

48433 Software Architecture

Reflections on Spring 2005

John Reekie
Faculty of Engineering
University of Technology, Sydney

Contributors

Lian Loke

December 2005

Contents

1 Preliminaries	2
1.1 Overview of the subject	2
1.2 Content delivery modules	2
1.3 Subject resources	4
2 Assessment	5
3 Survey results	7
3.1 Numeric results	7
3.2 Written feedback	9
4 Reflections by students	10
4.1 Blogs	10
4.2 Final reflection	14
5 Reflection by the instructors	18
5.1 JohnR	18
5.2 Lian	19

1 Preliminaries

This short report contains the reflections of the instructors and students concerning the delivery of the subject 48433 Software Architecture the Spring 2005 semester. This is the fourth delivery of the subject in this form.

The rest of this section contains text from the Subject Guide, and is provided for the benefit of readers unfamiliar with the subject.)

1.1 Overview of the subject

Software architecture is the study and design of the most pervasive aspects of a software system, and in particular, of complex software systems. As such, it takes on the task of the high-level decomposition of the system into components, the relations and interaction between them, and how, when combined, they form a system that meets the needs of the system's many stakeholders.

In this subject, you will be introduced to the field of software architecture from a pragmatic and practice-based viewpoint. The subject assumes that you have a good grounding in the basics of programming and software construction, and a good understanding of the basic principles of software engineering and software lifecycles. This subject aims to extend your thinking about software systems to larger and more complex systems.

1.2 Content delivery modules

There are a total of twelve delivery modules, each consisting of a lecture, readings, and other learning resources. The first six are considered to be “foundation” modules, and the remaining modules explore specific topics in more depth.

Module 1: Software Architecture in Context

This is the introductory module to this subject. It explains the concept of software architecture and places architectural work into the context in which software system development takes place.

Module 2: Conceptual Architecture

This module introduces the first effort at decomposing a system into an architecture, based on functional requirements. Simple tools like usage narratives and use-case maps are used to support the derivation of the initial conceptual architecture.

Module 3: Execution Architecture

The execution architecture expresses the system in terms of concurrent elements and hardware elements. This module explains execution architecture, and examines concurrent subsystems, concurrent objects, and deployment.

Module 4: Implementation Architecture

The “build-time” view of the system is contained in the implementation architecture. Program elements and modules, and off-the-shelf components and infrastructure form the third leg of the “architecture triad.”

Module 5: Architectural Styles

Experience and practice have led to a number of recurring architectural structures. These structures (styles or patterns) are part of the “body of knowledge” of the field of software architecture.

Module 6: Quality Attributes

This module takes a deeper look at quality attributes, and how they are analyzed and represented in the three architectures.

Module 7: Realtime Systems Architecture

In realtime systems, *time* is the dominant force that shapes the architecture. Specific architectural responses such as time-based tiers and stream processing are covered.

Module 8: Web Architecture

HTTP front ends are almost mandatory on client-server systems these days. This module provides an accessible introduction to the basic elements of web architecture. It builds on the N-tier architectural style, and covers both the principles of web systems and the implementation architecture of web servers and web services.

Module 9: Component Frameworks

Component-based software engineering (CBSE) is an active area of work in the effort to improve the way in which we build complex systems. This module surveys the current state of the art and practice and explains how components, infrastructure, and middleware support complex software systems.

Module 10: Design and Culture

Design is one of the fundamental activities of software development. This module explores the nature of design as relevant to software architecture, and is a journey through design patterns, iteration and exploration, and the importance of culture and the tension with process.

Module A: Computer-Integrated Manufacturing

Computer-Integrated Manufacturing is a case study that illustrates how many of the concepts covered so far come together in large, complex, “real-world” systems. This module is delivered by an industry-practitioner guest lecturers.

1.3 Subject resources

Subject-specific resources

These are the resources available for this subject.

- Subject Guide. This document.
- Software Architecture Workbook. Describes all tutorials, assignments, and laboratory exercises, and includes week-by-week exercises.
- Excerpts from *Introduction to Software Architecture*. Required readings written specifically for this class.
- Software Architecture Reader. Sections of various books and papers that supplement the material provided in the above.

Additional reading

The following books (from which many of the readings in the Reader are taken) are all in the UTS library. You may wish to use these to further your knowledge of the field.

