Lecture 21

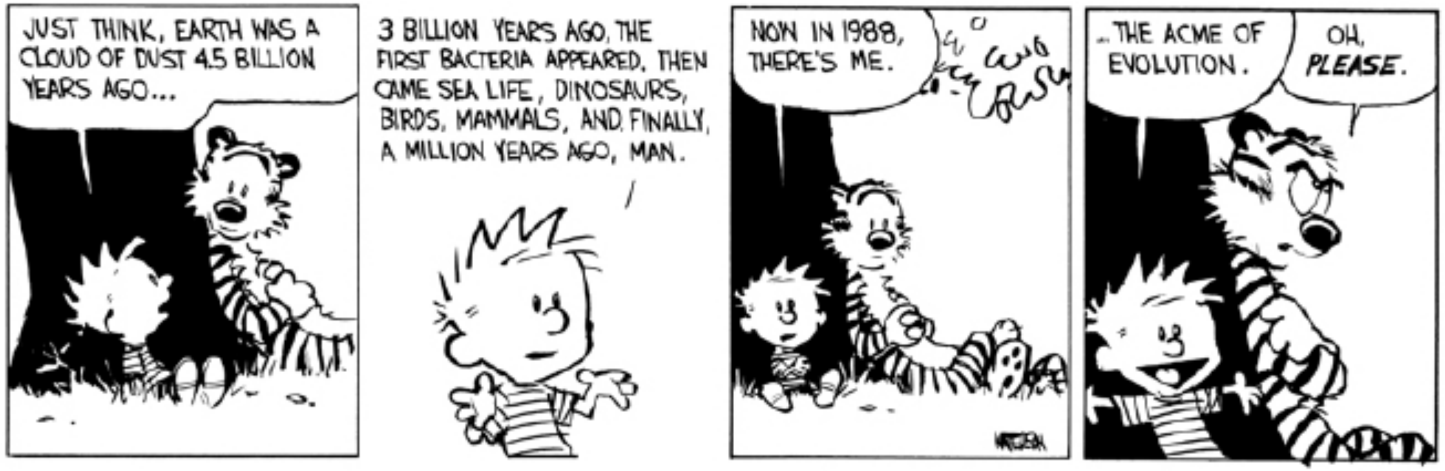
3. *Evolving the Requirements, cont .*

1. *Figuring your grade*

2. *Survey results*

Com S/SE 409/509

Robyn Lutz


JUST THINK, EARTH WAS A CLOUD OF DUST 4.5 BILLION YEARS AGO...
3 BILLION YEARS AGO, THE FIRST BACTERIA APPEARED, THEN CAME SEA LIFE, DINOSAURS, BIRDS, MAMMALS, AND, FINALLY, A MILLION YEARS AGO, MAN.
NOW IN 1988, THERE'S ME.
...THE ACME OF EVOLUTION.
OH, PLEASE.
WATTERSON

Calvin & Hobbes
by Bill Watterson

Figuring your grade (409)

Syllabus:

- Exam 1: 30%
- Exam 2: 30%
- Homework: 20%
- Team Project: SRS (409 & 509) & Presentation (509): 20%

Total % = 0.2 [(HW1+HW2+HW3+XC1+XC2)/300]. + 0.3(Exam1) + 0.3(Exam2) + 0.2(SRS)

XC1: 10 pts. extra-credit (scan & upload before Exam1) if you did it
XC2: 5 pts. extra-credit (survey responses) for all since good (68%) response rate
Final course grade is based on the Total % above & grading scale below
Exam 2 grades use the same grading scale

A+ (unofficial)	>99
A	93-100
A-	90-92
B+	87-89
B	83-86
B-	80-82
C+	77-79
C	73-76
C-	70-72
D+	67-69
D	63-66
D-	60-62
F	< 60

Plus/Delta Survey Results (68% response)

Thank you! very, very helpful feedback!

Keep:

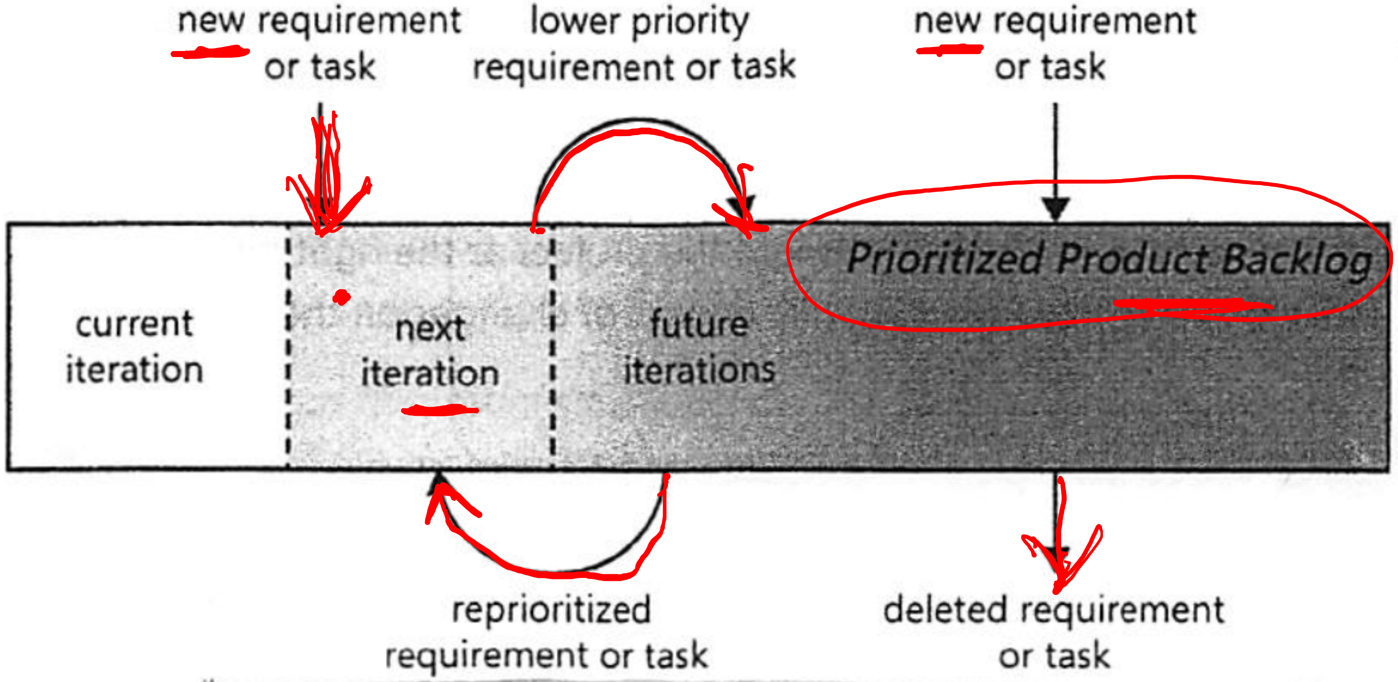
- Lecture videos
  - Short & to the point good
  - Asynchronous good (for most)
- Annotated slides
  - Having slides & videos preferred
- E-book used
- Group work
  - Discussions, collaboration, teammates good
- HW increments toward SRS

Change:

- Lectures:
  - More examples (for many)
  - More material, more in-depth (for some)
- Gradescope:
  - Canvas due dates used (by many)
  - Gradescope/Canvas integration (")
- Also suggested by one/a few
  - More projects
  - Individual assignments
  - More exam practice material
  - Guest speakers
  - Better textbook
  - Speak up
  - More communication

Managing requirements change [Wiegers & Beatty, 2013]

Agile method



new requirement or task
lower priority requirement or task
new requirement or task
current iteration
next iteration
future iterations
Prioritized Product Backlog
reprioritized requirement or task
deleted requirement or task

FIGURE 28-9 Agile projects manage change with a dynamic product backlog.

agile

Managing requirements change [Wiegers & Beatty]

