

# 48433 Software Architecture

Reflections on Spring 2004

John Reekie

Faculty of Engineering

University of Technology, Sydney

December 2004

## Contents

<b>1 Preliminaries</b>	<b>2</b>
<b>2 Personal reflections on Autumn 2005: JohnR</b>	<b>2</b>
<b>3 Numeric survey results</b>	<b>4</b>
3.1 Subject feedback survey (SFS) . . . . .	4
3.2 Discussion . . . . .	6
<b>4 Written student feedback</b>	<b>6</b>
4.1 Blogs . . . . .	6
4.2 Informal surveys . . . . .	10
4.3 Formal surveys . . . . .	11
<b>5 Assessment and results</b>	<b>12</b>
<b>6 Concluding remarks</b>	<b>12</b>

## 1 Preliminaries

This short report contains the reflections of the instructors and students concerning the delivery of the subject 48433 Software Architecture in the Autumn 2005 semester.

The original design document for the subject is [1]. Previous reflections for this subject were published for Autumn 2004 [3] and Spring 2005 [2].

## 2 Personal reflections on Autumn 2005: JohnR

I'm mostly very pleased with the way the subject ran this semester. There were some surprises and disappointments (see below), but for the most part the quality of work delivered by the students, and the amount and quality of discussion on the on-line forum, reflects positively on what I think is now a very solid foundation for delivering a mid-level undergraduate subject on Software Architecture.

### Lectures

The lectures went well this semester. Despite the feedback from one student (page 12), I was *not* “late and unprepared” for lectures, although I did move material around as I saw fit. Lectures generally started ten minutes after the scheduled start time, and generally finished within the allotted hour.

We had a proper lecture theater this semester, and I think it makes the world of difference.

One thing I am starting to feel is that, as I move material around to adapt to the settling structure of the course and textbook, some of the lectures are losing the continuity they had when they were originally designed using the “story” technique. I'll work on those when I have time, but there's no hurry.

### Course materials

This semester, I gave all students a copy of the current draft of the textbook being written by myself and Rohan McAdam. (Why are we writing this book? — because there are no books that clearly explain the basic concepts of architecture! If there had been a good textbook available 18 months ago when I redeveloped this subject, I would have used it!)

I think the book draft really helped in improving the quality of the student's written deliverables this semester. On the down side, few students read the supplementary materials I had printed for them. The supplementary reader, which cost only \$13 from the Union Shop, contained hundreds of pages of *selected* chapters from a dozen books on Software Architecture. Unfortunately, less than half of the students purchased a copy, even at that price! (And the informal subject survey indicates that students read between zero and 3 of these excerpts.)

So I was a bit flabbergasted that a number of students complained about the “half-written” textbook or gave a low mark to the survey question on adequate resources for the subject. I'm thinking that the subject feedback surveys have reached the limit of their usefulness and are now largely just a vehicle for venting. See Section ?? for more.

## Labs

The labs went OK. I think that adding “at home” sections was a mistake. It’s better to have labs that can be completed easily within the allocated two-hour slot.

This semester we had a lab on the Abacus program. Tim O’Neill refused to give a lecture to accompany the lab, and so that lab overall was of limited use and will be ditched in future.

## Blogs

This semester, blogs were an option again, and I very much enjoyed reading most of them. I’m happy to give the five bonus marks for a reasonable effort at a blog, and this semester the students commented on each other’s blogs and had some quite lively discussions about architecture-related topics. I think it worked really well, and I’d like to thank those students that wrote blogs and participated in the discussions. Thanks :)

## Assignments

This semester I moved to a “milestone” style of delivery. It didn’t occur to me that students wouldn’t have any idea of what a “milestone” is (in a software development context). Next semester I will revert to the simplest possible solution: three assignments, each marked out of 20. No gates, no milestones, no peer reviews. Just a mark. Done.

Overall, the quality of the work delivered in the assignments was far higher than it has been before. Some will be used as examples for future iterations of the subject—my thanks to those students that put in the extra work to prepare their assignments for use by future classes.

