

Team Project: 48433 Software Architecture Autumn 2007

Assignment Overview and Conduct

Modern approaches to software engineering have a different emphasis than more traditional approaches. In particular, it is generally accepted now that a software process:

1. Supports the “people” aspects (collaboration, teamwork, communication) of software development in addition to the “process” aspect.
2. Has a strong reliance on software architecture to guide and define the structure of the system.
3. Proceeds in a series of relatively short iterations.
4. Emphasises delivery of working software in relatively small increments.

For the purposes of this undergraduate class in software architecture, the delivery structure of the assignment is loosely based on the first two phases of the Unified Process, which are called Inception and Elaboration. In this subject, there will be one iteration for Inception, and one for Elaboration. Each iteration ends with a milestone.

The Unified Process emphasises the importance of producing an early working version of the system in the form of an executable architectural prototype. In parallel with this work, the architectural design is carried out, resulting in an artefact called the Software Architecture Document. Therefore, you will be working on two major assignment artefacts throughout the semester:

1. The written component of the assignment produces the Software Architecture Document.
2. The programming component of the assignment produces the Executable Prototype.

You will deliver these two artefacts in a series of two milestones. It is important to remember that the Unified Process is *iterative*. That is, each deliverable is not a completely different artefact, but a newer and better version of the same artefact. For example, for the written component, you first deliver the initial architecture design, but in a similar form to that which is expected for the final Software Architecture Document. For milestone 2, you improve this design and the document. Note that in this subject, we are focussing on a lightweight, high level description of the design of the system; we are *not* doing detailed design.

For the programming component, you start by creating a simple working program that illustrates that you are able to perform development. For milestone 2, you then elaborate on that same program to create and deliver a completed executable prototype. This prototype should reflect the architectural design that you developed in the Software Architecture Document. It is to be demonstrated in a lab (see subject schedule).

See the Subject Schedule for milestone due dates. Each milestone is worth 25 marks. There is also an Individual Contribution component worth 10 marks, for a total project mark of 60.

You will be able to select a project topic from the set provided in the Appendix.

The assessment criteria for the project will be accessible through ReView, an online criteria-based assessment tool. Make sure you take careful note of these assessment criteria whilst working on your project. Your project deliverables will be assessed against them.

Note that assignments are to be implemented using only approved technologies. By default, you are expected to use Java, to use the Eclipse development platform and the provided source-code repository, and to use the third-party products used in the Labs. One of the key issues in marking your programming deliverable is that it is easy to build and run your system, so we have to limit the number of different packages that are being used by the class.

Teams

All assignment work is performed in teams. At the beginning of the semester, you need to form yourselves into teams as quickly as possible. There are two types of team:

1. A small team consists of four people. This is the default team size.
2. A large team consists of 5 to 6 people. Large teams must submit a one-page proposal to the instructor, describing the team charter, skills and experience of each team member, and two meeting times per week that most team members can attend.

You must form yourself into teams by the end of the third week of semester. If you are not in a team by then, you will be assigned to one. Each team must formulate a written set of guidelines for managing their project teamwork. This should include allocation of team roles, schedule of meeting dates, modes of communication, conflict resolution, protocols for using the Wiki and source-code repository, etc. More information and resources on teamwork will be provided in class. You will need to show this set of guidelines to the instructor.

Individual contribution (10 marks)

All students are expected to contribute equally to the written and programming components of the assignment, although within each component there can be allocation of different tasks.

Verification of individual contribution will be done by examining the Wiki contributions, the source-code repository commits, your individual logbook, and your individual demonstration of contribution through participation in the presentation and lab demonstration sessions. Evidence of minimal contribution can result in a mark of zero for the project. Otherwise your individual contribution will be worth 10 marks out of the total project mark of 60. Students that show evidence of a thorough and thoughtful approach to all aspects of the team project will gain the full 10 marks.

Logbook

Each student is expected to keep an individual logbook. The logbook is your personal record of your approach, progress and contribution to the team project. Your logbook is used to assess your approach to the development of your project, and should include your design notes and sketches, meeting notes and actions, implementation notes and issues, record of time spent on specific tasks, and so on. The main purpose of keeping a logbook is to develop professional habits and skills in keeping technical notes, records of decisions and activities undertaken at the individual and team level. It will be checked regularly during class sessions. It is recommended that you use an exercise book and handwrite your entries. You may paste in copies of selected extracts from your Wiki documentation, if required.

