EECS 4314 - Advanced Software Engineering
Assignment Descriptions

1. Descriptions

The purpose of all these assignments is to give you an appreciation of navigating and understand the software design and architecture in the context of a large-scale software system. You will create a detailed architecture report for a large-scale software system. The report will be updated throughout the term and will be posted online on your group's webpage. After completing all the assignments, you would have:

1. Gained an overall view of the architecture of a large-scale software system,
2. Recognized the strengths and limitations of the architecture,
3. Verified the architecture against the actual implementation (i.e., conceptual vs. concrete),
4. Proposed an enhancement or a new feature based your gained understanding,
5. Described the steps needed to implement your proposal,
6. Compared alternative implementations,
7. Presented all your findings and proposals to your coworkers and the management team.

2. Assignment Overview

There will be a total of four assignments. All them will be done in groups.

- Assignment 0 (A0) – creation of a website about the studied software system.
- Assignment 1 (A1) – recovery of the conceptual architecture based on reading documentations.
- Assignment 2 (A2) – recovery of the concrete architecture from source code
- Assignment 3 (A3) – comparison between the conceptual and the concrete architecture and explain the differences
- Assignment 4 (A4) – development of architecture dependency extraction techniques from source code and comparing differences among alternatives.

You will have to present your findings for Assignment 1, 2, 3 and 4 in class (see Appendix B for presentation guidelines).

3. Assignment Deadline

The table below shows the dates for each deliverables. **All the deliverables are due before class (1 pm)**. Note that:

1. For each deliverable, a single group submission is required (i.e., only one submission from your group).
2. If necessary, a confidential peer review email can be sent to the Instructor's email address 24 hours after the deliverable date (see Appendix C)
3. Do not make your report longer than necessary. Write clearly. Organize your report so it is easy to find things in it. Avoid repetition. In general, shorter reports that contain appropriate information are preferred. Your report must not be longer than 15 pages (in the hard copy version), including all diagrams and appendices. Reports are to be interlinked.

4. Like much work in software development, you are being asked to do more that you have time to do. Part of your job is to choose what you will actually do for these deliverables.

Category	Description	Marks
A0	Group List + Website Links	0%
A1	Conceptual Architecture Presentation	7%
	Conceptual Architecture Report [15 pages]	8%
A2	Concrete Architecture Presentation	7%
	Concrete Architecture Report [15 pages] and Artifacts	8%
A3	Discrepancy Analysis Presentation	7%
	Discrepancy Analysis Report [15 pages] and Artifacts	8%
A4	Dependency Extraction Presentation	7%
	Dependency Extraction Report [15 pages] and Artifacts	8%
Total		60%

4. Assignment Details

Assignment 0: Group Member List and Webpage Links

For the first deliverable, you are to create a webpage with various links (e.g., tutorials, documents, books, discussion groups, group policy, and source code location) about the studied software system. The webpage should contain sufficient information to help a newcomer to a software project get up to speed.

Submission Process: By the deadline date (before 1 pm),

1. You should email the instructor with the following information:
    a. the name of your group,
    b. the names of the group members, their emails and student IDs, and
    c. the URL of your webpage to the course account.
2. On the group webpage, you need to display the name of the group and the list of group members

Assignment 1: Conceptual Architecture

For this deliverable, you will submit a report (10 - 15 pages long) that gives the abstract (i.e., conceptual) architecture of the studied software system (see Appendix A). Your report should be posted online. It should be web-readable, with appropriate links to web-accessible related content. The report should contain sufficient information to help a newcomer to a software project get up to speed. Your reports should present the conceptual structure of the software system rather than discuss the details of its implementation. When presenting the conceptual architecture you should include information about:

- What the system does (its functionality) and how it is broken into interacting parts? What are the parts? How do they interact?
- How does the system evolve?

- What is the control and data flow among parts?
- What concurrency if any is present?
- What are the implications for division of responsibilities among participating developers?
- At least two sequence diagrams (or state diagrams) to explain the flow of noteworthy use cases using the presented conceptual architecture.

