Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Elisabetta Di Nitto, Raffaela Mirandola

20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Software Engineering II

January 29 2016

Last Name
First Name
Id number (Matricola)

Note

1. The exam is not valid if you don't fill in the above data.
2. Write your answers on these pages. Extra sheets will be ignored. You may use a pencil.
3. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden.
4. You cannot keep a copy of the exam when you leave the room.

Question 1 Alloy (7 points)
We want to model the behavior of a component that migrates data from a source storage to a destination storage. Data is divided in chunks that are migrated independently from each other and possibly in parallel to improve performance (independence and parallelization of chunk migration is not to be handled in this exercise).
Each chunk can be in one of the following states: ready to be migrated, under migration, migrated.
Of course, the whole migration ends when all chunks in a data set are in the migrated state.
Queries are related to specific chunks of data in a one-to-many relation. These queries can be potentially executed on the chunks when these last ones are still to be migrated or when they are migrated, while they have to be postponed when one of the involved chunks is under migration.

Define an Alloy model for the above problem. Focus on the definition of the signatures and facts that describe the constraints described above.

Solution
```
sig Storage {}
sig Data {}

abstract sig ChunkStatus {}
one sig Ready extends ChunkStatus {}
one sig UnderMigration extends ChunkStatus {}
one sig Migrated extends ChunkStatus {}

abstract sig QueryStatus {}
one sig Executable extends QueryStatus {}
one sig Postponed extends QueryStatus {}

sig MigrationStatus {}
one sig Stopped extends MigrationStatus {}

one sig MigrationSystem {
  status: MigrationStatus,
  source: Storage,
  destination: Storage,
  dataSet: set Data
}

sig Chunk {
 status: ChunkStatus,
 data: set Data
} {status = Ready || status = UnderMigration || status = Migrated}

sig Query {
  status: QueryStatus,
  chunks: set Chunk
}

//Data in chunks belong to the set of data managed by the migration system
fact allDataInChunk {
  all d: Data | d in MigrationSystem.dataSet and d in Chunk.data
}

//Different chunks include different sets of data
```

```
fact disjointChunks {
all disjoint c1, c2: Chunk | c1.data & c2.data = none
}

//If the migration is terminated, all corresponding chunks are in the Migrated state
fact migrationEnd {
  all c: Chunk | (c.data in MigrationSystem.dataSet and MigrationSystem.status = Stopped)
implies c.status = Migrated
}

//if the status of a chunk is UnderMigration, the status of a query working on it should be Postponed
fact queryPostponedUnderMigration {
  all q: Query, c: Chunk | c in q.chunks and c.status = UnderMigration
implies q.status = Postponed
}

pred show{}

run show
```

Questions 2 Planning (5 points)
Referring to the data migration system mentioned in the previous exercise, identify two risks that you consider particularly critical and provide a motivation for their criticality. For each risk analyze their probability of occurrence (you can specify explicitly the circumstances to which you are referring when you evaluate this probability) and identify a contingency plan.

Solution
The ones below are two examples of risks we may envisage. They are not the only possible answer to this question.

Risk 1: the interplay between migration and query execution is not working as it should be. For instance, there might be race conditions that take the system in a state where a chunk is being migrated while subject to a query.
This risk is critical as the system shows a high degree of parallelism. The probability of occurrence depends on the ability of the development team to cope with this issue. If we suppose to have a team composed of experienced persons, we can assume that this probability is low.
As a contingency plan, to avoid the occurrence of this risk we will allocate some effort of an experienced consultant who will review the work done by the others at some critical milestones, for instance, at the end of the design phase, during the system testing phase, before any major maintenance activity on the system.

Risk 2: fault tolerance of migration becomes a key issue for our customer.
A case in which this risk may happen is when the number and size of data to migrate grows significantly. This implies that the time needed for migration may grow up to several hours. Thus, the probability of faults of the migration system increases and therefore the customer becomes very sensitive to fault tolerance.
This risk is critical as coping with fault tolerance can have a strong impact on the architecture of the whole migration system. The probability of occurrence of this risk varies depending on the application domain in which the migration system is going to be used. As from the text of the exercise we do not have specific information on this point, we assume that it is low.
As a contingency plan, we get the advice of a fault tolerance expert on the potential weaknesses of our architecture with respect to fault tolerance and we identify a plan to incrementally extend the system to cope with this issue.