Exceptional merit

In exceptional circumstances, the instructor may award an additional five bonus marks to an individual or team for outstanding quality of the assignment deliverables.

How/When to submit

The programming component of the assignment is to be submitted *incrementally* by checking it in to your subversion repository.

The written component of the assignment is to be submitted *incrementally* to the Wiki.

See the Subject Schedule for milestone due dates/times.

Brief overview of the Software Architecture Document and the Executable Prototype

The article you develop on the Wiki for the Software Architecture Document should contain the following items and descriptions. It is up to you how you structure the document, but remember to make it clear and easy to follow.

Software Architecture Document

The Software Architecture Document describes the design of the architecture, and provides the basis on which further design and implementation work takes place. It should:

1. Summarise the system's purpose
2. Analyse the system context
3. Describe elaborated customer needs as a set of usage and quality narratives.
4. Provide a set of views that describe the architecture, including its structure, behaviour, implementation, constraints, and so on.
5. Justify all architectural decisions. Refer to the system context, stakeholder input, results of prototypes, and so on.
6. Include notes and discussion on issues that will need to be addressed as development proceeds.

The Software Architecture Document's primary purpose is both to capture the reasoning behind the architecture, and to enable further development in the Construction phase. To that end:

1. Make sure that the document is no longer or more verbose than needed. Remember that the document will be read (amongst others) by impatient software developers.
2. Make sure that the document and your design are realistic. Don't design some fancy pie-in-the-sky architecture that the software development teams can't or won't build. Everything in the architecture must be feasible, and that feasibility must be supported by reasoning or prototypes.

Executable Prototype

The executable prototype is the core of the system being built. It provides the platform on which subsequent development takes place (in the Construction phase). It should therefore:

1. Contain all key pieces of the architectural infrastructure.
2. Implement key pieces of functionality that prove the architecture's feasibility and act as a model for further development.
3. Have resolved all configuration management issues.

The executable prototype must represent the architecture expressed in the Software Architecture Document (and vice versa). Any discrepancies between the two will reduce your marks severely. The quality of your configuration management and the readability of your source code also play a big part in your final mark.

The milestones below describe what is to be submitted for each component of the assignment at each milestone.

Milestone 1 (25 marks)

This is the first milestone, and marks the end of the Inception phase of the Unified Process. The artefacts produced for this milestone are therefore an analysis of the system---it's context, objective, stakeholders---and a complete conceptual architecture. In addition, you are seeking to establish that you have the necessary technical resources to construct the system.

Initial Architecture

For the written component, you will produce the first version of the Software Architecture Document. This document should include the following items for this milestone:

1. A brief overview of the system
2. Stakeholder needs and expectations, captured in personas and narratives
3. Identification and discussion of key contextual factors
4. Description of elaborated customer needs as a set of usage and quality narratives.
 1. The usage narratives are written for the various kinds of users of the system (remember the users are a subset of the stakeholders, and these narratives should focus on specific situations of use)
 2. The quality narratives are written for the two (2) key quality attributes that you have chosen as most relevant for your system.
5. An initial conceptual architecture, derived from the system narrative using the technique described in the textbook.
6. An elaborated conceptual architecture, including use of stereotypes, discussion of components and their responsibilities, data models, run-time and lifecycle events, and behaviour captured with use-case maps.
7. An initial execution architecture and implementation architecture, with some discussion of how the identified contextual factors will affect ongoing development of the execution and implementation architectures.
8. Brief notes on anticipated issues for following milestones.

Note that a diagram without any explanatory text is not very informative. For example, carefully explain the trace on a use-case map, making sure you identify and describe the responsibilities exercised by a particular component. Give reasons why/how this use-case map supports or reveals deficiencies in your architectural design.

Project presentation and feedback session

Each team will present to the whole class their initial architectural design for this milestone. You will have 10 minutes to succinctly describe the key features and issues of your architecture, with thoughtful reference to stakeholders, contextual factors and quality attributes. The instructor will provide on-the-spot critique and feedback on your design work.

Initial Development Capability

In this initial milestone, you are seeking to establish that you have the technical means to complete the project. To this end, you must first:

1. Demonstrate the ability to use the Eclipse development environment to write, compile, and execute Java programs.
2. Demonstrate the ability to check files into and out of your team's subversion repository.

In addition, you are required to produce, in Java, a user-interface technical prototype that consists of at least two processes:

1. a user interface that contains buttons to simulate key runtime events identified in your architectural analysis; and
2. a service that listens for requests from the user interface process and returns a response to each.

