SE 463

Software Requirements: Specification & Analysis

Overview and Admin Notes

Spring/Summer 2020

Daniel Berry

Welcome

- … to Software Requirements: Specification and Analysis
- This course is known as:
  - ECE 451
  - CS 445
  - CS 645
  - SE 463
  - SE 1 (not an official course, just for discussing the courses)
- It is one course of a three-course set on software engineering:
  - ECE452/CS446/SE464 (SE 2): Software Design and Architecture.
  - ECE453/CS447/SE465 (SE 3): Software Testing, Quality Assurance, and Maintenance.

Changes

- Previously, the courses could be taken only in order, as they shared an incremental project
  - SE1 → SE2 → SE3

- In Fall 2008, the three courses were de-coupled, so they can be taken (in theory) in any order

More Changes

- This term's course will be significantly different from those of the past decade
- Many new topics, new readings, some discussions, maybe even a movie
- I will be aiming for more real-world relevance, more realism, e.g., the project's requirements will change significantly as the term progresses, just like in real life.

More Changes, Cont’d

- Your customer will be changing his or her mind a lot, just like in real life.
- I will be addressing agile development and where requirements analysis fits in with it.

Contacting Us On Class Issues

Please use the course account email instead of our personal e-mail addresses, unless it is about a personal or administrative matter:

- se463 ATT student DOTT cs DOTT uwaterloo.ca
  - If there will be a head TA, he or she will read this too.

Dan

Dan

Prof. Daniel Berry

- Virtual office hours: by appointment made by e-mail
- Email: dberry ATT uwaterloo DOTT ca
- Web: http://cs.uwaterloo.ca/~dberry
- Webex: e-mail me to make an appointment
- Piazza: site for this course. Note that I am not very familiar with Piazza. So I will be playing by ear. The best way to reach me is by e-mail.

More about Communicating with Dan

The reason I have no telephone is that I am nearly deaf. I do not sign, but I do read lips. So I cannot use a voice-only telephone. I can use a video communication medium *if the bandwidth of the connection is high enough that the image gets updated at the frequency of television or movies and thus, the lip movement is smooth enough to be decipherable.*

Dan outside of the classroom

- I'm a researcher in the field of software engineering, particularly requirements engineering

- I specialize in:
  - Requirements Elicitation
  - Ambiguity in Natural Language Requirements Descriptions
  - Creativity in Requirements Elicitation

- I dabble also in Electronic Publishing: formatting, typography, etc.

Dan outside of the university

- I swim, skate (both kinds), and ski (downhill snow and water).
- I am considered a good cook.
- I am even semi-professional as a cook, having catered two weddings, one not my own!
- I am a “Star Trek” and a “Big Bang Theory” fan.
- I write scientific satire.
- I write Biblical commentary.
- I have 3 children and 4 grandchildren.
- I love programming.

[Now, do I seem human enough? ☺]

The Course TAs

- Michael Brooks, mlbrooks ATT uwaterloo DOTT ca
- Jeremy Chen, y522chen ATT uwaterloo DOTT ca
- Nikhita Joshi, nvjoshi ATT uwaterloo DOTT ca
- Matthew Lakier, mlakier ATT uwaterloo DOTT ca
- Charles Li, z942li ATT uwaterloo DOTT ca
- Ivens Portugal, iportugal ATT uwaterloo DOTT ca
- Aishwarya Ramanathan, aramanat ATT uwaterloo DOTT ca
- Cristina Tavares, mvtavare ATT uwaterloo DOTT ca
- Rafael Toledo, rftoledo ATT uwaterloo DOTT ca

They will be your mentors, one per SE 490 group.

Your Term Project

- The term project for SE 463 is to write a requirements spec for your planned prototype of your SE 490/CS 493 capstone project.
- Your group's SE 463 TA is also your group's SE 490/CS 493 TA.
- That is, your TA will be familiar with all aspects of your capstone project.

Back to the course

- The materials are online at the course Web site.
- No synchronous lectures. Instead, there are videos at the Web site.
- You will get invitations to attend alive recording sessions. I need an audience to laugh at my jokes ☺ and to ask questions so that I know how I am doing!

Grading scheme

Project	40 %
Assignments	10 %
Final assessment	50 %
Total	**100 %**

This distribution might get adjusted by reducing the percentage of the final assessment and increasing the percentage of the project by the same amount.

Course Web page

http://www.student.cs.uwaterloo.ca/~se463

- Lots of details will appear there over the term, especially the lecture slides and stuff about the project. This takes the place of a textbook, which the course does not have.
- Watch for announcements too.