Question 3 design (5 points)
Consider a simple project management tool for a company. The application should:
1. Handle information about projects (name, purpose, budget, tasks) and people (name, skills, employee number, role, department, allocated projects, allocated tasks).
2. Allow people with role "project manager" to allocate other people to projects and tasks.
3. Allow people to visualize the tasks they are working on and input the level of accomplishment of these tasks.
4. Compute the set of people allocated to a specific project/task.

In the hypothesis of adopting the JEE framework:
A) Identify the JEE components to be used in this case providing a justification for your choice.
B) Define the main relationships between components.
C) Define the interfaces of the JEE components, skipping the definition of the obvious set/get interfaces offered by entity beans.

You can add any hypothesis you think is important to consider in this case, provided that you do so explicitly.

Solution
Parts A and B
We can implement the project management tool as a multi-tiers web application.
The *client tier* contains the web browser.
The *web tier* is built using the JSF technology and is based on the MVC pattern. In particular, to fulfill the requirements explicitly defined in the exercise, at least two JSF pages are defined, one for the login (index.xhtml) and another for all other operations in the system (home.xhtml). This second page includes a part which is rendered only if the logged user is the project manager to allocate people to projects and tasks. The JSF pages communicate with the business logic by using three request scoped managed beans (LoginBean, AllocatePeopleBean, UpdateLevelBean).
The *business logic* is implemented by a session bean called RequestManager (visible from the JSF pages). This manages the three operations defined in the exercise and interacts with the database to persist data.
The *persistent layer* is represented by three entities, Employee, Project and Task. Employee and Task and Employee and Project are related through many-to-many relationships, while Project and Task show a one-to-many relation.
The application server manages the authentication and the transactions to the database. Moreover, it verifies constraints on method calls (for instance, only the project manager can call the allocateEmployeeToTask method offered by the RequestManager session bean).
Part C
The interface of the *RequestManager* contains the following methods:

```
@RolesAllowed(Roles.PM)
public void allocateEmployeeToTask(Long employeeNumber, Long taskId)

@RolesAllowed(Roles.PM)
public void allocateEmployeeToProject(Long employeeNumber, Long projectId)

public void setAccomplishmentLevelToTask(Long taskId, Long level)

public Set<Task> getMyTasks()

//This last method is used by the home.xhtml to distinguish between project managers and others
public Employee getLoggedUser()
```

The interface of *Employee* contains:

public String getName()
public boolean taskIsAllocated(Long taskId)
public boolean isProjectManager()
public Set<Task> getTasks()

The interface of *Project* contains:
public void allocateEmployee(Employee employee)
public Long getId()

The interface of *Task* contains:
public void allocateEmployee(Employee employee)
public void setAccomplishmentLevel(Long level)
public Long getId()
public Project getProject()
public Long getAccomplishmentLevel()

The interfaces of the three managed beans contain setter and getter methods plus the following:
AllocateProjectBean
//This calls the allocateEmployeeToProject method by the RequestManager
public void save()

UpdateLevelBean
//This calls the setAccomplishmentLevelToTask method by the RequestManager
public void save()

LoginBean
public String login()
public String logout()
The two methods above use authentication methods provided by java enterprise.

You can refer to the sketch of code in **pmt.zip** to get more details (which are not required by the exercise).

Question 4 testing (5points)
Referring to the project management tool analysed in Question 3, identify the three highest priority acceptance test cases for this tool focusing on functional aspects. While defining these cases specify the preconditions (if any) that should be true before executing the tests, the inputs provided during the test, and the expected outputs. Finally, explain why you think the selected tests are important for the specific system being considered.

Examples of possible tests

Test1- A project manager assigns a person to a project/task

Precondition: user logged-in, user is a PM
Input: a person and a project or a task
Output: the person is assigned to the project or to the task. If the person executes Test2 at this point, should see the new project or task in his/her user interface.

Test2- An employee visualizes information about projects and tasks:

Precondition: user logged-in, project and tasks exist in the DB
Input: project and/or tasks ID
Output: visualization of the information about the requested project and/or tasks assigned to the user

Test3- An employee updates the level of accomplishment for a task:

Precondition: user user logged-in, project, tasks exist in the DB, and the user is assigned to a task
Input: the task and the level of accomplishment
Output: the level of accomplishment for the task is changed.

The above tests are important because they allow us to test the main functions offered by the system. In particular, referring to the numbered list in Question 3: Test1 allows us to check point 2 and part of point 1. Test2 allows us to check point 4 and part of point 1. Finally, Test3 allows us to check point 3 and part of point 1.