Submission Process: By the deadline date (before the class begins),

1. You report should be accessible on your group's webpage.
2. You should submit a zipped copy of your report via your Prism/Linux account using the following command to submit:

   submit -l 4314 a1 .

Assignment 2: Concrete Architecture

For this deliverable, you will submit a report (10-15 pages long) that gives the as-built (concrete) architecture of the studied software system (see Appendix A). Your report should be posted online. It should be web-readable, with appropriate links to web-accessible related content.

You will need to extract the architecture of the software system from Understand (https://scitools.com/) and group the top-level entities of the system into subsystems (and those subsystems into subsystems if necessary). Make sure you describe the process you followed to create your own containment file (CustomizedFileDependencies.contain).

Your report should focus on the architecture of one of the top-level subsystems of the studied software system. Choose your subsystem to be distinct from the subsystems chosen by other groups. Ideally, the set of groups together will cover all the main top-level subsystems of the studied software system. You need to claim your subsystem under study to the course forum. If one subsystem is claimed, your group have to pick another subsystem to study. Your report should describe the used architectural styles and list any noteworthy design patterns.

You should include diagrams of the concrete architecture based on the tools that are presented during class. You will need to redraw the diagrams to make your presentation cleaner and easier to follow. You should present noteworthy aspects of the architecture and its subsystems using sequence diagrams (or state diagrams).

Submission Process: By the deadline date (before the class begins),,

1. You report should be accessible on your group's webpage.
2. You should submit a zipped copy of your report along with your containment file (CustomizedFileDependencies.contain) and the landscape file via your Prism/Linux account using the following command to submit:

   submit -l 4314 a2 .

Assignment 3: Discrepancy Analysis

For this deliverable, you will submit a report (10-15 pages long) that compares your concrete architecture from assignment 2 against your conceptual architecture from assignment 1 (i.e., perform a reflexion analysis). You need to complete the following two tasks:

- Investigate and report the rationales for any discrepancies at two levels: the top subsystem level, and the detailed design level for the subsystem of your choice.
- Propose ways to mitigate the discrepancies by either modifying the conceptual or concrete architecture or both. Please also explain your rationale on why you propose such fixes. Please make it clear what has been changed in terms of conceptual architecture (from A1) and concrete architecture (from A2).

Your report should be posted online. It should be web-readable, with appropriate links to web-accessible related content.

Submission Process: By the deadline date (before the class begins),,

1. You report should be accessible on your group's webpage.
2. You should submit a zipped copy of your report along with your containment file (CustomizedFileDependencies.contain) and the landscape file via your Prism/Linux account using the following command to submit:
   submit -l 4314 a3 .

Assignment 4: Dependency Extraction

For this deliverable, you will submit a report (10-15 pages long) that investigates the pros and cons of various dependency extraction techniques. Your report should be posted online. It should be web-readable, with appropriate links to web-accessible related content.

You will need to compare the following three architectural dependency extraction techniques:

1. the artifacts that you have extracted through the Understand (https://scitools.com/) tool;
2. the program dependency extracted using the "include" directives; and
3. ***at least*** one other technique for dependency extraction.

You will need to implement software programs to realize the "include" dependency technique (part 2). For part 3, the technique can be either leveraging the data produced by another tool (e.g., srcML) or write a program using some APIs (e.g., using the JDT APIs http://www.eclipse.org/jdt/ for Java-based source code or Clang https://clang.llvm.org/ for C/C++-based source code). Make sure the output format from (2) and (3) conform to the TA format (http://plg.uwaterloo.ca/~holt/papers/ta-intro.htm) so you can compare the data produced by three techniques automatically.

Your report should first briefly describe your implementations to realize part (2) and (3). Then it should provide a quantitative analysis on the comparison of the three techniques. You should report various statistics like the total number of extracted dependencies from the three techniques and the number/percentage of commonalities among them. Finally, you should also conduct a qualitative analysis on the comparison. As there can be many differences, it would not be possible to investigate them manually one-by-one to understand the rationale of the differences. One approach is to leverage the statistical inference techniques (http://www.surveysystem.com/sscalc.htm) and only examines a

sample portion of them. Make sure you report performance measurements like precision and recall for your analysis (https://en.wikipedia.org/wiki/Precision_and_recall).