- *Design and Use of Software Architectures*, by Jan Bosch. Addison-Wesley, 2000.
A nice introduction to the “industrial” view of software architecture.
- *Software Architecture in Practice, Second Edition*, by Len Bass, Paul Clements, and Rick Kazman. Addison-Wesley, 2003.
Useful supplementary reading for students that want to read a lot.
- *Applied Software Architecture*, by Christine Hofmeister, Robert Nord and Dilip Soni. Addison-Wesley, 1999.
Interesting look at factors and system context.
- *Pattern-Oriented Software Architecture—A System of Patterns*, by Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. John Wiley and Sons, 1996.
Contains a good collection of the basic architectural patterns.
- *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis* by William J. Brown, Raphael C. Malveau, Hays W. “Skip” McCormick and Thomas J. Mowbray. John Wiley and Sons, 1998.
Excellent. Contains patterns at all levels, not just architecture.
- *Documenting Software Architecture: Views and Beyond* by Paul Clements and other authors. Addison-Wesley, 2002.
A slightly different set of viewtypes than in this subject.

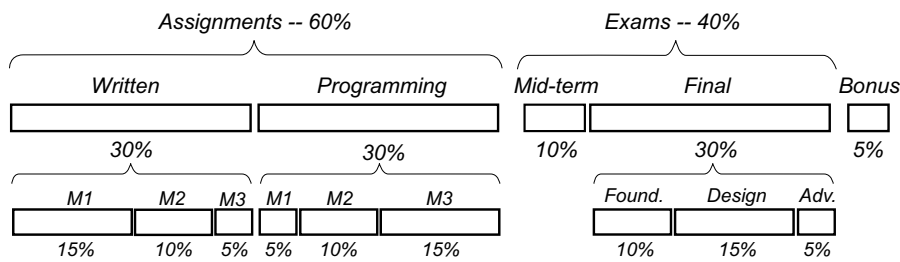


Figure 1: Assessment structure

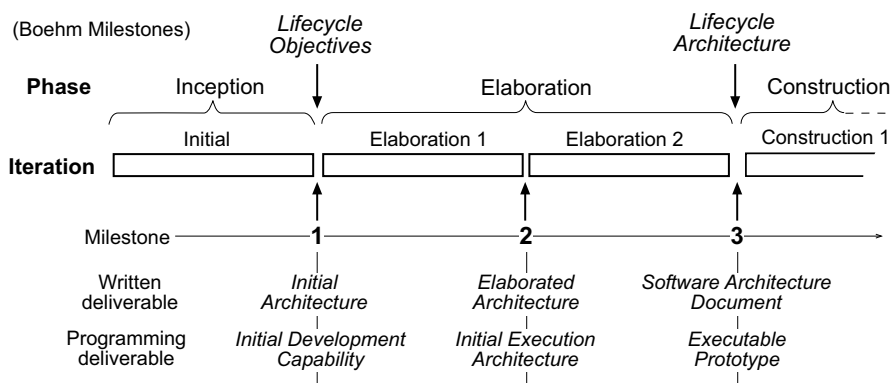


Figure 2: Assignment milestones

2 Assessment

Figure 1 summarizes the assessment structure of this subject. The assessment is split between teamwork-based assignments and individual examinations. All teamwork submissions also include an individual component.

For the purposes of this undergraduate class in software architecture, the delivery structure of the assignments mimics the first two phases of the Unified Process, which are called Inception and Elaboration. The Unified Process emphasizes 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 artifact called the Software Architecture Document. Both artifacts are delivered incrementally, in a series of three milestones. Figure 2 summarizes the structure of the assignments.

Figure 3 summarizes the assessment results for Spring 2005. Figure 4 shows the marks distribution as a scatter-plot. Of 37 students enrolled, only 2 failed. Note that neither of these two students attended the final exam.

There were no appeals lodged. Two students sat the alternative exam, due to travel conflicts and illness. The mean is 70.4. Overall, this a good result and reflects the quality of work submitted by the students during the semester.