(This second process can be based on the ThreadedServer program from Lab 1, unless you want to do something fancier.)

The program must be checked into the `src` directory of your subversion repository. That is, do not create a directory called something like “milestone1” -- the repository contains your executable prototype, and evolves as the milestones proceed.

Note that each team member must use subversion to check in your work. The instructor will check the subversion logs to verify that you are using subversion correctly and participating in the development of your team's code. If you are not, you may receive zero for the programming component of this milestone and be required to submit milestone 2 individually.

Laboratory demonstration

You need to demonstrate your working technical prototype in the lab. A quick check will be made on your ability to use the Eclipse development tool and the Subversion repository.

Milestone 2 (25 marks)

This is the final milestone, in which you deliver, at the end of the Elaboration phase, a completed Software Architecture Document and Executable Prototype. The main focus of this milestone is to design, explore, and validate the execution architecture.

Elaborated Architecture

This is the written component for this milestone. In this iteration, you are expected to have completed the execution and implementation architectures, and revised or refined the conceptual architecture and any other relevant artefacts.

Specifically, you will have:

1. A complete execution architecture section. You must have at least a process view or concurrent subsystems view, and a deployment view based on it. You will also show, using use-case maps from key events, that the execution architecture is feasible and does not have any inherent performance problems.
2. A complete implementation architecture section. You will explain the structure of the implementation architecture, and provide analysis explaining your choice of off-the-shelf and custom-built components.
3. Brief notes on anticipated issues for following milestones.

In this milestone, you are to revise and complete the architectural design, especially in light of any prototypes you have built along the way. Please refer back to the overview of the Software Architecture Document to ensure you have not missed any elements.

Include a one- to two-page critique of your architectural design now that you have produced an Executable Prototype. Discuss the areas in which your design was strong or weak, and how the prototype served to illuminate various aspects of the design. You may use diagrams to support your claims.

You will need to justify all architectural decisions that you have made during the analysis and design. Make sure that the items and diagrams are supported with clear, explanatory text. Refer to the system context, stakeholder input, and so on, as part of your justification.

Executable Prototype

This is the programming component for this milestone. You must deliver the following to meet this milestone:

1. A running system that contains the key elements of your execution architecture.
2. A user interface (non web-based) that allows exercise of key runtime events.

3. A web interface that provides the key user functions available over the web.
4. The structure of the prototype must match the execution and implementation architectures. That is, it must:
 1. Use at least some of the technologies identified in the implementation architecture, and
 2. Have the same concurrent subsystems as the execution architecture design.

(If your design has 18 concurrent subsystems, then so must your program. This is a powerful incentive to make your design no more complex than necessary.) For the purposes of this subject, a concurrent subsystem is a process started by a shell script; a process is started either by a shell script, by using an `exec` call, or by the Java Process class; a thread is a Java thread.

In addition, for this milestone you need to solidify your development structure. Specifically:

1. Modify your program to use Java packages.
2. Modify the provided ant script to compile your program (if it does not already)
3. Modify the provided startup shell script to start all of your concurrent subsystems.

Laboratory demonstration and Peer assessment

Each team will demonstrate a working prototype in the lab. Each member of the team will be asked to explain the mapping between their architectural design and the executable prototype. Any student unable to do this, will be liable to receive zero for the programming component of this milestone.

Your prototype will be peer assessed by another team. The assessing team will be provided with a copy of the assessment criteria for this milestone. The focus will be on the quality of the prototype and how well it matches the architectural design as specified in the execution and implementation architectures. The assessing team must provide informative comments for each assessment criterion, indicating the reasons for the grade assigned to a given criterion. ** In turn, the quality of the peer assessment will be an explicit part of the assessment criteria for each team's total project mark.

Individual reflection

Deliver a personal reflection on this project. You should describe what you learnt, and how, during the conduct of the project. You should also include some reflections on the overall process of creating the Software Architecture Document and the Executable Prototype, and on the subject as a whole. The reflection should be one to two pages (no more) in length. Include the reflection as part of your logbook. (It can be typed and pasted into the logbook)

Appendix. Software systems narratives

These project topics are written as “vision” level descriptions of software systems, from which you need to extract requirements as needed. The purpose of these system descriptions is simply to provide you with a starting point and focus for exploring the field of software architecture. You won’t be penalised for embellishing or altering your system, provided that you’re reasonable and consistent about it.

