Requirements Engineering and Software Architecture Project Description

This project is student-driven. There will be external sponsors, users, and others that you need to interact with, but the project team is responsible for the quality of all deliverables, so they must drive the project. The requirements project will be completed in the first seven weeks of the semester. The follow up software architecture project will be completed during the remainder of the semester.

You will work in teams. Each team will be partnered with another team for collaboration. You will be provided with a generic application domain. Each team will then select a specific system problem within that domain to represent as stakeholders for the partner team. Teams will collaborate with their partner team in the roles of business stakeholder and developer. Developers will elicit requirements from stakeholders and potentially other users stakeholders recruit. Stakeholders will provide review feedback to developers. External user experience designers may also collaborate. Tasks will progress from requirements through architecture activities, building on each other in a progression. Work will be performed as class activities in conjunction with lecture material and outside of class. Each team will produce prescribed requirements and software architecture documents. In addition each individual will be periodically quizzed to answer questions about recent work activities to be submitted for grading as directed by the instructor.
Project Activities and Artifacts

The project will be incremental. The first increment will cover product concept and formulate a team plan. Subsequent increments will evolve and review a software requirements specification and associated artifacts. The schedule of deliverables is posted on the course schedule and all dropboxes in myCourses have due dates.

#0 Project Kickoff

Teams will be announced.

Organize teams – choose a team name, define roles, and plan process logistics.

Each team must report their status once per week. The report tool is open but reports must be available to the instructor via a public URL. Content should be professional, and do not provide personal information such as full names or addresses.

The weekly status report needs to contain:

- A list of what tasks were accomplished for the week, and number of hours contributed by each team member.
- A list of what tasks are planned for the next week and who will work on each.
- Any challenges or issues that you need to share.

Teams are required to maintain a project plan. Use the “Requirements and Architecture Project Plan Template”.

- System concept statement you are representing as stakeholders.
- A link to the status report.
- Requirements engineering risks and risk management plan
  - Note: these are risks for the requirements engineering phase of the work, NOT the risks for the entire project life cycle.
- Defect log
- Schedule (dates for significant milestone tasks and deliverables for requirements engineering work) ¹

#1 Select a System

The instructor will recommend a target application domain. Do some research, brainstorm and innovate to identify a system within that domain. Better yet, find a collaborative external sponsor. Write up a short system concept paper to be provided to your partner team. You must become knowledgeable enough to adequately represent your
system as stakeholders. Review with the instructor to validate the appropriateness and scope of the system.

The project scope should be sufficiently broad so as to provide an interesting system problem for software architecture design. Scope might be derived from some combination of external device interfaces, external system interoperability, core computational algorithms, business process user role collaboration, distributed services, and data relationships and operations. There should be an expectation of significant architecture requirements derived from quality attributes such as performance, security, availability, interoperability, and modifiability.

The instructor will assign team stakeholder-developer partnerships.

NOTE: The following activities are written from the perspective of the developer role. In each case you will review your work with the stakeholders as directed by the instructor. It is also a good idea to conduct an internal team review.

#2 Vision and Scope Document
Use the Vision and Scope template. All sections should be completed. Also submit an up-to-date project plan.

#3 Draft Software Requirements Specification
Use the Software Requirements Specification (SRS) template. “Draft” means that all sections in the SRS template have been broadly considered. No SRS template boilerplate remains. Use case models and analysis models should be documented using the Use Case Document template.

- All sections addressed with some content, some TBDs acceptable.
- Have identified high value and architecturally significant use cases and actors.
- UML use case context diagram
- With brief descriptions of use cases and of actor groups and characteristics
  - Have detailed at least one use case (most architecturally significant – the “core” use case)
  - Normal flow of events and significant alternate flows of events
- Have object-oriented models for use-case analysis (use-case realizations) of at least one use case (architecturally significant use case)
  - Detailed class diagram and sequence diagram
- Results organized in initial draft of a SRS package (see the notes below on the contents of an SRS “Package”)²
- Updated project plan – risks, schedule, defect log

#4 Final Software Requirements Specification
- All sections are complete and have been reviewed.
- Have identified 90% of use cases and actors
  - With brief descriptions of use cases and of actor groups and characteristics
• Have detailed 60% of use cases (most risky – high value, architecturally significant)
  ○ Normal flow of events and significant alternate flows of events
• Have object-oriented models for use-case analysis (use-case realizations) of 30%
  (maximum of three) of architecturally significant use cases
  ○ Class and sequence diagrams

  Notes: The class model should provide sufficient detail so as to be traceable to its use case. At a minimum, details should include class attributes and responsibilities, and message labeling. Stereotype classes are acceptable supplemented as necessary with other classes. Proper UML notation is expected.