So the *good* thing to take from this semester’s assignment structure is that the assignments are structured as a written and programming part carried out in parallel. This is how the Unified Process does it, and by gum it worked in this subject!

## CVS and the “documentation specialists”

OK, so here is this semester’s eye-opener. I required—for the first time this semester—that the programming part of the assignments be checked into CVS. The reason is simple: it’s a lot easier for me to check out and update a CVS module than it is for me to muck about with CDs or floppy disks and zip files. A secondary benefit is that the students can start to learn about *the* most basic tool in software development (after an editor and compiler, that is).<sup>1</sup>

Along the way, I realized that this would also be a good way for me to do a quick verification that all team members were making a reasonable contribution to their team’s project. (All CVS entries are logged and time-stamped.) The protest after I mentioned this on the discussion board was “eye-opening.” The long and short of it is this: some of our Software Engineering students consider themselves to be “documentation specialists” (their term, not mine!). Apparently, they simply cannot write a few hundred lines of code, and they have managed to hide this for most of their degree program.

---

<sup>1</sup>I mean revision control, not CVS specifically.

I was very lenient this time around, but next semester strict requirements for reasonable participation in both the design (written) and the executable (programming) parts of the deliverable will be published in advance, and enforced.

### **Exam**

The mid-term exam was useful, although much too much time was spent on it, since I had a tutorial specifically about it. Again, an opportunity for carping that I would rather avoid in future. I do think the students benefit from a mid-term so I will continue to do it next semester.

The final was quite a hard exam. Some students commented that perhaps 40 marks crammed into a three-hour time-slot is a bit much, and I'm inclined to agree. Next semester the final exam will be worth 30 marks.

### **Concluding remarks**

I really enjoyed delivering the subject this semester, for the most part. There were a few low points, like the silliness with students who can't write a line of code to save their lives, and one student who smugly remarked about the subject being "software design for idiots" (and then proceeded to moan about not being able to write an executable prototype.) Well, you can't help everyone. Still, overall it was a really great class. My thanks to the students in this semester's class, it really was a good class.

And finally, since I'm supposed to assess everybody in this subject, I'll give *myself* four stars out of five :-)

## **3 Numeric survey results**

### **3.1 Subject feedback survey (SFS)**

Here are the numeric results from the University-run subject feedback survey (SFS) conducted towards the end of the Autumn 2005 semester. The number of responses to the survey was 13.

The scale used for answers to these questions is:

1. Strongly disagree
2. Disagree
3. Neutral
4. Agree
5. Strongly agree

- Table 1 summarizes the overall results for Spring 2004 and includes the questions.
- Figure 1 shows the results compared to the Faculty and the University.
- Figure 2 shows the history of SFS results over the last four semesters.

Nr	Question	Mean	SD
1.	The subject was delivered in a way which was consistent with its stated objectives	4.0	0.7
2.	My learning experiences in this subject were interesting and thought provoking	3.8	0.9
3.	I found the assessment fair and reasonable	3.7	0.8
4.	There were appropriate resources available to support the subject	3.3	0.8
5.	I received constructive feedback when needed	3.3	1.0
6.	The teacher appears to be well prepared and presents the material in an organised manner	3.5	0.8
7.	The teacher is able to explain concepts clearly	3.5	0.8
8.	Overall, I am satisfied with the teaching of this staff member	3.6	0.7
9.	Overall I am satisfied with the quality of this subject	3.5	0.8