Submission Process: By the deadline date (before the class begins),

1. You report should be accessible on your group's webpage.
2. You should submit a zipped copy of your report via your Prism/Linux account using the following command to submit:

   submit -l 4314 a4 .

Appendix A: Report Content

Your report should be single space 12 point font. The reports should be 10-15 pages long. The report should include the following information:

Title:
About 3 to 6 words making clear what your report is about.

Authors:
List of authors, their email addresses, date, etc.

Abstract:
About 2/3 of a page giving an overview of the key points and findings in your report. This is targeted to a manager or a developer.

Introduction and Overview:
About 1 to 3 pages summarizing the purpose of report, its organization, and its salient conclusions. A person should be able to read just the abstract or just the introduction and have a good idea what is in the rest of your report.

Architecture:
Give the overall structure of the studied system, with descriptions of each major component and the interactions among them. These components are to be primarily subsystems or modules, but may also include threads or processes, files, and databases. In your descriptions, concentrate on goals, requirements, evolvability, testability, etc., rather than on lower level concepts such as classes, variables and control flow. Discuss any parts of the system that are performance critical, i.e., which might not run fast enough. Discuss how the architecture supports future changes in the system.

You should clarify global control flow, such as units of concurrency and method of passing control from one component to another.

Your system's architecture should be easy to understand, with simple interfaces, and modest interactions among subsystems and modules. Clarify the architecture style (in the sense of Garlan and Shaw) that characterizes the overall system and its various parts.

You are not to concentrate on minor components, such as classes and procedures, which are smaller than packages or modules. However, you may wish to discuss important abstractions, patterns, classes, data structures or algorithms that are critical to the success of the architecture.

Diagrams:
You should use diagrams that clearly illustrate the structure of your system. You may choose to use various kinds of UML diagrams. Do not give diagrams that are too low level, e.g., do not give diagrams of details within objects, nor within modules. You can use tools such to help you draw diagrams, but this is not required. Do not include diagrams that are difficult to read. Make sure that all your diagrams have a clear legend explaining the meaning of the various arrows and symbols.

Use diagrams as needed in your report. You are to include a "dependency diagram", in which the boxes are subsystems and modules and the arrows are dependencies. A dependency from module M to N could be due to a call from a procedure in M to one in N, or a reference from inside M to a variable in N, etc. These diagrams can be created using any method you choose.

External Interfaces:
List information transmitted to/from the system, such as by Graphical User Interfaces, files, databases, messages or networks. Do not give details such as menus, but rather concentrate on information content. Although information may be stored in a database, the exact form of this data need not be a concern.

Use Cases:
Give a small number of essential Use Cases which illustrate the use of the system or your proposed feature. Show or explain how each Use Case activates or uses the various major parts of the current or proposed architecture.

Data Dictionary:
Include a glossary that briefly defines all the key terms used in your architecture, giving when appropriate, the "type" of the item being explained.

Naming Conventions:
List any naming conventions used in the described architecture. Explain any abbreviations that you use.

Conclusions:
A summary of your key findings and proposals for future directions.

Lessons Learned:
Document any noteworthy lessons: things you would do differently or things you wished you knew ahead of time.

References:
List any documents that your reader may wish to or need to read in conjunction with your report. Since the report is to be web-readable, include links to references when appropriate.

Appendix B: Presentations

As a software architect, you need to have the charisma and communication skills to communicate your architecture and your findings to the developers who will implement it and to your managers to get funding.

In addition you must be willing to:

- take a fresh and unbiased look at your own work,
- get constructive feedback from others, and
- backtrack and reconsider.

The presentation of your analysis is your opportunity to showcase your findings and share ideas with your peers. Your target audience is a team of developers who are responsible for the analyzed project and who would implement your proposals.

Style

Each group will be given a 20 mins slot (*including the setup time*) to present their assignment in class. The slides must be put online on the group's webpage. Other groups are expected to comment and critique the findings (5 mins).