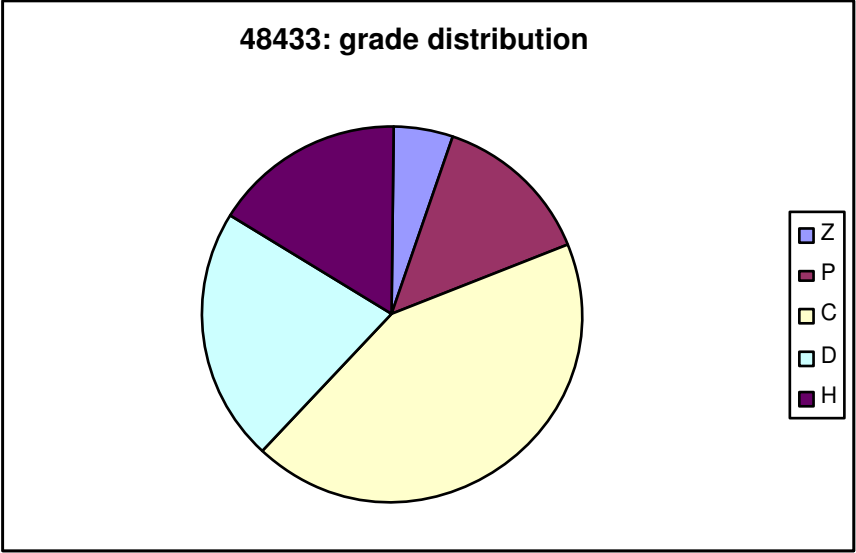


Figure 3: Assessment summary, Spring 05

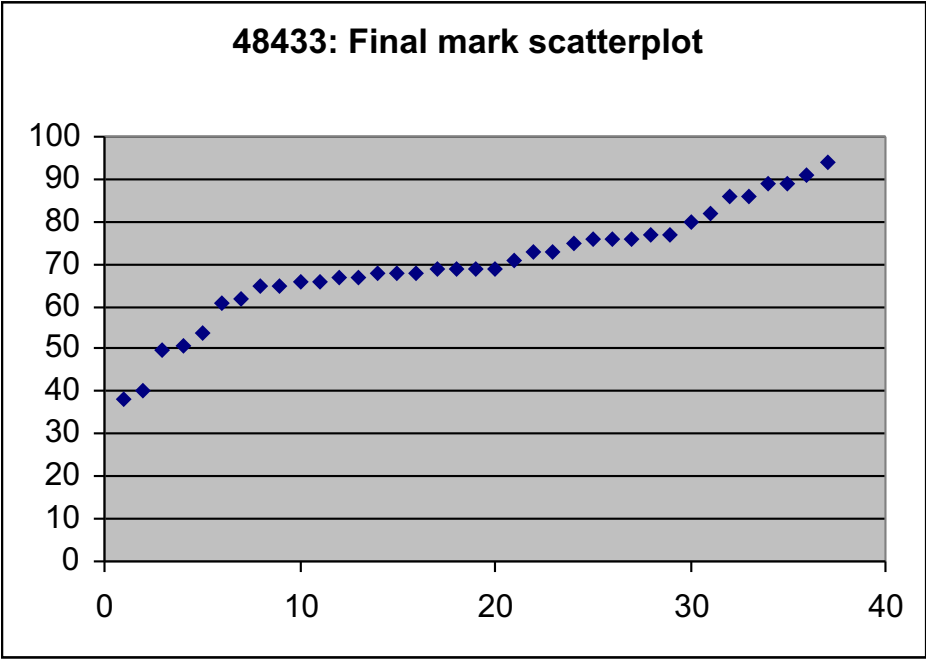


Figure 4: Scatter-plot of final marks, Spring 05

3 Survey results

Here are the numeric results from the University-run subject feedback survey (SFS) conducted towards the end of the Spring 2005 semester. The number of responses to the survey was 16.

3.1 Numeric results

Table 1 summarizes the numeric results for Spring 2005. The scale used for answers to these questions is:

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

Nr	Question	Mean	SD
1.	The subject was delivered in a way which was consistent with its stated objectives	3.56	1.09
2.	My learning experiences in this subject were interesting and thought provoking	3.6	1.24
3.	I found the assessment fair and reasonable	3.5	1.26
4.	There were appropriate resources available to support the subject	3.44	1.31
5.	I received constructive feedback when needed	3.38	1.36
6.	The teacher appears to be well prepared and presents the material in an organised manner	4.06	0.93
7.	The teacher is able to explain concepts clearly	3.56	0.93
8.	Overall, I am satisfied with the teaching of this staff member	3.63	1.41
9.	Overall I am satisfied with the quality of this subject	3.56	1.26

Table 1: Subject Feedback Survey results, Spring 2005

Figure 5 compares the subject to the University and Faculty averages. Figure 6 shows the progression of survey results of the last five semester. (Note that the questions were changed at the beginning of 2005. As a result, some of the questions are not included in this chart, including the only one that improved significantly (question 6).

