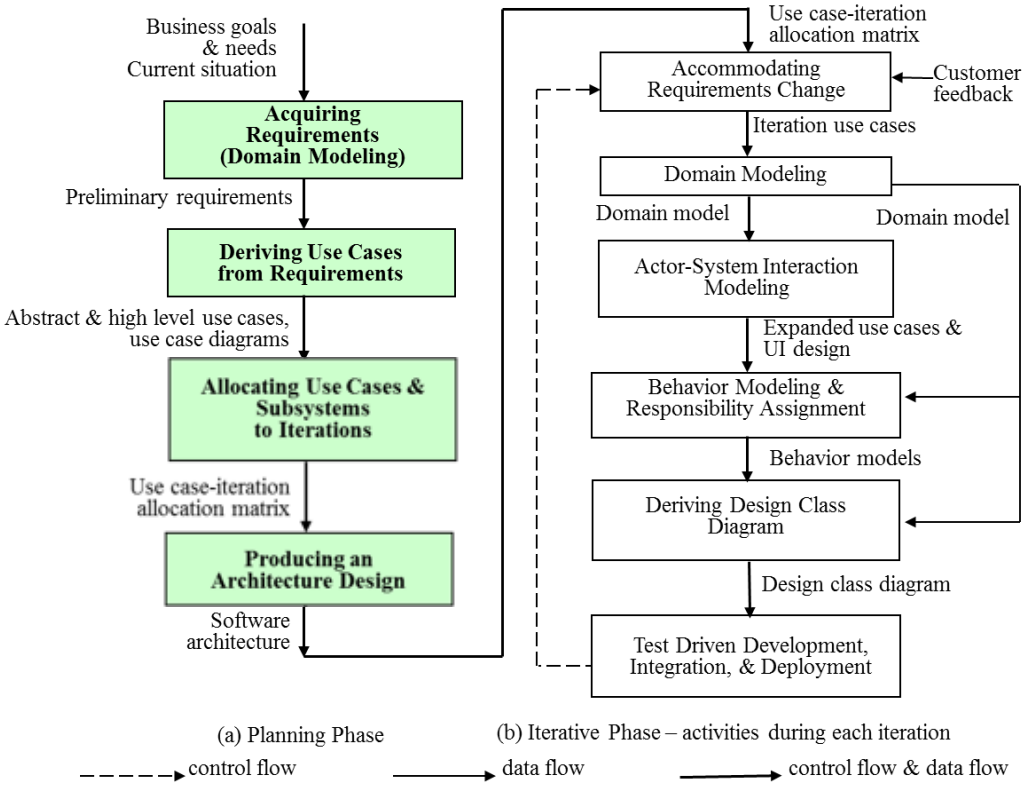
Core Project #1: Planning Phase

For the first core project, your team is to go through the Agile Unified Methodology planning phase. The requirements elicitation part has been done for you and collected as the Cougar Health core project scenario. Your team must now classify the requirements, create an initial domain model, derive the high-level use cases from the scenario, allocate use cases to iterations and produce an architecture design.



Your submission includes

- Effective use of the Redmine project management tool,
- An initial domain model,
- A requirements specification,
- High-level use case descriptions,
- A requirements traceability matrix,
- A schedule of which use cases will be developed in Iterations 1-3, and
- A prose description of the architecture design for the project.

General Approach

After reading the scenario a couple of times, I suggest that each team member begin an individual assessment of the requirements and overall domain model. Then, the team should meet and brainstorm with the individual ideas to develop the team's concept of the requirements and initial domain model. Capture the requirements in the Requirements Specification and capture the domain model in a UML class diagram. Once complete, the team should define the high-level use cases and derive the associated high-level use case descriptions.

The team may find instances where work can be done in parallel. For example, while coming to agreement on a domain model, the team may find that it can more easily capture glossary items for the Definitions, Acronyms, and Abbreviations section of the Requirements Specification. Alternatively, there may be times when iteration is appropriate. The team may define an initial domain model, put items into the Requirements Specification, and discover changes necessary to the domain model.

- **Assign roles to team members** such as project manager, scribe, rat-hole watcher, etc. The rat-hole watcher is the person who tries to get the team back on track if conversation turns away from the immediate purpose of the meeting.
- **Assign responsibility to team members**. If the team agrees someone is to do something, write it down and provide the team member a clear understanding of what is to be done and by what date. In team meetings, note progress on each of these action items.
- **Take team meeting minutes**. Go in with specific goals for the meeting and come out with specific items completed and actions to be performed—including to whom they were assigned and when they are due.
- **Use Redmine** to plan, and to store and share the artifacts that you produce. There is a DMS documentation section for you to upload these directly versus using a Git client.

Requirements Elicitation

Refer to Kung, Chapter 4, for information regarding requirements elicitation. Keep in mind that the information gathering results are presented as the project scenario. As you review the project scenario and draw requirements from it:

- Begin developing a list of keywords, acronyms and abbreviations to define and list in the Requirements Specification, and
- Capture requirements in a table by type (functional or non-functional) and uniquely identify with a label—use an 'FR' prefix for functional requirements and a "NFR' prefix for non-functional requirements; add columns for dependencies and priority. For this scenario, the core functionality of computing costs and benefits is higher priority than collecting enrollee and healthcare service provider inputs. See example table below.

