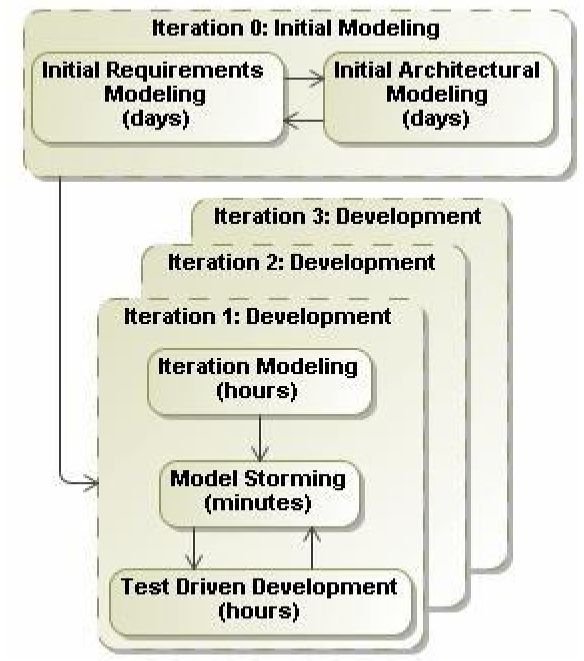
Project Guideline

1 Introduction

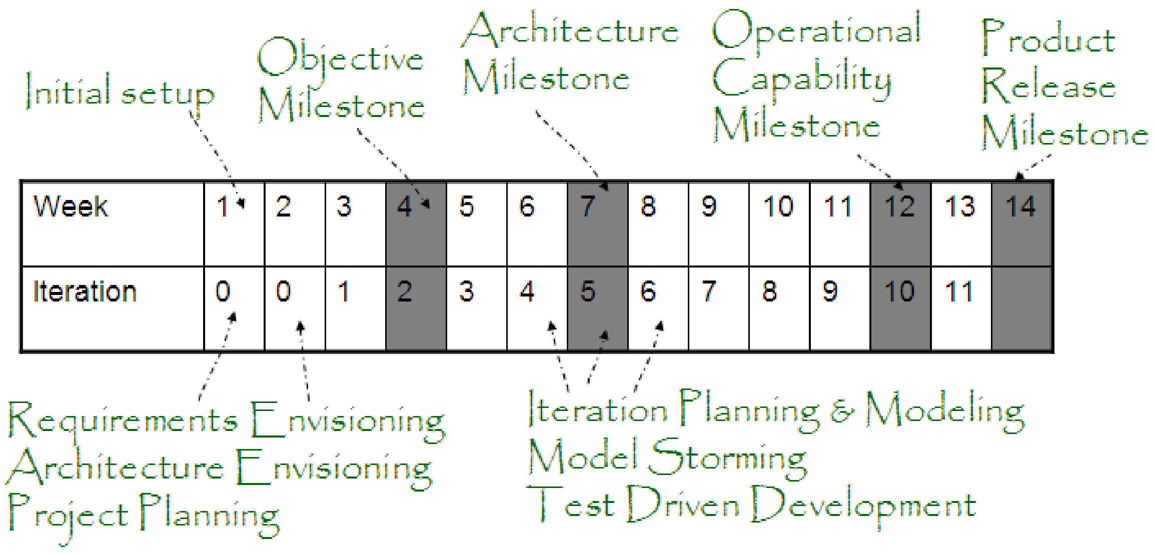
The project must offer a solution to the following problem statement.

The project team must follow the Agile Model Driven Development (AMDD)[1] methodology as described below. Figure 1 shows a high-level lifecycle for AMDD for the release of a system.



Iteration 0: Initial Modeling
Initial Requirements Modeling (days)
Initial Architectural Modeling (days)
Iteration 3: Development
Iteration 2: Development
Iteration 1: Development
Iteration Modeling (hours)
Model Storming (minutes)
Test Driven Development (hours)

Figure 1 AMDD Activities



Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Iteration	0	0	1	2	3	4	5	6	7	8	9	10	11	

Figure 2 Development Lifecycle: iterations and milestones

Figure 2 depicts how of the AMDD activities and the project milestones will be

[1] Scott W. Ambler. *Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development*. http://www.agilemodeling.com/essays/amdd.htm

planned during the development lifecycle.