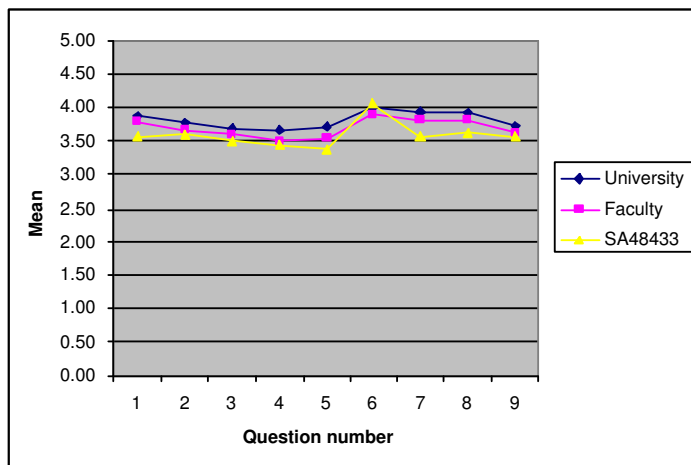


Figure 5: SFS means compared to the Faculty and the University

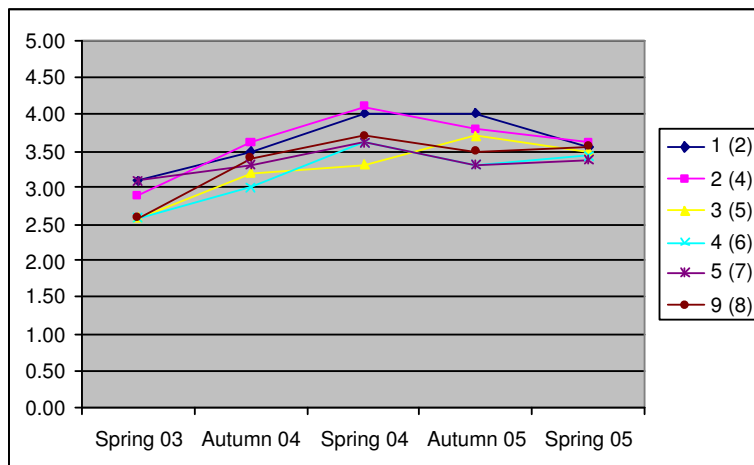


Figure 6: SFS means over the last five semesters

3.2 Written feedback

The University-run Subject Feedback Survey has two open-ended questions at the end. Following are the responses to these questions. Students did not write many comments in this survey, presumably because they had already been asked to write final reflections on the subject as a whole as part of their final deliverable (see next section).

1. *What did you particularly like in this subject?*
 - Very interesting
 - Forces the students to be creative
 - Different architecture
 - Blog concept; endless resources; marking criteria
 - Labs
 - Programming the assignments was a challenging learning experience
 - Learning new stuff

2. *Please suggest any improvements that could be made to this subject.*
 - Needs more information about what was expected from the written assignments
 - Provide examples of software architecture for a case to assist with assignments – clearer objectives and requirements for assignments
 - Assessment where objectives are clear and realistic
 - Please, make the subject less boring

4 Reflections by students

The students were asked to submit a reflection with each deliverable, and some of them earned bonus marks by keeping a blog (“weblog”). It is always interesting to read what students have to say when given the opportunity to be thoughtful and reflective.

4.1 Blogs

The following are reflections written by students in their blog on softwarepractice.org. In some cases I have included only an excerpt (as some of the blog entry was about a specific assignment rather than the subject as a whole). Names given are the student’s username on softwarepractice.org.

spl889

The subject Software Architecture has really introduced me to a part of the development cycle that I was previously more or less unfamiliar with. I initially thought architecture was high level design but I have come to a deeper understanding of the importance of architecture in software just like architecture is important in building a building. It’s a bridge between design and requirements, and it is a very important step to grasp.

The subject helped introduce architecture in a way which was very logical and easy to follow. The programming aspect worked well with the theory and they both complimented each other in the assignment.

The most important things that I learnt in the subject are:

- The different views of architecture and how they appeal to different stake holders and also show different parts of a system. It was great to see how software can be viewed by how it is conceptually constructed, to how it executes, how it is built and how it is deployed. This really helps a developer get a good grasp of what he should build.
- I liked learning about how architecture is designed due to an architectural response that comes from requirements. Architecture isn’t chosen cause an architecture is good or bad but cause some architectures provide things that customers need,
- This leads to quality attributes, every architecture will have its strengths and weaknesses on different quality attributes but to some systems security for example is a must so a architecture (example a closed architecture) that provides that should be chosen.