Req #	Priority	Requirement	Dependencies
FR-01		Calculate the daily interest to be added to the account.	NFR-08
NFR-08		All output must be displayed within five seconds.	[N/A]

If you have questions or need clarification about the requirements in the scenario, post them to the Moodle General Class Forum.

Requirements Specification

Refer to Kung, p. 98, Figure 4.5. Provide a document entitled "Software Requirements Specification" that includes the following sections from Figure 4.5:

1. **Introduction**
   - 1.1. **Scope**: Provide a one paragraph summary of the software product; include a name for the product.
   - 1.2. **Definitions, Acronyms, and Abbreviations**: List the term and define it
2. **Interfaces** [Use only prose descriptions, no diagrams]
   - 2.1. **System Interfaces**: Provide a one paragraph description of other systems with which this software system will interface.
   - 2.2. **User Interfaces**: Provide a one paragraph description of the user interfaces that the software system will possess.
   - 2.3. **Constraints**: Describe any constraints on the software development such as language or other that you perceive.
3. **Specific Requirements**
   - 3.1. **Functional Requirements**: Provide a listing of each functional requirement and provide each requirement a unique identifier.
   - 3.2. **Non-functional Requirements**: Provide a listing of each non-functional requirement and provide each requirement a unique identifier.

Note that for the Section 3 requirements, insert the tables in which the requirements were captured during requirements elicitation, above.

Initial Domain Model

Refer to Kung, p. 117, Figure 5.7 for the steps in Domain Modeling. Do not perform Actor-System Interaction Modeling from Chapter 8 for this project. Follow each step in this diagram as described in section 5.4 and subsequent subsections. Also, refer to Sections 5.5 and 5.6 for completing the domain model.

- Conduct the brainstorming steps described in Kung, Sections 5.4.2 and 5.4.3.

- Construct a UML class diagram that represents the visual domain model as described in Kung, Sections 5.4.4 and 5.4.5. Remember, the domain model does not include behavior (methods).

Once the domain model is complete, create a table that lists each component in the domain model and provides an associated, unique two or three-letter abbreviation for it. Add a description column to briefly describe the purpose of the component. See the example in the table below:

Identifier	Component	Description
BKS	Bank System	Component with main() entry that manages software system flow of control
TI	Teller Interface	User interface for tellers using the Bank System

High-Level Use Case Descriptions

Develop high-level use case descriptions. Refer to Kung, Section 7.4 (and subsections) for the steps in developing high-level use cases. While developing your use cases, you should employ the checklist shown in Kung, p. 177, Subsection 7.4.1 to ensure that your team has identified a proper use case. However, do not provide a table of the checklist (like the one shown in Figure 7.3) as part of your project submission.

As part of your project submission, do provide:

- A table of your developed use cases that includes a use case label with a 'UC' prefix, the use case name, the "begins with" and "ends with" conditions, and the subsystem to which the use case belongs (referenced to a domain model component); see example table, below

Use Case	Name	Begins With	Ends With	Subsystem
UC-04	Deposit Money	Customer providing money to be deposited	Customer receiving receipt for deposited money	TI
UC-05	Withdraw Money	Customer submitting filled out withdrawal slip	Customer receiving cash from account and a receipt	TI

- A requirement use case traceability matrix (see Kung, p.184, Figure 7.7), including priority.
- A *New* issue in Redmine for each use case.

For Redmine use case issues:

- For the *Subject*, enter the use case prefix and number, a colon, and the name of the use case. For example, **UC-04: Deposit Money**
- In the *Description*, enter the desired functionality from the perspective of the primary actor, in the form of a user story. This user story takes the form:
  The [actor name] wants to [use case name] in order to [use case goal].
  For example, **The Customer wants to deposit money in order to add funds to an account**.
- In *Priority*, set it to *High* if the use case is high priority; *Normal* otherwise.
- Do not modify the other fields, including *Assignee*, for now.

Use Case Allocation Table

The team must decide during which of three iterations (see below) each use case will be developed. Construct a Use Case Allocation Table as discussed in Kung, section 7.4.5, and exemplified in Figure 7.15. Instead of "person-week", **state the estimated effort in "person-hours"**. To determine the estimated effort, hold a team planning meeting. Note that the 'priority' in this table is the use case priority calculated in the traceability matrix. Numerically, 'High' is a '3', 'Medium' is a '4', and 'Low' is a '5'.

Iterations

Starting with the next core project (CP02), your team will enter the Iteration Phase of the Agile Unified Methodology. The dates for the start and end of each iteration is as follows:

- Iteration I: 24 September – 17 October 2019
- Iteration II: 18 October – 14 November 2019
- Iteration III: 15 November – 05 December 2019

Architecture Design

Using chapter 6 of Kung and the course lecture slides as a guide, define the architecture style for the system that you will build. Select from the styles mentioned in Figure 6.3. In the Architecture Design Report, provide a summary of why the team chose this system and particular architectural style(s) for subsystems around which to design the software for this project. While multiple styles may relate to different parts of the system, identify one specific architectural system and style that best fits the primary purpose of the software that you intend to build.