Recommendations

1. Rehearse and time your presentation ahead of time.
2. Make sure you leave enough time for at least 1 or 2 questions.
3. Use slides and make sure you have your architecture diagrams on the slides.
4. Explain and put a diagram legend on your slides.
5. Articulate the principles and key mechanisms used by your group to reach your findings.
6. Present some modeling alternatives that were considered and explain why they were rejected.
7. Highlight any tradeoffs.
8. Be prepared to defend your architecture decisions.
9. Provide constructive feedback to other groups.
10. Document any noteworthy lessons: things you would do differently or things you wished you knew ahead of time.

Students must participate in all project presentations; missing the presentation slot for a deliverable will result in a 25% reduction in your mark for that presentation.

Appendix C: Peer Evaluations of Your Group Members

Why peer evaluations?

Your group will have to decide how to organize itself. If you are amazingly lucky, your tasks will split naturally into an equal-size task per person, with obvious clean boundaries between them, and the people in the group will have equal skill levels, and everyone will get along so well that you'll want to form a startup company right after exams. More likely, however, you will have to divide labour in some less trivial way, have some people assist others in emergencies, co-ordinate mismatching schedules, and so on.

We expect that all members of your group will receive the same grade for all assignment deliverables. Because not everything in life works out evenly, we will allow for some accounting of this lumpiness. We have had some problems with a student's not participating very much or even at all in his or her team, expecting that the rest of the team do most or even all the work and he or she gets the same grade as the rest of the team. In an effort to discourage this sort of laziness, we have instituted a way that a student's peer evaluation can affect his or her entire mark for the assignment: At the discretion of the instructor of the course, a student's mark for any assignment deliverable may be multiplied the percentage that his or her average peer evaluation is of the total possible. Thus, for the final deliverable, if your evaluation is 1 out of a total possible of 5, you may find your mark (out of the 10 possible for the final deliverable) multiplied by 1/5. Moreover, in the extreme case of student's not participating at all with his or her team in a part of the assignment, the instructor may decide to give a 0 (zero) mark to that student for that part of the assignment. The moral of the story is: *participate, participate, participate!!!*

Therefore, we want each individual to prepare a confidential evaluation of your group members (including yourself). You will prepare three such evaluations, the first will be due within 24 hours after you hand in A1, the second will be due within 24 hours after you hand in the A2, and the third will be due within 24 hours after you hand in the A3. You will email the evaluations to the instructor. **Please send only plain text. No HTML!**

The group evaluations will help us track the progress of your group for the assignments.

How does it work?

You are to prepare an email message to be sent to the instructor (*once again, please send only plain text; no HTML!*). In your message you should clearly indicate your full name, the full names of your group members, what your group number is, and which assignment you are doing. Please send the evaluations of your group members as a single email message. In the message you send, include the exact phrase "Peer Evaluation for Group NN" in the subject line, where NN is your group name.

What should the evaluation look like?

Prepare a paragraph or two on the contributions of each group member. Try to be as objective as you can. Did they attend all of your meetings? Were they on time for the meetings? Was their work done on schedule/as promised/with due diligence? Did they take on a leadership and/or organizational role?

Were they easy to work with? Did their work require extra checking by the rest of the group? Did they stay late when extra effort was required?

Be sure to evaluate yourself as well. Be as honest as you can.

As part of your evaluation, assign each member a numerical grade. You will be given 5 marks per person in your group to distribute as you see fit, plus one bonus mark if you think it appropriate. That is, a group of three will have a up to sixteen marks to distribute between the group members.

How should you distribute the grades?

Here are some rough guidelines on how you might want to distribute the marks:

- A mark of one or two indicates someone who is an unmitigated disaster. You rue the day they were born, and hope you never end up with someone like this on your team when you work in industry.
- A mark of three indicates someone you really wouldn't want to work with again. There are serious shortcomings to their abilities and/or attitude. (S)he is the sort of person you will probably run into in industry, but will try to avoid as much as you can.
- A mark of four means that a group member did reasonable work, but lacking in some area. Their work was OK, but not great, or they weren't serious enough, or ... (S)he is the sort of person with whom you could work without too much trouble, although you wish they were a bit smarter/more diligent/easier to deal with/what have you.
- A mark of five indicates a group member did a really fine job on everything that was asked of him/her. (S)he showed up on time to meetings, did their work on time and well, and chipped in as needed. (S)he is the sort of person you would hope to work with in industry.
- A mark of six indicates someone took on a leadership role, did extra work that others did not, perhaps picked up the slack from a member who didn't perform as well, and went well above and beyond the call of duty. You really hope that you'll get at least one person like this on your team when you work in industry.
- A mark of seven indicates someone who leaps tall buildings in a single bound, walks on water, and solves NP-complete problems in linear time. You hope your company has at least one person like this, and that (s)he knows you by name so that there is a chance you might one day get to work on a assignment/project with him/her.