I feel this was an important subject in the whole degree and despite the fact that it is challenging and has a fairly large work load it enables you as a student to be able to move into more advanced understanding of what software is and how it is developed. I look forward on looking back on the subject notes in coming semesters and perhaps as a professional software engineer. It was well worth the effort.

joselam

So it's the end of software architecture, tears or laughter I don't know, or I won't say because I haven't got my mark yet. =P

Well it's been a fast semester (not fast enough). Software architecture has been very helpful to me because I am currently doing Software System Design and the two compliment one another. Learning software architecture, it has helped me understand my system in software system design more intimately, to the point where I'm able to have my input in the design of the system. It has affected they way I look at the system, and the way I go about the implementation of the system. Well that's how I have applied what I have learnt in software architecture elsewhere.

I understand that there is much more to software architecture than just the conceptual, implementation and execution views (mind you I was somewhat confused between views, viewpoints and architecture until very recently), however I found that those lectures on the views were very interesting. Rohan's lecture really opened my eyes to the complexity of software architecture. I knew that for example Microsoft Windows XP Operating System would have a complex architecture, but something like an oil refinery would have many systems incorporated into one. There are so many things to consider beyond just the software, and they would ultimately affect the software system and the architecture. This is reminiscent of the lecture on enablers, constraints and other external/internal influences. Overall I felt the lectures were helpful, however I was a bit lost in the last few lectures.

The labs I believe would be helpful if you go through it, however I personally would of liked some labs on architectural views and more examples explaining how to break up a problem. I found the text provided by John useful enough for our purposes and somewhat complete (however, I have not read any other literature to really make a comparison). I would of liked a large problem/example to be used in the text and followed through each chapter to really allow us to see how a system unfolds through architectural design.

Finally, I would like to comment on the assignments and examinations. I really like the way it's structure in term of marks, because a lot of effort goes into the assignments (not to bag IDS out, but you do ridiculous amounts of work for 10 marks). It's good that the assignments are worth a lot, and in the end the subject isn't really about theory, but applying that theory hence if the exam was worth too much I don't think it really proves much, except whether you can remember a lot of text or not. The WEWS stirred me and my partner thought-wise, a lot of thinking, planning and reconfiguring took place to make the system work and feel like it was a real working product. I felt however, a lot of work was done in milestone 2 programming wise and not much was left for the final milestone.

I would have to say this subject is one of the better ones, where the people involved actually care, and a lot of effort goes into running the subject. I'm overall quite pleased with what I have gained and learnt in this subject and the manner in which it is.

rishyan

Also, with regard to the subject as whole. I learnt a lot in an enjoyable process. I perform better in assignments etc then in exams so i think the way the course is assessed is fairer than most other subjects. The BLOGs and milestone activities accentuated my appreciation of topics covered in the course. Some topics that were engrossing included quality attributes; architectural styles and real-time systems.

BrettS

It doesn't seem all that long ago that I started doing this subject, and although i have had to do a lot of work for the assignments, I've found that I've also learnt a great deal about the subject of software architecture and how it fits into the wider scope of software development and the lifecycle of software. When i first started the subject, i was unsure about what software architecture was all about, as it was not a term that i had encountered in any of my previous work and university experiences, but now i have a much better understanding of what is actually entailed in the process of software architecture, in looking at the entire software system and seeing how the individual components interact, in a variety of different views. I found that the use of the conceptual view was very helping when it came to starting the assignment, as it helped to develop an initial idea of what the various components of the system are, and how they interact to provide the functionality of the system, through the use of use case maps. Out of the three views that we studied in this subject, i found the conceptual one the most useful, as it helped to start the project along, whereas the implementation and the execution architectures i found were more Dependant on the quality of the conceptual architecture, which made it all the more important to develop a conceptual architecture that was a proper representation of the system to the developed.

I found that the assignments were very demanded in terms of the time required to spend on completing them, and while i did learn alot from them, i believe that it may have been better if the assignments had not been as large, and focusing more on the key concepts of the subject. While i did enjoy coding up the architectural design, i found that it was also time consuming in that we had to do both the documentation and the development concurrently, and i probably would have found it a bit better had we had to concentrate on one or the other, but not both at the same time.

Overall, i found that my experiences in the subject were interesting and useful in terms of my development as a software engineer, and architecture is an area i would be interested in looking into again in the future. I hope that i will be able to one day use these skills in the workplace, as i believe that they are a great tool in understanding and designing a software system, and that i am able to build on these skills in future subjects at university.