Team Tools

1. Use Redmine's DMS tool to track document artifacts. Be sure to assign an *Assignee* and make use of start and end dates.
2. You may use Word, OpenOffice, or LibreOffice for the reports, as long as they can export to PDF for the project submission.
3. Use Dia or Visio or Visual Paradigm for your UML class and use case diagrams. **Do not submit Visio, Dia, or Visual Paradigm files, nor should you submit images**. Export the UML diagram to .png or another common image format. Then, add the image to a word processing file.
4. Tables and outline headers for the Requirements Specification can both be easily created in the word processing applications mentioned in Item 1. If you don't know how to use these tools, post to Moodle for help.

Include your team name, team members' names, and assignment identifier on all artifacts.

Submissions

This project requires both team submissions and individual submissions. **All submissions are due 24 September 2017. Teammates evaluations are due a few days after.**

Team Submissions

All team submissions must be well-organized in the Redmine DMS. Create a folder called "CP01" to hold all of your submission artifacts. The latest version of each artifact will be used for grading.

T1. Redmine issues for planning and development of the project artifacts. Use the *Documentation* tracker for planning and artifact development.
T2. Provide the initial domain model and include the component abbreviation table on a separate page.
T3. Provide the requirements specification with the table or functional and non-functional requirements.
T4. Provide the high-level use case descriptions.
T5. Provide the requirements use case traceability matrix.
T6. Provide a schedule table that allocates each use case to an iteration and an estimation of person-hours required each iteration for all use cases allocated.
T7. Provide a summary of your selected architectural style.

Each artifact must be placed in its own PDF document. Name the items as follows:

"CP01A_[TeamName]_DomainModel.pdf" for artifact T2,
"CP01A_[TeamName]_RequirementsSpec.pdf" for artifact T3,

"CP01A_[TeamName]_UseCases.pdf" for artifact T4,
"CP01A_[TeamName]_RqmtsUseCaseTraceaility.pdf" for artifact T5,
“CP01B_[TeamName]_ImplememntationSchedule.pdf” for artifact T6,
"CP01B_[TeamName]_ArchSummary.pdf" for artifact T7.

You may also submit the brainstorming classification work as tables similar to Figure 5.8 and Figure 5.11. This may allow me to provide partial credit if your domain model is erroneous. If your team elects to provide the work, then please put it into an appendix named, “CP01A_[TeamName]_AppA_Brainstorming.pdf”.

Individual Submissions

These are required by each and every team member. Note that artifacts I2 and I3 will not be shared with the class or team in a manner that is directly attributable to you.

I1. Maintain your time log in Redmine by associating your time to specific issues.
I2. Provide an individual post-mortem that includes at least one thing that worked well for you personally and one thing that did not. Describe why they did/did not work and how you might ensure they can be used again or avoid the same problems. These must be different items than what is provided from the team post-mortem. A link will be provided in Moodle for this submission.
I3. Evaluate your team members’ reported times and efforts. Complete the Teammates peer-evaluation for this core project.

Artifact I2 must appear in a document named, "CP01A_[LastnameFirstinitial]_Post-mortem.pdf". **Please put your name IN the document**.

Please note that if (for example) the team agreed that your part would take a proportional amount to other work but you complete it in an hour, it is then expected that the remainder of the work will be redistributed appropriately.

Grading

A base grade will be developed for each team. Each team member's grade will be based upon the amount of effort put into helping the team earn what it did. An individual team member's grade will (directly or indirectly) be based upon factors such as:

- Having planned to do a fair portion of the whole effort.
- Having successfully completed the portion of the effort that was planned.
    - This includes completion and correctness of the developed artifacts, what your team members state that you did, and the effort that you claim.
- Your time logs, Teammates evaluation, and post-mortem.

This assignment is worth a maximum of 100 points. Points will be earned based upon effort in the following areas:

- (15 points): Team Use of Redmine, including issues for estimated planning and artifact development, and management of issues as work progresses
- (15 points): Team Develops and Documents Accurate Requirements Specification
- (20 points): Team Develops and Documents Accurate Domain Model
- (15 points): Team Develops and Documents Accurate High-Level Use Case Documents
- (5 points): Team Develops and Documents Accurate Requirements Use Case Traceability Matrix
- (10 points): Team Develops And Documents Implementation Schedule
- (5 points): Team Develops and Documents Architecture Summary
- (10 points): Individual Use of Redmine, including Time Log
- (5 points): Individual Post-Mortem

Notes:

1) What is submitted as being from the team is being accepted as being approved by all team members. Therefore, whether you developed it or not, you are responsible for its correctness. Take time to review other's work.
2) If you do not complete the Teammates Survey within the allotted time, ten points will be deducted from your individual grade on top of any other deductions that you may receive.
3) Do not include the square brackets '[' and ']' in the file names. They are simply indicative of placeholders for you to put your name or team name.