

### last time: Product idea proposal

- First assignment: **Due at Sunday, Feb 1, 9PM**  
<http://www.cs.umass.edu/~brun/class/2015Spring/CS320/productIdea.pdf>
- Groups of 1 or 2
  - get into groups after class or use the Moodle class discussion forum
- Submit 4 slides:
- 3-minute presentations in class next week

Does everyone have a 1–2 person group?

### This Wednesday

- Discussion section (1:25-2:15)  
 Optional: meet to work on proposals in CS142
- 2:30-3:45 Lecture on Teamwork
- 4:00-5:00 Guest lecture in CS 150  
 DLS talk on testing software  
 Free **snacks** right after class!  
**Part of class material** (will be on test)  
 Recorded, in case you cannot attend.

**Distinguished Lecturer Series**

**Automated Support for Reproducing and Debugging Field Failures**

**Alessandro Orso**  
Georgia Institute of Technology

Two extremely challenging software maintenance tasks are reproducing and debugging field failures, failures that occur on user machines after release. We have developed a family of techniques that help developers with these tasks. I will overview our work on this area and present one of these techniques: BugRelex and F3. BugRelex reproduces field failures by collecting dynamic data about failing executions and synthesizing executions that mimic the observed field failures. F3 uses those executions to automatically locate faults. I will present the results of an empirical evaluation on a real-world program and field failures. The results of our evaluation are promising. For all the failures considered, our approach was able to (1) synthesize failing executions that mimicked the observed field failures and (2) use the synthesized executions to locate the fault. I will conclude discussing our latest results, open challenges, and future research directions.

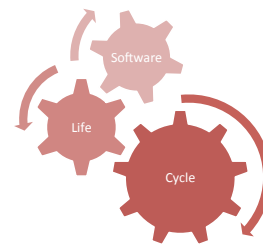
BO: Alessandro Orso is a Professor in the College of Computing at the Georgia Institute of Technology. His area of research is software engineering, with emphasis on software testing and program analysis. He is a senior member of the ACM and of the IEEE Computer Society.

Wednesday, January 28, 2015 @ 4:00pm  
in CS 151

<https://www.cs.umass.edu/speakers/alessandro-alex-orso/2015/jan/28>

**UMASSCS**  
SCHOOL OF COMPUTER SCIENCE

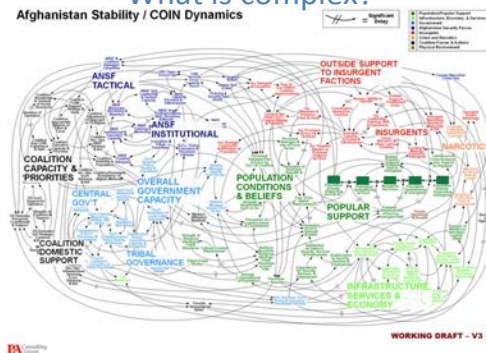
### Software Development Lifecycle



thinking about the process

### How complex is software?

### What is complex?



## How complex is software?

- Measures of complexity:
  - lines of code
  - number of classes
  - number of modules
  - module interconnections and dependencies
  - time to understand
  - # of authors
  - ... many more

## How complex is software?

- Measures of complexity:
  - **lines of code** Windows Server 2003: 50 MSLoC  
Debian 5.0: 324 MSLoC
  - number of classes
  - number of modules
  - module interconnections and dependencies
  - time to understand
  - # of authors
  - ... many more

## How big is 324 MSLoC?

- 50 lines/page  $\Rightarrow$  6.5M pages
- 1K pages/ream  $\Rightarrow$  6.5K reams
- 2 inches/ream  $\Rightarrow$  13K inches
- 13K inches  $\approx$  taller than the Prudential
- 5 words/LoC @ 50 wpm  $\Rightarrow$  32M min  $\approx$  61 years

And we don't just want random words,  
we want compiling code!

## Managing software development

- Requirements
- Design
- Implementation
- Testing
- Maintenance

## Outline

- Why do we need a lifecycle process?
- Lifecycle models and their tradeoffs
  - code-and-fix
  - waterfall
  - spiral
  - staged delivery
  - agile (scrum)
  - ... there are many others

## Ad-hoc development

- Creating software without any formal guidelines or process
- Advantage: easy to learn and use!
- Disadvantages?

### Ad-hoc development disadvantages

- Some important actions (testing, design) may go ignored
- Unclear when to start or stop each task
- Scales poorly to multiple people
- Hard to review or evaluate one's work

The later a problem is found in software, the more costly it is to fix.

### What makes a lifecycle?

- Requirements
- Design
- Implementation
- Testing
- Maintenance

How do we combine them?

### Benefits of using a lifecycle

- provides a work structure
- forces thinking about the "big picture"
- helps prevent decisions that are individually on target but collectively misdirected
- assists management and progress control

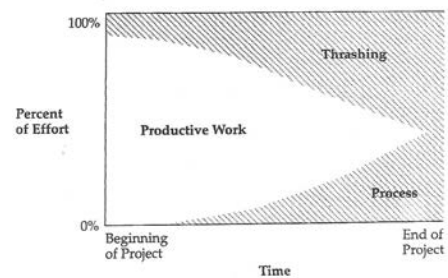
### What are some drawbacks?