Phong

Overall, I believe this project and this subject as a whole, has been very fulfilling and thought provoking for my career as a Software Engineer. It has taught me about an important stage in the software lifecycle and how to approach the problem in a methodical manner. However, the most important lesson is

probably the significance software architecture has on any software system and how it impacts software lifecycle as a whole. Software can be developed without any architectural planning but it would likely result in unmanageable code that is resistant to future maintenance requirements.

chrissT

Throughout the progression of this subject, I have learnt and experienced many things which previously I have seen but never put it to practice until now. Yes, I'm talking mainly about architectural design of a software system. I understood that it is part of projects which takes place in the workforce but never have I designed architecture for a system. I admit that it had been quite confusing at times yet thought provoking because I learnt to put myself in the customers' shoes by trying to understand their needs. It is not just about understanding the requirements of the system as a whole but is also very important to come around to knowing which parts of the system requires what as well as many other things including quality attributes and external subsystems. The subject is structured quite well because I believe that starting out with user scenarios then gradually moving onto different architectural views was a very good way to help our heads get around to what software architecture is really about.

This subject not only taught me how to analyse the system from different view points but also how one system can be designed in a variety of ways in order to understand the problem, meet user requirements, to suit external subsystems and most importantly to take into account the quality attributes of the system. I guess I have experienced a smaller version of a big real world system where we start from something as simple as a page of user requirements to an explosion of design processes and implementation into creating prototype and then back to making sure the prototype meet user requirements.

Reflecting back on the milestones we have completed, we had some issue with time management in milestone 3 because all the assignments were due at the same time but not only that, it was a bit hard to divide work for the writing and programming deliverables. I think if we had more time, we could have improved on our final software architecture documentation.

Also reflecting back to the midterm exam we had, I thought it was a very fair one to test our knowledge and understanding of the subject. I'm glad it wasn't worth a lot like midterm exams in other subject because it's a good way to test yourself (test the water) and find out how much improvement you need and indicates what areas in the subject needs more focus on. Overall, I'm quite happy with the subject although the assignments do require quite a bit of effort as they are worth 60subject has allowed me to broaden my view and understanding to developing software system and its architecture. From the processes undertaken throughout the subject, I believe it will benefit me in the future with bigger project developments that involve teamwork and time management as well as having the knowledge of how to go about dealing with projects.

4.2 Final reflection

The following are excerpts from the reflection delivered with the final assignment, in no particular order. No names are given.

- The subject had at first seemed too abstract to be useful or even understandable. However, it has forced me to think more about how a system should work and how the functionality will be divided between the system components before even starting to think of the actual implementation. Overall, I believe it has made me think more thoroughly about the system design and taught me to look at the whole system and its context (ie. the bigger picture) which I think will be valuable in the future.
- Software architecture, the subject, was kind of different. I found it to be similar to Formal Software Engineering - not the content, but the concepts. The concepts were new and foreign to me (for both subjects) and it was hard to see how these things fit in with software engineering as a whole. With software architecture, I saw that if you knew what you were doing with the architecture it could lead to a more simple development process.
- I feel after this subject I have a more realistic idea of architecture and how it relates to the Software Development Life Cycle. Also the combination of the programming helped make this subject more challenging and allow us to improve our programming knowledge as well as our understanding of software architecture. The both worked hand in hand and complemented each other well.
- I would have enjoyed this subject much more had I had enough time to study it thoroughly and dedicate more attention to it. It was very thought provoking and at times frustrating but the end product is something we can be proud of. This subject was a refreshing change to the subject where projects and assignments are small and unrelated to each other and the quality of the whole thing is just not something you can be proud of because of the short amount of time available.
- I definitely don't regret choosing this subject as one of my electives, and I know that it will help me greatly with my future ambitions of software development.
- The entire subject has been interesting and enjoyable, it was a challenge since I was doing another very time consuming subject at the same time (Software System Design). I believe it was highly beneficial because it help me understand the SSD system on a much higher level, not only that but Software Architecture also allowed me to provide a greater level of input in the design of the SSD system. Thus it was very helpful, by doing both at the same time, I felt I learnt much more, as I put theory into practice, not once but twice.
- It has come a long way from the start of the semester to the end of the semester. From not knowing anything about software architecture to actually knowing something about it, was something that I have accomplished through doing this milestones. From doing those milestones (milestone

1-3) I have learned what is involved in software architecture. To me software architecture is totally a different experience from the other subjects I did. I was first thought it was similar to software engineering that I did in previous semester. Software engineering is about developing a software project whereas software architecture involve in analyzing the system using tools such as use case map, conceptual, execution and implementation diagrams.