Topic A. Gurgle Planet

In the year 2045, the earth's resources have been drained to the point where the ecology is widely considered to be at breaking point. What used to be the “third world” has imitated the consumption patterns of what used to be the “first world,” and many scientists believe that, despite significant advances in the management of natural resources in the last 50 years, the earth is on the verge of a complete ecological collapse.

An urgent meeting has been convened to find a solution. For the first time ever, the leading world powers have agreed to radically curb their own consumption in the interests of a global solution. Your company is one of several that have been asked to propose solutions to one part of this problem: a global resource monitoring system, that can be used by the policy makers to find out where resources are being created and consumed. You have been asked to assemble a crack team of technologists and architects to investigate and propose a solution based on your company's know-how and existing products.

In your initial investigations, you have discovered that the problem is more complicated than you thought ... natural resources are so decimated that there is perceived to be a need to manage and detect not only classes of ecosystem such as forests and agricultural zones, but also the presence of every species of plant and animal, regardless of whether it is directly used in food or other production. Water and soil quality data must be collected and analysed. The last remaining oil reserves must be detected and mapped out, as well as all other minerals.

Consumption patterns also need to be analysed. Since not all companies or governments are forthcoming with useful data, much of it must be obtained surreptitiously, by the use of fly-over aircraft, “smart dust” wireless sensors, and satellite imagery in the visible, infrared, and radio spectra.

Finally, waste management data needs to be obtained and tracked, in order to maximise reuse of mineral resources and minimise exposure of the remaining ecosystems to harmful toxins. You have also realised that effective use of this information demands an effective human interface that conveys the right information at the right level of detail. As a model, you have chosen a system initially developed 40 years earlier as your vision of an effective interface that combines context, variable information display, and spatialisation capabilities: Google Earth. One of the cynics on your team has dubbed your developing project “Gurgle Planet,” as a pun on the phrase “going down the gurgler,” and referring to the earth's ecosystem.

Topic B. CyberTone Systems

CyberTone Systems is poised to make a splash into the “ubiquitous targeted advertising” market. They are vying with a number of other startups and established companies trying to gain a prominent position in this market. CyberTone believe that they have the edge, but they need architectural expertise to make their product vision a reality. Wys I. Wyg, CEO, explains:

“This is an extremely challenging project. No two ways about it, it's a tough nut. Never mind the business and legal stuff, that's been hard but I think we've got that on track now, but technically it seems to be getting more complex every day. We need to put a lid on it, scope it into something feasible that we can deliver in a short time frame and wow the public and the media ... oh, and the paying customers as well. Yes but of course, we want it to look good but it has to work properly and the basic architecture has to take us through several versions. We can't afford to re-architect this thing every time, we need [thumps table] the infrastructure in place to keep one jump ahead of those guys [indicates the window and therefore, presumably, their competitors].

“You know the scenario, I expect. Seen *Minority Report*? Tom Cruise. Great actor. One of the best. Anyway, there's a scene where he walks along the shopping mall and the walls are talking to him. They know it's him and they know what he buys, and they try to get him to buy more. Well OK, we don't say that in our press releases but hey, ... anyway, look, don't repeat that and you can do it, right? Good job, we're depending on you.”

A little technical perspective is provided by Mr Con Currency, CyberTone's Chief Technical Officer: “The reason we are going to win this is because we have the pedigree. We've got the research behind us now - ground-breaking research, we learned a lot and now it's time to make the early investment a reality. We're calling this the Universal Targeted eXperience, you know technically speaking we can use this for a lot of things other than advertising...

“What our strength is in the algorithms for adapting image display and audio in response to physical movement information. Once someone is within our sensor field, we know how to figure out where the person is, and what they doing. Where they're looking, what gestures they are making. All of this information feeds into algorithms that can be configured according to how you want the screens and audio to respond to that person. We can localise audio in 3D space even.

“So it's not much of a stretch to add the information about *who* the person is. Sure, there are privacy issues but I think eventually most people will opt in. Retinal scanning is a bit of a fantasy but a simple wireless device can be worn by anyone who wants our “experience.” You carry a credit card, right? Of course ... well that's all the space it takes. We can pick up the code and connect to a centralised identity bank. This is what makes it targeted, of course, because now we can have the algorithms tailor its output based on information like your consumer profile, demographic, and so on. Assuming it's in a database we can access, that is.”

Just before you sign the contract with Bystander Technologies, they tell you that they've decided to use the architecture that you will be designing for them for other products as well, not just advertising. “You could tell someone walking into the news-agent at the airport, that they are going to be late for their flight if they don't hurry up. Now *that* would be useful!”