• Updated project plan – risks, schedule, defect log

• Presentation
  ○ Introduce the client sponsor and stakeholders
  ○ Summarize the system vision and scope
  ○ Describe the elicitation process
  ○ Discuss functional requirements
  ○ Describe the system use case context model including actors and use cases
  ○ Discuss non-functional requirements
  ○ Summarize project status and reflection report

Grading
Vision and Scope: 100 points
Draft SRS: 100 points
Final SRS: 200 points
Project Presentation: 50 points

Notes:
1. This is a schedule for your team's requirements engineering work. It is not a proposed (hypothetical) development schedule for the product (you won’t be building the product!). Instead, the schedule might have things such as what your deliverables are in each iteration of requirements engineering plus, perhaps, time for interviews, time for evaluating competitive projects, time for artifact reviews with stakeholders, etc. It is
to be a tool for your team to focus and coordinate your requirements engineering efforts on the most effective tasks. Note that your schedule should be driven by risk (address high exposure risks early), so use your requirements engineering risk management plan as information to guide your schedule.

Note that the Vision and Scoped document may hint at a "product development" schedule by prioritizing features and placing features in a suggested incremental release plan.

2. The SRS may not be a single document but, instead, a collection of related documents and artifacts. Combining all the content into one document is not wrong, but (as with software) it is easier in large projects to separately modify and concurrently work on and evolve separate parts of the whole. For example separate documents for:

- The requirements overview (product purpose, stakeholder description, glossary, use case overview, feature/requirements priorities, etc.),
- User models (if they are simple, put them in the overview document),
- Each of the detailed use-case specifications,
- Supplemental requirements (quality requirements, development constraints, etc.),
- The glossary, priorities, traceability matrix and other artifacts,
- Analysis models (they are not formally part of the SRS, but can complement it) and other forms of validation. These can also be provided as appendices if you want the SRS to be a single, integrated document.
Software Architecture Project Description:

As a team you will design a software architecture for the system for which your team specified requirements. You will analyze that specification to identify functional and architectural significant requirements, then design an architecture for that product. [Your instructor may assign you a SRS from a different system from the one you specified.]

Project Activities and Artifacts

The project will be incremental. The first increment will cover product concept and formulate a team plan. Subsequent increments will evolve and review a software requirements specification and associated artifacts. The schedule of deliverables is posted on the course schedule and all dropboxes in myCourses have due dates.

#1 Project Startup

The weekly status report started for requirements should be continued for the architecture work.

The project plan started for requirements should be updated to reflect the architecture design work:

- Architecture engineering risks and risk management plan
  - Note: these are risks for the architecture design aspect of the work, not the risks for the whole product development
- Defect log
- Schedule (architecture design tasks, assignments, deliverables, dates)

The team should implement any process improvements identified in the reflection document from the requirements engineering project.

#2 Draft Software Architecture Design

Use the Software Architecture Document template. “Draft” means you have documented the big picture overview plus a thin vertical slice – a key view and associated element catalog.

Provide an updated project plan and status report.

#3 Final Software Architecture Design

The final version of the documentation, incorporating changes from the draft reviews and feedback.

Include views from each of the three categories of structural views as described in the text:

- Module
Components and Connectors
Allocation

The architecturally significant interfaces provided by the modules identified in the Module View should be documented in the style of the template described in the text. You should also describe the patterns and tactics you have chosen to address the quality attributes in the Supplemental Specification.

Provide an **updated project plan and status report**.

Most architecture design documents will probably be 15-20 pages long.

Presentation:

- Project introduction and background
- Summarize functional requirements and architecturally significant requirements including quality attributes
- Explain your architecture design. This should be the majority of the presentation.
  - Big picture overview
  - Show the views and explain their role and relationships.
  - Design rationale – what patterns and tactics were used and why? Why is this a “good” design?
- Status and reflection

Grading:

The grade for the final version of the documentation will be based on:

- 10% - Organization and presentation of content, spelling, grammar, etc.
- 10% - Project plan and status report
- 80% - Technical merit (accuracy, completeness, clarity, etc.)

Your grade for the project will be calculated from the deliverables as follows:

- Project startup: 25 points
- Draft: 50 points
- Final Version: 100 points