2 Initial Setup

The initial setup and planning activities will be performed during the first lab meeting. **The following roles must be established** and written into the project vision and project plan documents:

- *Stakeholder* – interest groups whose needs must be satisfied by the project (e.g. lab teacher);
- *Project Manager* – leads the planning of the project, coordinates interactions with the stakeholders, and keeps the project team focused on meeting the project objectives.
- *Developer* – develops a part of the system, including designing it to fit into the architecture, possibly prototyping the user-interface, and then implementing, unit-testing, and integrating the components that are part of the solution.
- *Team Leader* – has the developer responsibilities and is also responsible for a subsystem or a team of developers.

For more details about the above project roles see OpenUP[2] (Open Unified Process) documentation.

The project artifacts (documents & code) will be managed using a SVN[3] Repository (version control system). The repositories will be accessible as follows https://www.scs.ubbcluj.ro/svn/gr221a, https://www.scs.ubbcluj.ro/svn/gr221a, etc. The teachers must communicate the project members to ilazar@cs.ubbcluj.ro in order to configure developer access rights to their repositories. Users will be able to access the svn repositories using their SCS usernames and passwords.

The Bugzilla[4] defect tracking system will be used to manage the project tasks and bugs. A separate guide will describe how to access and use the system.

3 Initial Modeling

The initial modeling iteration is performed during the first week of a project, or the first two weeks for large projects. The scope of this iteration is to:

- Identify the high-level scope of the project;
- Identify initial requirements stack;
- Identify an architectural vision.

3.1 Usage Models

These models refer to "how users will work with your system". A feature list is given in problem_statement.

(A) [Requirements list] Write a requirements list based on the feature list above

[2] OpenUP, http://epf.eclipse.org/wikis/openup/
[3] Subversion, http://subversion.tigris.org/
[4] Bugzilla, http://www.bugzilla.org/

- Features are for customers/stakeholders, requirements are for developers...
- Define one requirement at a time. Avoid conjunctions (and, or) that make multiple requirements.
- Before writing a requirement, define all entities involved
- Write a requirement as a complete sentence, with a subject and a predicate (usually a verb). The subject is an actor, or a design entity.
- Examples: **Teacher** *evaluates* a **Student Assignment**; A **Student** *is enrolled* in many **Course**s.

(B) Capture also the supporting requirements (e.g. Business Rules) as the supporting requirements template shows.

(C) [Use Case Model] Make a use case model.

- A model of the system use cases and actors and the relationships between them.

(D) Detail some use cases and scenarios. Follow the use case template.

- Some use cases and scenarios may need to be described in more detail to validate our understanding of the requirement and to permit software development to begin.
- Usually only architectural significant use cases and those prioritized by the stakeholders as being important.

(E) [Initial Domain Model] Define a conceptual model if the model helps you to have a big picture of the business entities and transactions.

- The model should be easily readable. Even the stakeholder should be able to read it.
- At this stage the model usually represents only the business transactions, i.e. objects that represent the main focus of business in the problem domain.

3.2 Initial Architectural Modeling

(F) Identify an architecture that has a good chance of working.

- Document the architecture as the architecture template shows.
- At least the logical architecture should be established – the large-scale organization of the software classes into packages (or namespaces), subsystems, and layers.
- We recommend considering a logical architecture built around the layered architecture.

3.3 Project Planning

(G) Plan iterations.

- Map requirements to development iterations.
- Document the mapping into the project plan document.