Table 1: Subject Feedback Survey results, Autumn 2005

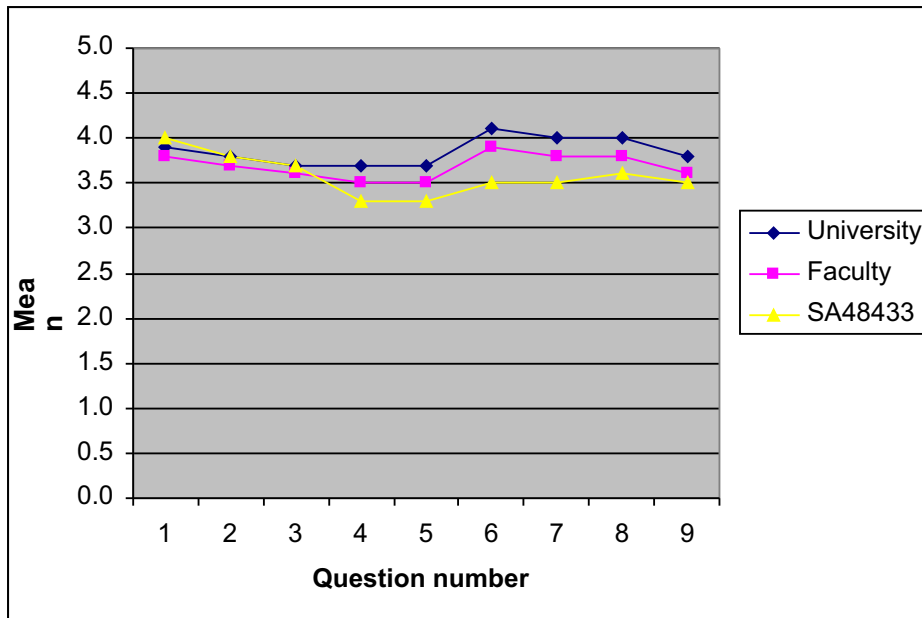


Figure 1: SFS means compared to the Faculty and the University

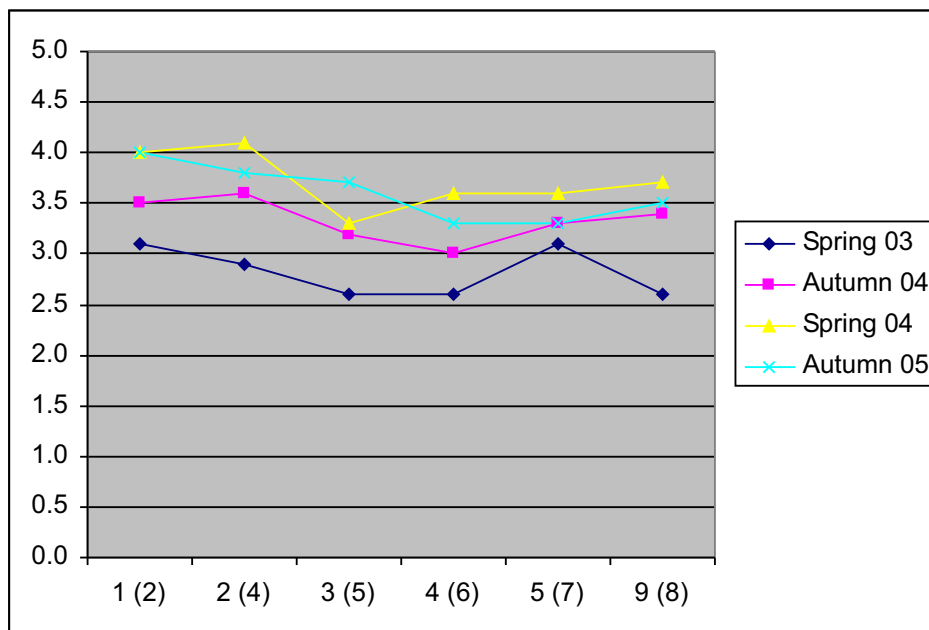


Figure 2: SFS means over the last four semesters

## 3.2 Discussion

...

## 4 Written student feedback

### 4.1 Blogs

A number of students kept blogs this semester. The blogs were supposed to enhance the learning experience of the students, by giving them an opportunity to reflect on what they were doing and learning and to share it with other students. To encourage students to write blogs, 5 bonus marks were awarded to each blog, provided that a reasonable entry was made about one a week.

As part of this, “bloggers” were asked for the last blog entry to be a summary of their experiences in and thoughts on the subject. Here are the blog entries of that nature.

(Attributions are to the student’s username on the class website.)