Course email

se463 ATT student DOTT cs DOTT uwaterloo DOTT ca

- Please send most questions here
  - You may send to Dan questions that relate to course administration or are personal in nature directly.

Course project

The term project for SE 463 is to write a requirements spec for your planned prototype of your SE 490/CS 493 capstone project.

Course project

- Your group will be assigned a TA, who will serve as your mentor and will grade all of your deliverables.
  - Thus, you'll get some consistency in marking.
  - He or she will initially know nothing about the project and will be learning along with you.
  - He or she will give you feedback on your interaction with your customer and on your deliverables.
  - He or she will meet with your group frequently.

Course project

- Your job:
  - to create detailed models of the various entities and processes,
  - to decide what features should be there,
  - to decide the correct functionality of these features,
  - to work out *all exceptions and variations* of these features,
  - eventually, to use these models and decisions to create a specification describing your prototype.

Final Deliverable

The final deliverable is a specification of your prototype in the form of a user's manual (UM), an SRS, a complete set of scenarios, a complete set of UML models, velc. (discuss it with me)

("vel" = "exclusive or" in Latin; so "velc." is to "exclusive or" as "etc." is to "and")

Course software

- You will need to use:
  - A UML modelling / drawing tool
    - We recommend MagicDraw (free to UW students)
    - But you could use MS-Visio, OmniGraffle, dia or ….
  - A document editor
    - MS-Word or OpenOffice
    - OR use LaTeX
- Many more details on course webpage under "This Term's Project Description"

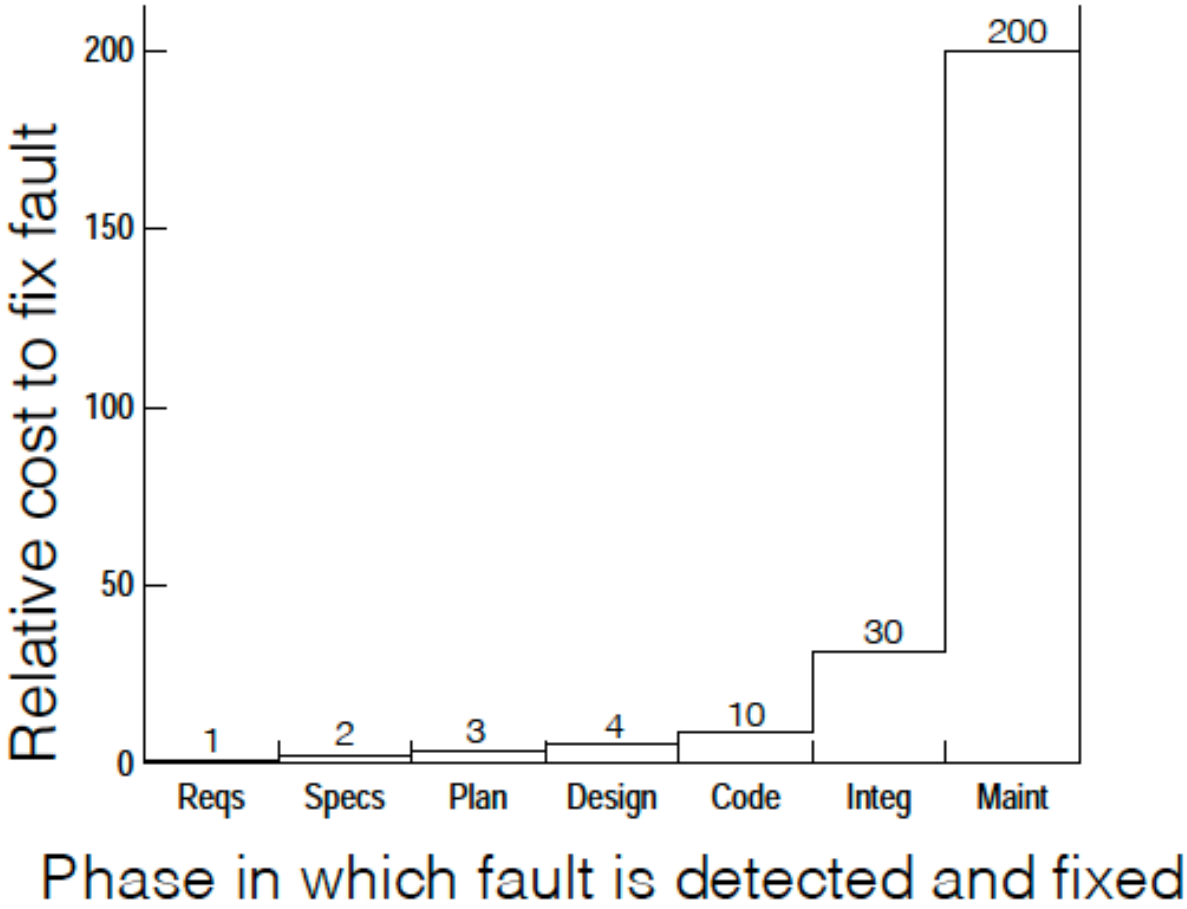
More Details About Project

- Everyone says
- “We *know* that we should work out all the requirements before we start to code,
- but we *don’t* have time!
- We gotta get started coding; otherwise we will not finish in time!”