- About this subject, I really like this subject as this subject goes straight to the point how to design architecture for a software system, which is very useful for my career in the future.
- Even Through I have almost completed the whole subject I find it difficult to define what is architecture? Yet I have come to understand what makes good software architecture, which is very practical. Understanding the customer needs is one important aspect, another is experience, and the others are simulation and use-cases. Use cases in that good architecture is adaptable to possible future use cases.
- The executable prototype required a great deal of thought and time. A perfect correlation between the architecture described in the software architecture document and it, was ideal. At first not all functionality depicted in the architectures previously established in milestone 2 proved a significant issue. Nevertheless, adjustments made to the software architecture document and executable prototype were necessary to ensure the feasibility of WEWS.
- When I started the subject I was unclear on the concepts taught in the lectures and uncertain as to how the subject could be applied in practical software engineering. Now I have a much greater appreciation of software architecture, since I have applied it to a real software development project myself, and have seen its benefits and uses. I am now certain that having conceptual, execution and implementation views at the start of a software development project would streamline the rest of the process of development, and produce a better end-result.
- Personally I am extremely happy with the knowledge I have gained this semester and have been able to link the theory with the practical application of Software Architecture. I have gained a lot of experience in Java programming including the use of multiple processes, threads and socket connections. My GUI design skills have also dramatically improved with the ability to implement complex pop-ups, panels, image icons for buttons and the ability to create an image panel including the updating of pictures.
- Thus I guess we've concluded that architecture is indeed mastered only through practice. I think we've also learned that there is no hard and fast way of doing architecture, but indeed is a mindset rather than a rigid process.
- After completing this final milestone, I felt a sense of relief and perhaps some sense of accomplishment with our executable prototype up and running. My partner and I were able to produce a product that was fully

functional and followed our software architecture fairly closely. I think the incremental development of this whole project was fairly well structured as it gave our team a goal and task to complete every couple of weeks. This helped me significantly in managing my time since I have a habit of leaving things close to the due date. I know that if all three increments were merged into one large project, I would have a very difficult time completing the project and that my time would not have been used as effectively.

- Overall, I believe this project has been very fulfilling and thought provoking for my career as a Software Engineer. It has taught me about an important stage in the software lifecycle and how to approach the problem in a methodical manner. However, the most important lesson is probably the significance software architecture has on any software system and how it impacts software lifecycle as a whole. Software can be developed without any architectural planning but it would likely result in unmanageable code that is resistant to future maintenance requirements.
- In the course of this subject I have learnt that planning is an absolute necessity. Without planning, difficult problems such as the issues addressed by the assignments would be very hard to solve. Introducing the aspect of architectures allows for a methodical structured approach to solving these problems. When given the problem and told to solve it from scratch, one may find this possibly impossible as there are an infinite set of solutions and the solution created from jumping straight into the coding will most likely not be the most efficient one for this problem.
- I would say that the main thing I learned from this milestone was that the most difficult part of building a software system was designing and implementing the architecture. As I mentioned previously, adding in the functions did not take very long in comparison to the time spent putting the infrastructure together. This will be something I will remember for future projects. Not that I should plan things out (of course I already know that), but rather I should first try to put together the skeleton of the system together in the form of interconnected black boxes, followed by adding in the functions. This will allow me to focus more on the whole issue rather than the immediate problems.
- Overall, I found that while completing the assignment was a demanding task, it has helped me learn a great deal more about the concept of software architecture through applying what I have learnt and seeing how the architecture is used in a small scale project. I believe that I have learnt much more from doing the assignment than what I would have learnt had I just gone to classes and sat an exam, and I have picked up some new skills that I am interested in applying to other works, such as the conceptual, implementation and execution diagrams.
- Throughout the progression of designing and implementing the Audio Magic System, coming up with a conceptual architecture was one of the most difficult phase because we hardly had any understanding of the system requirement as a whole. At first, I thought it was quite confusing but the outcome was rewarding because we were able to analyse the way the

system could be designed and were able to look at the system from different point of views just like the way we look at a building from different angles and heights.