### Are there analogies outside of SE?

Consider the process of building the Prudential

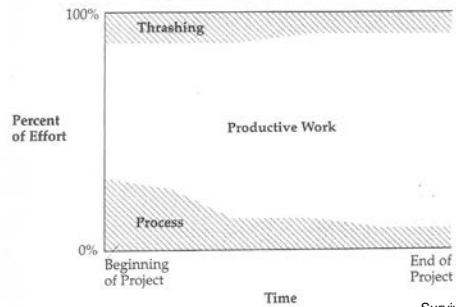


### Project with little attention to process



Survival Guide: McConnell p24

### Project with early attention to process



Survival Guide:  
McConnell p25

### Let's talk about some lifecycle models

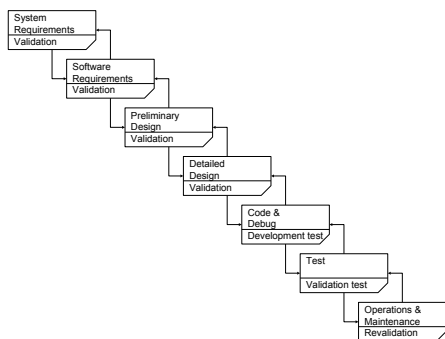
### Code-and-fix model



### Code-and-fix model

- Advantages
  - Low overhead
  - Applicable to small, short-lived projects
- Dangers
  - No way to assess progress and manage risks
  - Hard to accommodate changes
  - Unclear what and when will be delivered
  - Hard to assess quality

### Waterfall model



### Waterfall model advantages

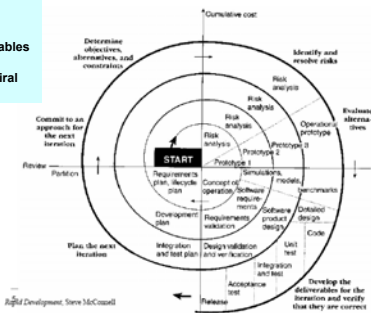
- Works well for well-understood projects
  - tackles all planning upfront
  - no midstream changes leads to efficient software development process
- Supports experienced teams
  - Orderly, easy-to-follow sequential model
  - Reviews help determine readiness to advance

### Waterfall model limitations

- Difficult to do all planning upfront
- No sense of progress until the end
- Integration occurs at the very end
  - Defies the “integrate early and often” rule
  - Without feedback, solutions are inflexible
  - Final product may not match customer’s needs
- Phase reviews are massive affairs
  - It takes a lot of inertia and \$ to make changes

### Spiral model

- Determine objectives
- Identify and resolve risks
- Evaluate alternatives
- Develop and verify deliverables
- Plan next spiral
- Commit (or not) to next spiral



### Spiral model

- Oriented towards phased reduction of risk
- Take on the big risks early
  - are we building the right product?
  - do we have customers for this product?
  - is it possible to use existing technology?
    - tomorrow’s technology?
- Progresses carefully toward a result

### Spiral model advantages

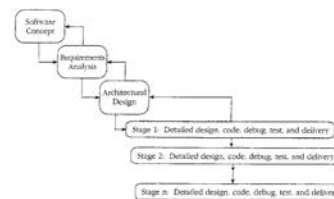
- Especially appropriate at the beginning of the project, allowing requirement fluidity
- Provides early indication of unforeseen problems
- Allows for change
- As costs increase, risks decrease!

Addresses the biggest risk first

### Spiral model disadvantages

- A lot of planning and management
- Requires customer and contract flexibility
- Developers must be able to assess risk

### Staged delivery model



first, waterfall-like  
then, short release cycles: plan, design, execute, test, release  
with delivery possible at the end of any cycle

### Staged delivery model advantages

- Can ship at the end of any release cycle
- Intermediate deliveries show progress, satisfy customers, and lead to feedback
- Problems are visible early (e.g., integration)
- Facilitates shorter, more predictable release cycles

Very practical, widely used and successful

### Staged delivery model disadvantages

- Requires tight coordination with documentation, management, marketing
- Product must be decomposable
- Extra releases cause overhead

### What's the best model?

Consider


- The task at hand
- Risk management
- Quality / cost control
- Predictability
- Visibility of progress
- Customer involvement and feedback

Aim for good, fast, and cheap.  
But you can't have all three at the same time.

### This Wednesday

- Discussion section (1:25-2:15)  
Optional: meet to work on proposals in CS142
- 2:30-3:45 Lecture on Teamwork
- 4:00-5:00 Guest lecture in CS 151  
DLS talk on testing software  
Free **snacks** right after class!  
**Part of class material** (will be on test)  
Recorded, in case you cannot attend.

**Distinguished Lecturer Series**



*Automated Support for  
Reproducing and Debugging  
Field Failures*

**Alessandro Orso**  
Georgia Institute of Technology

Two extremely challenging software maintenance tasks are reproducing and debugging field failures, failures that occur on user machines after release. We have developed a family of techniques that help developers with these tasks. I will overview our work in this area and present two of these techniques: BugRelex and F3. BugRelex reproduces field failures by coloring dynamic data about failing executions and performing executions that mirror the observed field failures. F3 uses those executions to automatically locate faults. I will present the results of an empirical evaluation on a real-world program and field failures. The results of our evaluation are promising. For all the failures considered, our approach was able to (1) reproduce failing executions that mimicked the observed field failures and (2) use the synthesized executions to locate the faults. I will conclude discussing our latest results, open challenges, and future research directions.

Dr. Alessandro Orso is a Professor in the College of Computing at the Georgia Institute of Technology. His area of research is software engineering, with emphasis on software testing and program analysis. He is a senior member of the ACM and of the IEEE Computer Society.

Wednesday, January 28, 2015 @ 4:00pm  
in CS 151

**UMASSCS**  
SCHOOL OF COMPUTER SCIENCE

<https://www.cs.umass.edu/speakers/alessandro-alex-orso/2015/jan/28>