4 Development Iterations

4.1 Iteration Modeling

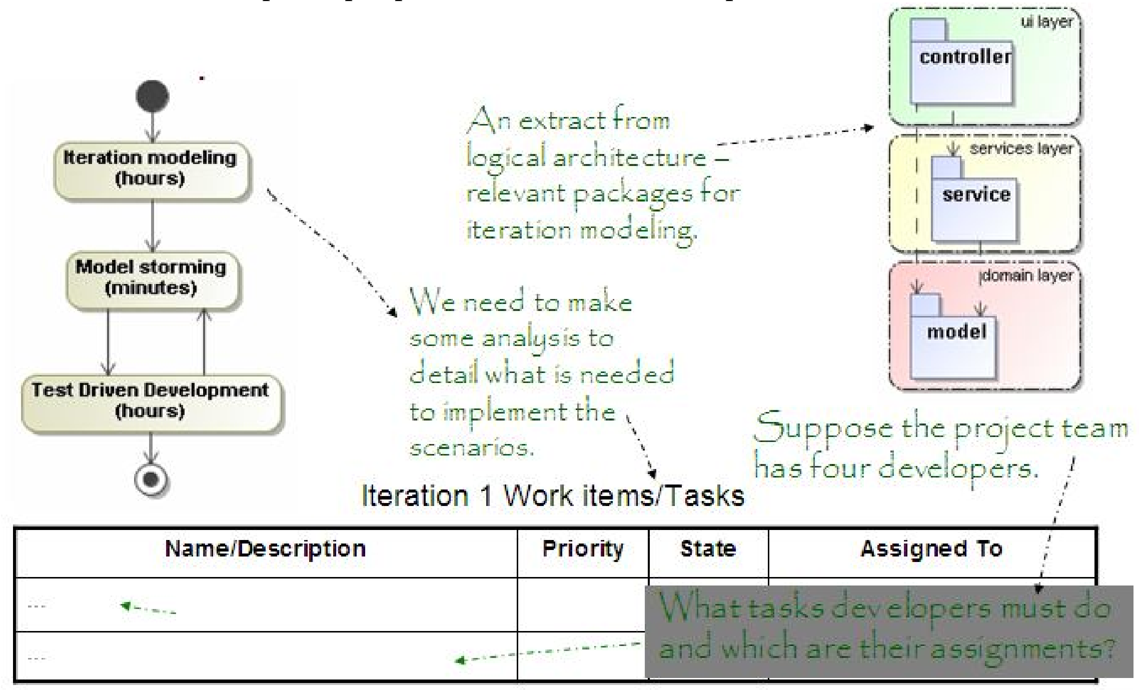
The scope of this activity is to determine the iteration tasks and to assign those tasks to developers – see Figure 4. The entire team could be involved in this activity. The project manager and the team leaders are responsible for assigning the iteration work items. Document the design as the design template shows.

(A) Refine the domain model.

- Model entities bounded by the current iteration scenarios.

(B) Determine the system behavior.

- **Write a system sequence diagram for each scenario**
    - We identify the system operations (system public interface).
    - The methods should be written in terms of domain objects.
- **Write system operation contracts**
    - Description, pre/post conditions for each operation.



Iteration modeling (hours)
Model storming (minutes)
Test Driven Development (hours)
An extract from logical architecture – relevant packages for iteration modeling.
ui layer
controller
services layer
service
domain layer
model
We need to make some analysis to detail what is needed to implement the scenarios.
Suppose the project team has four developers.
Iteration 1 Work items/Tasks
Name/Description
Priority
State
Assigned To
...
...
What tasks developers must do and which are their assignments?

Figure 3 Iteration Modeling

(C) Model controllers and services

- Apply Controller pattern and the patterns indicated by the established logical architecture.
- Assign the system operations to controllers
- Establish the services used by the controllers

(D) Make a work item/task list as the work item/tasks template shows.

- Now you can build a task list expressed in terms of
    - new controllers, services, and domain objects designed above
    - data access objects needed for the new designed domain objects according to the logical architecture
    - required views for the current iteration – the views will call the controller methods
- Of course, model storming sessions will be needed because the design usually is not complete.

4.2 Model Storming Sessions

(E) Develop a sequence diagram for each (non-trivial) system operation (controller operation).

- Apply GRASP patterns (General Responsibility Assignment Patterns).
- Developers design now all repositories, validation rules, and other needed domain objects.
- The design is made by developers assigned to implement that system operation.
- Document the design as the design template shows.

(F) Update Software Class Diagram.