To re-iterate: the maximum number of marks you award must total to no more than N*5+1, where N is the number of people in your group. You may award fewer if you wish.

Note that we will not simply take an average of the submissions to determine your mark. Instead, we will use our common sense based on having read all of the evaluations. Thus, if one evaluation seems out of line with the others, it will not unduly influence your grade (so don't just give yourself 7 marks and think we won't notice!)

Notes

- If you are having serious problems within your group, don't just wait for the evaluations to roll around. Have a meeting with your group and try to address the problems. If that doesn't work, approach the instructor.
- *Your paragraph(s) of explanation are important. If you do not turn in reasonable justifications of your grades, your own performance evaluation will be a 0 for this assignment.*
- The evaluations are due 24 hours after the major deadlines so that (a) you can get some sleep, and (b) you have a little time to ponder what you want to say. It also allows for some cooling off in case there were some last minute tensions.
- The evaluations will be kept confidential; only the TAs and/or instructor will read them.
- Don't sweat too much about these evaluations. Try to be a good group member and you'll do fine. And remember that in industry, evaluations will be part of your working life.

An example evaluation

Here are some example evaluations for an imaginary programming assignment given on Tuesday and due the following Tuesday.

Candorsnaffity Hixelbröd was an annoying teammate. He was late for both meetings, and had not looked at the assignment or done any kind of preparation in advance. At the Wednesday meeting, Candorsnaffity offered to do the whole project himself, which the rest of us found very insulting --- maybe he's the best programmer in the class, but the rest of us are still good enough to be successful Waterloo students. Finally he agreed that he'd do the sushi-ionization module by Sunday afternoon, but he didn't finish it until almost midnight, and the rest of us couldn't combine his work with ours until Monday evening, which meant we had to stay up late to get it done by Tuesday. I'd give him 2 points, except that on Monday evening (after he finally got there), he found all the bugs that the rest of us were missing and got the assignment working with twenty minutes work. He's probably a great hacker when he's alone: honestly, he's the best programmer of the four of us, and we're all annoyed that he was so little help. So he gets 3 points.

Prosnitau Sgrachita de la Mnirhoihoi was good, helpful and co-operative. She arranged the first meeting and had worked out a module structure and a division of labour before we met, which we adopted after a bit of bickering. She had her parts of the program done on time, and they only had one small bug in them. And she stayed late to make sure that the integration went smoothly, when we all know she hadn't slept the night before. She gets 6 points.

V'snl'chm'tkn!k of the Fortress of Doom (that's me) was an effective worker in this assignment. I had sketched algorithms in advance for two parts, which Wub and I implemented. My code was ready on time, and worked except for one boundary case. Also, I helped Wub with his part of the assignment. I award myself 5 points.

Wub Catcherofahundredsquids was willing and enthusiastic but not very smart. He volunteered for the hardest module, but I had to help him get it working. But he did tell us all on Friday that he couldn't get his part working alone, so at least it wasn't an emergency. He also brought cookies to the Monday meeting, and stayed up extra-late testing the program. I give him 4 points on skill, and 1 more for being helpful: 5 points total.

[These examples were written by my friend, Dr. Samuel Weber of IBM's T.J. Watson Research Lab near New York City. We apologize in advance if, despite our intentions, these names resemble those of people you know. -- MWG]

Acknowledgements

- This peer evaluation write-up is based on a write-up by Professor Michael Godfrey at UW.
- Assignment ideas are inspired by Professor Richard C. Holt at UW and Professor Ahmed E. Hassan at Queen's.