- This subject has taught me how a real system should be designed with great emphasis on analysis and design of the project. This subject has helped me to develop an awareness of responsibilities with respect to the quality of the software systems. I have learnt to understand key issues in implementing complex distributed and real time system, implement architectural prototypes using off-shelf components and develop and refine multiple views of software systems architecture.
- Apart from technical skills, I also developed management skills. Documenting the system required competence in written and oral professional communication. We focused on developing not only the technical ability but also to effectively convey the outcomes to others. Most importantly I improved my interpersonal skills, my communication skills, and leadership skills. I had also learnt how to prioritise my workload with other subject. This was definitely a great learning experience and I am sure it will be very useful in my industrial training and my future as an engineer.
- I really liked the subject, however the sections from the labs and theory was never related, so I found really hard developing the documents, because I never had enough examples and good practice about what I learned in the lectures. The subject is absolutely abstract and hard to understand. Definitely developing software always will be seen as each mind's creation attached to standards. There must be another way to create software in a better way, it must be fun and inspiring, not boring. This subject made me realize that I have to invent a new way to develop software for the next generations.
- The most interesting part of software architecture for me was the fact that there was no one correct solution to creating a software architecture. There could be two or three or maybe more answers to the same problem. Taking the best pick amongst them was almost as challenging as creating the architectures. Quite a few times during the course of our development we were tempted to change our architecture for various reasons like for ease of programming or thinking that maybe we had made a mistake. This was another critical learning that in real-time systems it would be far more crucial and expensive if a wrong decision had been made regarding the architecture.

5 Reflection by the instructors

5.1 JohnR

Strangely enough, I don't have a real lot to say about this semester! Lian took care of a good part of the lecturing while I focused on the online support and on trying to make the subject materials bullet-proof. The materials this semester were reasonably solid, and everything went pretty much without a hitch.

Overall, I think that the quality of the assignments submitted and of the final programming deliverable were higher than last semester, and absolutely streets ahead of earlier semesters.

Survey results

Given the generally high quality of the student's work, I'm a bit mystified by the luke-warm survey results. The course materials are better than they have ever been, the structure and objectives are clear (or as clear as I know how to make them...), and we provide as much support as we could reasonably be expected to.

I think that perhaps the survey has outlived its usefulness now as an indicator of what is right and wrong with the subject. For example, question 4, "There were appropriate resources available to support the subject" received a fairly dismal 3.44. But I would like to see a subject with more resources than we provide:

- A free (draft) textbook
- Copies of all lecture slides
- A workbook with exercises and additional suggested readings for each module
- An online discussion board on which we are fairly responsive
- A nice thick reader with excerpts from over a dozen different books on software architecture

All but the last are provided to the students for free. The last one, the Reader, cost only \$13. Here's the kicker: *only 4 copies of the Reader were purchased!* Yep, students complain about "not enough resources" but don't even bother to spend \$13 to buy (and read) a substantial collection of supporting material.

The written comments in the survey did indicate that some students are still having trouble with understanding what is required in the assignments. I guess this is a problem with trying to assess an incrementally-developed artifact (combined with what seems to be an unwillingness in many students to simply engage with the material and have a go). Nonetheless, for next semester I'll probably just make each deliverable "final" in some sense i.e. the analysis and conceptual architecture in one, the execution and implementation architecture in the next, and the program in the third. This is *not* how it's really done, and so I think we will lose something here—but you can't win 'em all...

Infrastructure

Thanks to the sterling efforts of James Lucas, the adoption and use of subversion as the source code repository went off with barely a hitch this semester.

James has also been assisting me with the purchasing and setup of a more realistic development platform, in the form of SAiL, the Software Architecture Laboratory. This consists of a small number of OS/X machines, and some external hardware to generate video and audio signals. Next semester, some students will be able to use this lab and should find this a more realistic deployment platform than the normal lab machines.

Course changes

Apart from the introduction of the distributed Unix (-like) lab, there are no major changes expected for next year, just a few refinements.

As mentioned above, we should probably simplify the assignment structure (again). Following the Unified Process is a neat idea but it turns out to be a little hard to mark an artifact that is developed over several iterations.

Overall

Overall, I think the subject has stabilized and is fairly mature. Not everybody will agree that about what should be in it, but just read through the student's thoughts in the previous section and I think you'll see that most of the students found something of value in the subject.

JohnR, December 2005

5.2 Lian

My major contribution to the subject this semester was presenting the first six foundational lectures. The lecture slides are artfully prepared with strong visual elements, allowing one to talk around them quite flexibly. I found that most students were listening and prepared to engage in discussion during the lecture sessions.

I think students do like having the tutorial and lab sessions centred around the assignment, and there is plenty of opportunity for them to discuss their assignment progress with the instructors.

I was also in an apprentice teaching role, in preparation for taking over the coordination of the subject next semester. There is a fair amount of behind-the-scenes work that goes on, mostly handled by John this semester, that I am a bit nervous about. Including the web forum softwarepractice.org and the labs.

Lian, December 2005