- Add methods
- Refine associations

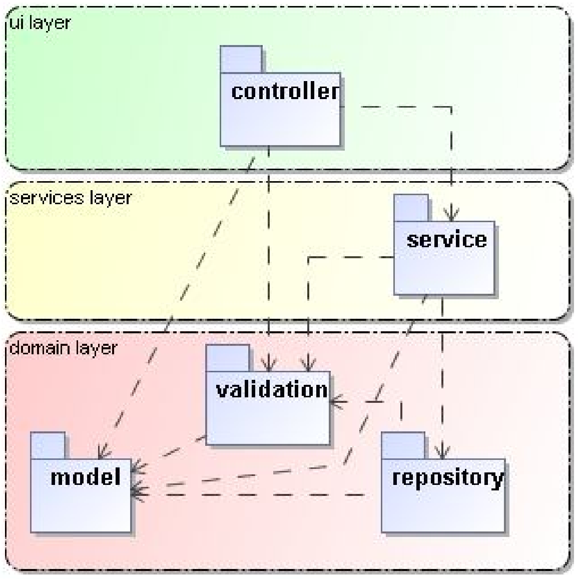
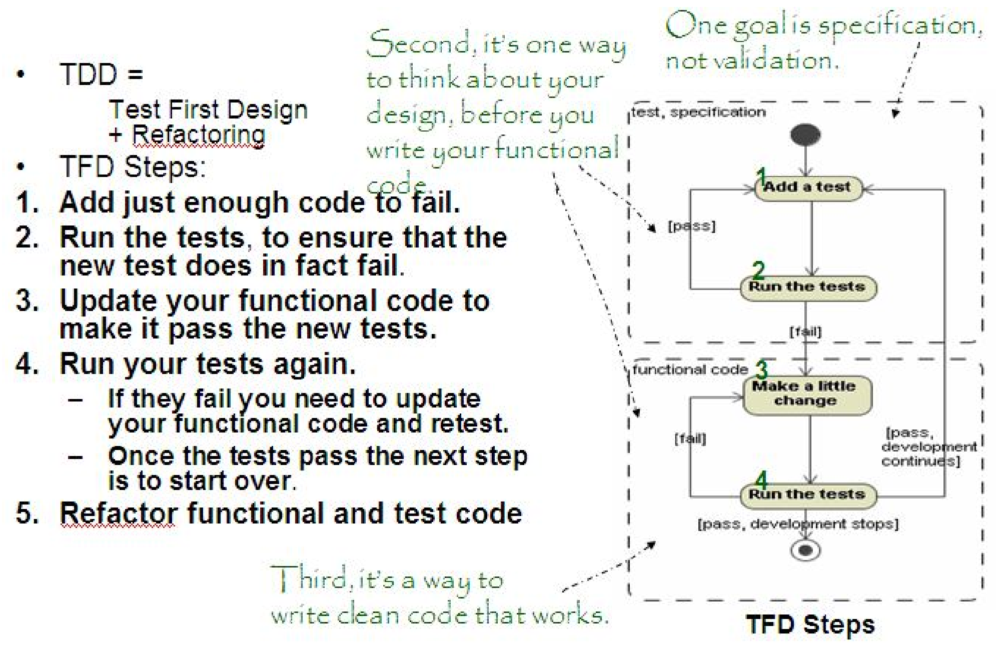


Figure 4 Logical Architecture Extract for Model Storming

4.3 Test Driven Development

Figure 6 shows the TDD steps. We strongly recommend following the Spring MVC tutorial[5] for more details.

[5] Thomas Risberg. Developing a Spring Framework MVC application step-by-step. 2005. http://www.springframework.org/docs/MVC-step-by-step/Spring-MVC-step-by-step.html



- TDD =
  Test First Design
  + Refactoring
- TFD Steps:

1. Add just enough code to fail.
2. Run the tests, to ensure that the new test does in fact fail.
3. Update your functional code to make it pass the new tests.
4. Run your tests again.
   - If they fail you need to update your functional code and retest.
   - Once the tests pass the next step is to start over.
5. Refactor functional and test code

Second, it's one way to think about your design, before you write your functional code.
One goal is specification, not validation.
test, specification
1 Add a test
[pass]
2 Run the tests
[fail]
functional code
3 Make a little change
[fail]
[pass, development continues]
4 Run the tests
[pass, development stops]
Third, it's a way to write clean code that works.
TFD Steps

Figure 5 TDD

5 Milestones

The following evaluation criteria should be met for project milestones:

- **Objectives milestone**:
  - Scope definition
  - Schedule estimates
  - Definitions and priorities for an initial set of requirements
  - Risks identified and mitigation strategies proposed
- **Architecture Milestone**:
  - Vision, Requirements, and Architecture are stable
  - Major risk elements are addressed and resolved by testing and evaluating executable prototypes
  - Stakeholders agree that the current vision can be met if plans are executed to develop a complete system on top of current architecture
- **Operational Capability Milestone**:
  - The product release is stable and mature enough to be deployed in the user community
  - All functionality has been developed, and all alpha testing (if any) has been completed
  - In addition to the software, you have developed a user manual and a description of the current release
- **Product Release Milestone**:
  - User satisfaction and product acceptance