- **Final blog!**

Firstly, about our final delivery of the architecture of the system. In terms of the programming part, for the previous milestone, we just had two concurrent subsystems, the input concurrent subsystem and the output concurrent subsystem, actually that are the client and the server. Then, for this milestone, we separate that into 5 concurrent subsystems as in the execution architecture: Speaker(Output), Screen(Output), Database,

ProcessControl (Like the Image and Sound Controlling Algorithm) and the Input Subsystem. Acutally, the ProcessControl is the server and other except the Database is the client. We use socket to connect between Client and Server. The execution prototype is good, I think and it can be used in the future when building the real system.

In terms of the documentation, we got useful advice from John. For example, in milestone 3, we did not consider about how the architecture response to the stakeholders which is very important due to the fact that the documentation is assumed to deliver to the key stakeholders including the CyberTone company, the development team and also must satisfy customers'needs. The other big improvements are in the implementation architecture which should show clearly the communication between components, infrastructure components such as Java Image should also be showed. Moreover, the analysis of the choice of infrastructure must be more clear and specific. The comparison between different infrastructures is also useful.

In general, I am happy with what have been done.

Secondly, in terms of the whole subject, I think this subject is really useful. I am now relatively confident in building the architecture of the system. At least, I know what should be do, which steps that I should follow to construct the Conceptual Architecture, Execution Architecture and Implementation Architecture as well as building the prototype for the system. Working in team is also useful so that we can share knowledge, understandings to make the product better. And as each person has strengths and weaknesses, so that can help others to get out of weaknesses. The way that the assignment is organized is better, in my opinion, in comparison with the last semester. Working with milestones improves the system gradually. These are true for both the documentation and the programming. For example, for the execution prototype, developing it gradually from the first milestone to the last delivery reduces the workload and makes us improve that rather than building it in the last step. For the lectures, it is really organized and I know what is expected for each lecture. The assistant of the texbook is necessary for me to understand more about each issues.

Generally, I think I have studied a lot from this subject and the knowledge from this will certainly help me in my future career.

— *utsstudent*

- **Final blog about the subject**

This week is the last week of the semester, so I think it is about time to give a final blog which reflects my opinions on the whole subjects. Even though we have already done the survey of the subject last week, I think an individual reflection would be more helpful in giving feedback about the subject. To be more precise, in this blog, I will show my opinions about the three parts of the subject Software Architecture: the lectures, the assignment and the labs.

Firstly, I think that the knowledge provided in the lectures is very relevant to the purpose and objectives of the subject. From the lecture

notes, we have learned many things in the Architecture phase of the software developing lifecycle. To be more precise, we have been introduced to various types of architecture: conceptual, execution, implementation and deployment. In addition, knowledge about usage narratives, use-case maps, impact maps, quality attributes, etc... are also provided. However, in my opinion, I think that only the first six lectures (in the text book) are really relevant to the subject's objectives. The others (provided in separate handouts) are more difficult for me to understand and relate them to the subject's purpose. The lectures about "culture," web services and middleware are such the examples.

Secondly, since I have already had a separated blog for the assignment, only a brief reflection on the assignment is given here. Generally speaking, the two parts of the assignment (documentation and programming tasks) have supported each other as well as the subjects purpose and objectives. To be more precise, the documentation task provide practical experience for the student after learning the theories from lecture notes whereas the programming task supports the documentation task and makes the system become more "realistic".

Thirdly, the labs also provide new experience as well as support knowledge to do the programming task of the assignment. To be more precise, by doing the labs, we have learned how to use Eclipse as Java IDE, how to use CVS and how to use HTTP server as well as access database using Java. In addition, the use of Ant (Java "makefile") can be also learned from the lab. However, just like the lectures, there are some labs that, in my opinion, are not very interesting or much related to the subject. The lab about the Abacus methodology is a good example for this situation.

Generally speaking, I am quite satisfied with the subject's quality as well as the way it is organized. All of the modules (lectures, labs and assignments) have co-operated with each other very well to fulfill the purpose and objectives of the subject.

