	T	F	F	F	F	F	EditSubmission()
Approved	Edit	Approved	Student	T	T	T	T	F	F	F	F	EditSubmission()
Submission												
New	Submit	Valid	Student	F	F	F	F	T	F	F	F	FinalizeSubmission() & date ≤ problem.submissionDate
New	Submit	Late	Student	F	F	F	F	F	T	F	F	FinalizeSubmission() & date > problem.submissionDate & date ≤ problem.submissionDate + 3
New	Submit	Invalid	Student	F	F	F	F	F	F	F	T	FinalizeSubmission() & date > problem.submissionDate + 3
Appealed	Submit	Valid	Student	F	T	F	F	T	F	F	F	FinalizeSubmission() & date ≤ problem.submissionDate
Appealed	Submit	Late	Student	F	T	F	F	F	T	F	F	FinalizeSubmission() & date > problem.submissionDate & date ≤ problem.submissionDate + 3
Appealed	Submit	Invalid	Student	F	T	F	F	F	F	F	T	FinalizeSubmission() & date > problem.submissionDate + 3
Forwarded	Submit	Valid	Student	F	T	T	F	T	F	F	F	FinalizeSubmission() & date ≤ problem.submissionDate
Forwarded	Submit	Late	Student	F	T	T	F	F	T	F	F	FinalizeSubmission() & date > problem.submissionDate & date ≤ problem.submissionDate + 3
Forwarded	Submit	Invalid	Student	F	T	T	F	F	F	F	T	FinalizeSubmission() & date > problem.submissionDate + 3
Approved	Submit	Valid	Student	F	T	T	T	T	F	F	F	FinalizeSubmission() & date ≤ problem.submissionDate
Approved	Submit	Special	Student	F	T	T	T	F	F	T	F	FinalizeSubmission() & date > problem.submissionDate & date ≤ problem.submissionDate + 7
Approved	Submit	Invalid	Student	F	T	T	T	F	F	F	T	FinalizeSubmission() & date > problem.submissionDate + 7
Timeout												
New	TRIGGER	Invalid	System	T	F	F	F	F	F	F	T	date > problem.submissionDate + 3
Appealed	TRIGGER	Invalid	System	T	T	F	F	F	F	F	T	date > problem.submissionDate + 3
Forwarded	TRIGGER	Invalid	System	T	T	T	F	F	F	F	T	date > problem.submissionDate + 3
Approved	TRIGGER	Invalid	System	T	T	T	T	F	F	F	T	date > problem.submissionDate + 7

Part 1e: The Sequence Diagram for major activities in **AMS** are:



Part 1f: The Design Patterns in **AMS** are:

Situation	Pattern	Marks
Generate assignment objects of appropriate specialization.	Abstract Factory, Factory Method	0 mark – Not covered in class
Browse / Display lists of assignments, problems in an assignment, students, TAs etc.	Iterator	2 marks
There is only one instructor	Singleton	2 marks
Instructor asks Coordinating TA to upload assignment, students appeal to TAs, TAs forward to instructor, etc. Asking for the action and the execution of the action are separated in time as is needed in Command Pattern.	Command	2 marks

Part 1g:

(a) **Unit Testing:**

i. **Testing for Classes:**

For every class tests need to be performed for:

- Constructor / Destructor / Copy Constructor / Copy Assignment Operator etc
- Member Functions
- Static Member Functions

In addition, some important scenarios for every class must be tested:

- Class Instructor: The class must be a singleton.
- Class TA
- Class Student
- Class Problem
 - Trigger (Time) based tests – Before Assign Date, Between Assign Date and Submission Date, After Submission Date
- Class Assignment
 - Test for Specializations
 -
- Class Submission

ii. **Testing Sub-Systems:**

(b) **Integration Testing:**

i. Login to **AMS**