Wrong!

The problem is that if you start coding before you work out all the requirements, then the cost of correcting the code when a missing requirement is finally discovered is *10-200 times* — depending on when the defect is found — the cost of writing the code with that requirement already specified.

The Data Show



Start Coding Later, Save Time!

A: Starting coding before all the requirements are worked out and specified completely means that you finish coding much later than if you had delayed the starting of the coding until after all the requirements are worked out and specified completely!

Start Coding Later, Save Time!

A: In other words:

- start coding later, finish earlier
- start coding earlier, finish later

This truth goes against every manager's guts, so *no on*e delays coding until after the requirements are completely specified.

But But But…

B: But but but.. requirements keep coming with no end in sight. Users think of new requirements *all the time*. So what difference does it make? We're going to have to deal with new requirements after the coding is done anyway?

That's absolutely right. In fact, both A and B are right! So, now what?

Two Different Kinds of Requirements

You see, each of A and B is talking about a different set of requirements!

- *Scope DeterminiG Requirements (G requirements)* that keep coming and phenomenon B
- *Scope DetermineD Requirements (D requirements)* that are expensive to fix and phenomenon A

Pocket Calculator Example

- Pocket calculator: with +, -, *, and /
- G requirements: +, -, *, /, and **
- D requirement: NZD: “in /, the denominator cannot be 0”

If You Start Coding Too Soon

So if you start coding the G requirement /, and you are not aware of its D requirement, NZD, you will write code that will break if ever / is presented with a 0 denominator.

At that point fixing the code will cost 10-200 times what it would have cost to have just specified NZD upfront so that coding takes it into account from the beginning.

The G Requirements *Are* Different

- Yes, if you now add a new G requirement, particularly one that is not anticipated, there is a chance that it will clash with the existing architecture, and you'll have to do an expensive restructuring.
- But that's unavoidable. And that's the sort of thing iterative and agile methods are designed to deal with.

The G Requirements *Are* Different

- And, if you have to restructure, it will cost 10-200 times more than it would have cost if you had included the G requirement from the beginning.
- There is evidence that throwing out the code and starting all over with all the requirements is much cheaper.
- But no manager's guts permits doing that!

Inescapable Fact Affecting D Requirements

The basic fact is that there is *no* way that you can write any code without knowing what its requirements are, i.e., what it is supposed to do, *even* if you have to decide what the requirements are **as you are coding**.

It's inevitable, like death and taxes.

So the nature of D requirements is:

Once you have picked a scope for your next sprint or iteration, i.e., a particular set of G requirements, the D requirements associated with the chosen G requirements are there *even if you have not written them down*.

The Nature of D Requirements:

If you start coding with them missing from the specification, and you discover their existence during coding, you will have to specify the missing D requirements before you can finish the coding, at 10 times the cost of having determined them before coding.

So the nature of D requirements is:

This is a stupidly expensive way to discover and specify D requirements, because they were already apparent when specifying them was much cheaper.

Worse Comes to Worst

If worse comes to worst, and as very typically, you deliver the code *before* a D requirement is discovered, then a *user* — the best defect finder in the universe — will eventually discover it, …

and it will cost 100 times more to fix it than having written it down up front.

Reality of Your Capstone Project

So for your Capstone projects, you have been likely postponing working out the details of all requirements, because you don't have enough time.

You have probably picked a small viable set of G requirements as the scope of your prototype and are heading into design and coding without having fleshed out the G requirements' D requirements.

You don't have the time!

The Gift of SE 463!

OK.. So what SE 463 is going to do is give you the gift of time to work out the D requirements!

So thank me! ☺

How The Gift Will Be Given

And the way I am going to give you this time is to take advantage of my dictatorial powers of not giving you a passing grade in SE 463 unless you satisfy the course requirements, which is to write a requirements specification of the selected set of G requirements, the selected scope, of your capstone prototype, in which each of the scope's D requirements has been worked out and a response for it is specified.

Aren’t I nice? ☺

You may applaud! ☺

SE 463

Software Requirements: Specification & Analysis

Overview and Admin Notes

Spring/Summer 2020

Daniel Berry