— *vinhkanhit*

- **So where did 'Shadow' come from and whats the relationship to Software Architecture** (excerpt)

Before this subject my understanding of software architecture was like looking for my shadow in a dark room. Generally architectures i had worked with consisted of one view that was a somewhat clumsy conceptual architecture with some nice big blobs. I didnt consider what the domain level responsibilities were and in most cases it was just a part that was completed quickly without too much thought before getting into the design.

A shadow is a different view of the world around you it can look very different based on time of day and each different time of day can highlight something different. Like a shadow the software architecture can be drawn in different views each giving a different perspective helping to describe the whole. We have explored the conceptual architecture, execution architecture and implementation architecture. The approach taken in SA moulding the assignments to phases of the RUP has been effective.

— *shadow*

- **Final Milestone Reflection**

### **The Assignment**

The assignment deliverables were fair and set out in a reasonable manner. The assignment milestones were set out progressively to also show the evolving nature of the system architecture, and it shows how from our original conceptual architecture, our final assignment continues to be consistent from our original aims. Most importantly, we realised that by analysing quality attributes, especially in the facets of security and usability, the architectural responses followed in a logical manner.

This two facets concentrate mainly on the Web UI, which I mainly worked upon with Mike:

- Security: The development of session management in `CartManager.cs` was an important aspect in facilitating secure access to user accounts. Additionally, IIS supports SSL encryption by standard to develop a secure web interface for users when critical or sensitive data is transmitted over the net.
- Useability: Whilst Internet Explorer was the recommended web browser to access the AWC Website, the `aspx` pages are also viewable in other popular browsers, including Firefox and Opera. `ASPX` pages are not just limited to being designed in the Visual Studio IDE; they can also be designed in many popular HTML editors – including Dreamweaver and FrontPage.

It was unfortunate I had problems with CVS, particularly as a result of configuration problems initially, and file conflicts later. Hence, I cooperated with Mike to commit together. In hindsight, it would have been more ideal to commit the code into CVS as a single file, as some files, particularly when files/folders with underscores named at the beginning, such as `'_private'`.

### **Lecture Material and Labs**

Well presented, in a logical layout and well structured. The guest speaker from Honeywell stressed the importance of the software architectural model in a commercial environment and the need for designing the interface specific to user needs.

An important point that I took note is designing the system to suit user needs:

- Front-line workers (such as plant operators and maintenance staff) need access to real-time data. Typically, memory-resident real-time databases are employed to ensure information can be conveyed in an immediate and instantaneous manner.

- Managers and Financial officers need access to a complete set of data, to make decisions based upon statistical data - but without the need for immediate information. Long term data are stored in relational databases, in secondary hard-disk storage.

## SA and my studies in SSA

The subject was delivered as stated in its goals in the subject outline. As well, the subject was essential my studies in SSA, for which I also did in the same semester. In SSA, it was important to develop a system architecture which could adequately need the client's requirements. The tailoring of the architecture to meet the requirements was an essential task throughout my assigned tasks in Architecture and Interface design.

The method we approached in SSA assisted greatly as a result of my studies in SA, from Quality Attribute analysis to the development of candidate architectures to respond to these quality (non-functional) needs. Finally, an analysis based upon PURS was used to determine the choice and rationale behind the architecture selected (This document is available upon request).

Software Architecture will assist in my studies for SSD (if all goes well in SSA), and potentially in my future career (but it's hard to realistically predict without a crystal ball).

— *Chiu.Co*

## 4.2 Informal surveys

The informal survey asked the following:

*Please include a short reflection on the subject. Any items of particular note that you would like to comment on would be well-received. Was there anything about the subject that you thought was particularly good? Anything particularly bad? Was there anything that you learnt that you thought particularly interesting? Do you have specific suggestions for improving the subject?*

Responses received (in no particular order):

- I think this subject is quite useful as it involves studying a set of informal requirements and deriving an architecture for the whole software system. I think it's better if we had a proper textbook instead of the blue handout with uncompleted sections. I like it how the assignment is flexible, allows us to build whatever we want, but we haven't used Java for a while, so it took me awhile to get familiar with it. Also, better if I had a group!!!