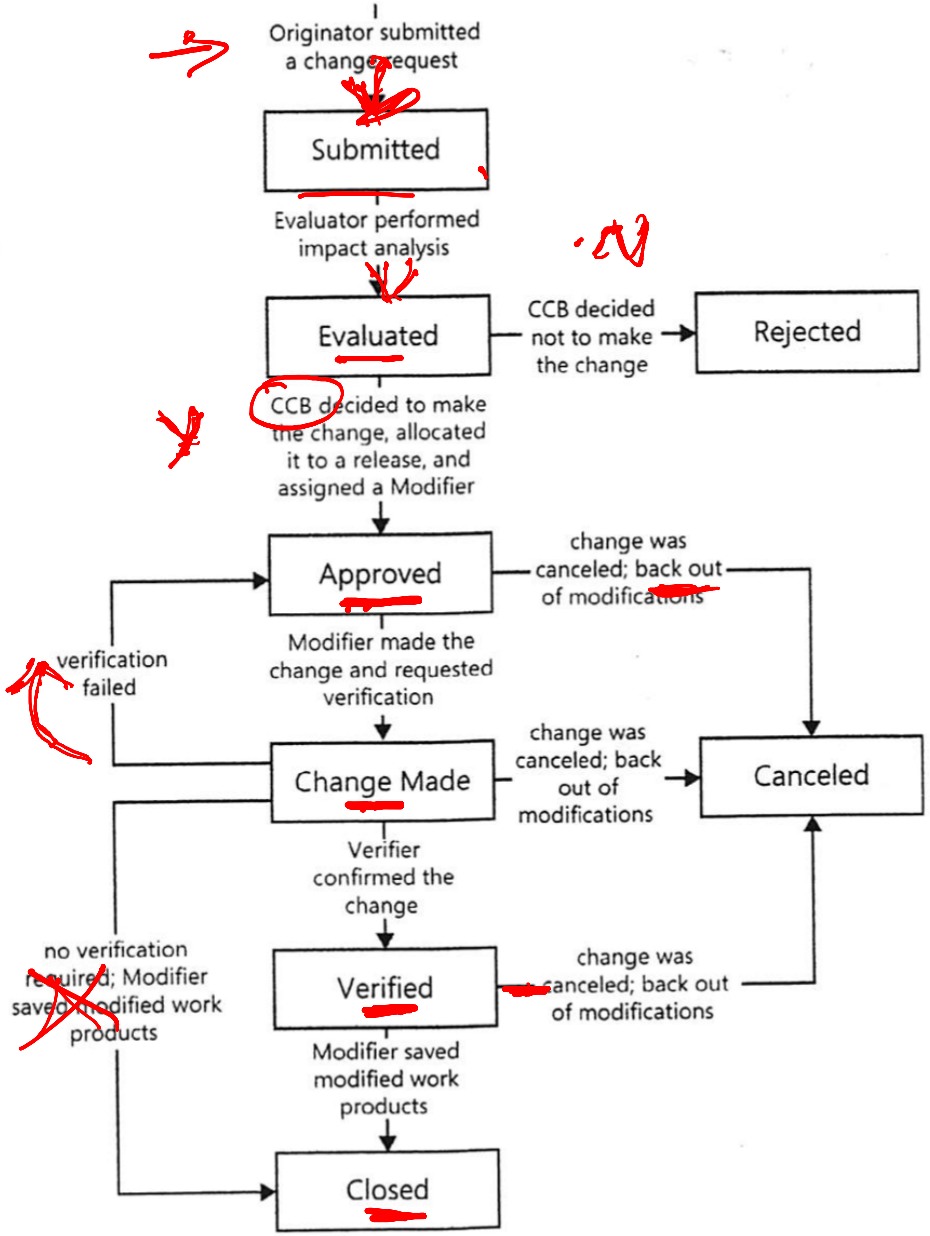
"elephant"

heavy-weight

CCB - ~~Conf~~.

Change Control Board

State-transition diagram



Originator submitted a change request
Submitted
Evaluator performed impact analysis
Evaluated
CCB decided not to make the change
Rejected
CCB decided to make the change, allocated it to a release, and assigned a Modifier
Approved
change was canceled; back out of modifications
verification failed
Modifier made the change and requested verification
Change Made
change was canceled; back out of modifications
Canceled
Verifier confirmed the change
no verification required; Modifier saved modified work products
Verified
change was canceled; back out of modifications
Modifier saved modified work products
Closed

FIGURE 28-2 State-transition diagram for a change request.

Dynamic requirements evolution
[van Lamsweerde, Requirements Engineering, 2009]

- Requirements are specified under specific assumptions about the operational environment (such as number of users)
- Some of those requirements/assumptions will change over time
- * Requirements may need to evolve in response
- Requirements monitoring at runtime addresses this problem ← Solution

Run-time monitoring [van Lamsweerde]

1. At RE time, identify assumptions to be monitored (mission-critical or likely to change. Specify these assumptions as assertions.

- Ex for a meeting scheduler system:
- Assumption: If a participant receives an email request for ~~his/her~~ their scheduling constraints, ~~he/she~~ they will provide them within x days.
- Software requirements depend on this being true
- Likely to be violated for very busy people

2. For each assertion to be monitored, build a software monitor that will run concurrently with the software in order to detect violations of it

- Ex: We build a monitor that detects the absence of response events from participants by the deadline
- This step can be fully automated when the assumption is specified formally in temporal logic

Run-time monitoring & reconfiguration [van Lamsweerde]

3. For each assertion to be monitored, require an alternative path to be taken if the assertion is violated, or is violated too frequently
    - Ex: : alternative courses of action may be to send a reminder to the participant, send an email to their secretary, issue a warning to the meeting initiator, etc.

4. At runtime, the monitor tracks events and if an assumption is violated (invalidating the requirement), it generates information for runtime reconfiguration
    - Reconfiguration may be governed by rules such as reconfigure when the violation occurs for the time.
    - Most simply: a warning & the execution trace that led to the violation
    - More effectively: the generated information is used by the software to reconfigure itself automatically to the alternative course of action.
    - Ex: the monitor detects violations & passes the information to the software for a rule-based decision on an alternative course of action:

- Ex: if the number of times the participant has not returned their constraints within X days over the past Y months exceeds threshold Z
  then get this participant's constraints directly from their meeting scheduler

Run-time monitoring & reconfiguration [van Lamsweerde]

In the example, we require an alternative behavior (a reconfiguration) when the monitor detects that the assumption on which the existing requirement depends has become invalid

Why are we studying this when it's not in the book?

✱ Runtime monitoring for requirements evolution (updating the requirement when an assumption becomes invalid) is essential for autonomous systems—self-driving vehicles, drones, robots

Future

✱ Some systems also learn how to evolve (that is, change) the requirements for the monitor & the alternative behavior