— Chandra Kwok

- The major problem I had with this subject was that the textbook (i.e. blue handout) was incomplete. As that was supposed to be where all the information was. I think next semester the supplementary readings should be optional, as they together will be able to fill in the blanks in the subject and if it's optional many students don't purchase it.

— *Name withheld*

- It took me a while to understand execution architecture. I had problems in understanding the concepts. Maybe I'm slow. But now I know each component of the architecture spawns a thread (or a separate process). I think more examples would help when concepts of execution architecture is introduced.

— *Name withheld*

- — Lack of clear outcomes required make it difficult at times.
  - Group expressed concern that we did not truly understand what is expected of us.

— *Name withheld*

- — In general I thought it was an interesting subject and I learnt a lot.
  - I didn't feel though that there was a clear provision of what was required and expected throughout the semester.
  - I would have liked some more useful feedback.
  - Too much work was required for milestones with such a small weighting. I would have preferred less milestones where this was the case.

— *Name withheld*

- — Examples in the text should be completely followed through in the text or in the lecture. i.e. from conceptual to implementation architecture on the SAME example.
  - Feedback should be made clearer on assignments.

— *Name withheld*

- I would have to say that this subject was very relevant to me. Out of the 3 viewtypes promoted by this subject, I would think that the execution (concurrent) view is the most insightful and gives a good understanding of how the system operates. Since we can represent the concurrency views at a varying level of detail, it can also be used to get a more conceptual overview of the system.

— *Name withheld*

### 4.3 Formal surveys

The University-run Subject Feedback Survey has two open-ended questions at the end. Following are *all* responses to these questions.

1. *What did you particularly like in this subject?*

- Tying in the laboratory/coding aspects with theory.
- John has an idea of what he is talking about, unlike many other lecturers.
- It is a new concept that expands on the software engineering.
- Free handouts. Bonus marks.

2. *Please suggest any improvements that could be made to this subject.*

- The material did not exactly relate to the subject — half of the lectures I felt had no relation to software architecture.  
The lecturer would sometimes turn up late and unprepared – we began the semester with receiving printouts of lecture notes and towards the end of semester we didn't receive them.  
I have not received sufficient feedback from our assessments — I still haven't received a mark for an assessment handed in a month ago.  
I would have liked to have more supporting material – not an unfinished half-written textbook which shows only one particular view of software architecture.  
Overall, I felt myself not really motivated towards this subject.
- Improved written feedback.
- A completed textbook. Clearer marking scheme/objectives/expectations of milestones.
- Feedback on assignments earlier, more idea on what marks can be earned on assignments (basic and extra sections)
- Better definition of depth that programming/execution architecture has to go into.
- Concurrent viewtypes should be given greater focus as they are more insightful.
- Written feedback about each milestone or assignment.

## 5 Assessment and results

Foo

## 6 Concluding remarks

Well.

I suppose that the feedback surveys have confirmed JohnR's Theory of Student Complaint, which states that the amount of complaining by students remains constant at best, and on average increases slightly with instructor effort to improve a subject! I don't think I will bother compiling information from surveys in future, but focus on what students have to say in their blogs and reflections.

If I step back once again and look at the work that students have submitted this semester, I have to say that it is uniformly a very solid cut above anything that has been delivered in this subject before. There are some students in this class that I would hire in a second, and many others are no slouches. I would encourage any students that read this report to focus on what is important—that is, delivering working software. Good luck for the future.

## References

- [1] John Reekie. The architecture and design of 48433 software architecture—autumn 2004 version. Online at <http://www.eng.uts.edu.au/johnr/pdf/sa-design.pdf>, October 2003.
- [2] John Reekie. 48433 Software Architecture: Reflections on Spring 04. Online at <http://www.eng.uts.edu.au/johnr/pdf/spring04.pdf>, July 2004.
- [3] John Reekie and Lian Loke. 48433 Software Architecture: Reflections on Autumn 04. Online at <http://www.eng.uts.edu.au/johnr/pdf/autumn04.pdf>, July 2004.