Computer Science
UNIVERSITY OF TORONTO

lecture 18
software development planning

csc302h

winter 2014

Computer Science
UNIVERSITY OF TORONTO

recap from last time

- discussed the top-10 essential practices for software development:
  1. source code control
  2. issue tracking
  3. build automation
  4. automated regression tests
  5. **release planning**
  6. design specifications
  7. architecture control
  8. effort tracking
  9. process control
  10. business planning

Computer Science
UNIVERSITY OF TORONTO

today

software development planning

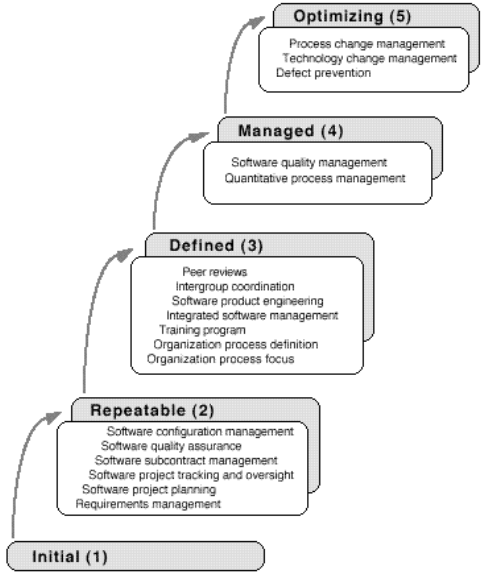
Computer Science
UNIVERSITY OF TORONTO

capability maturity model

- classifies an organization's maturity into 5 levels
  - each level prescribes a group of practices
  - CMM is also a road to process improvement
  - must have all lower-level practices in place before attempting next level
- can be certified to a certain CMM level
  - some similarities to ISO 9000
  - not universally agreed to be a good thing, but is an interesting exercise

capability maturity model (2)

capability maturity model (3)



relationship to ISO 9000

- ISO 9000 is a set of quality standards
  - subset of these are specific to software
  - must document the process
  - must maintain "quality records"
    - used in audits to ensure adherence to the process
  - process can be anything

relationship to top-10

- top-10 practices are necessary to achieve CMM level 2 (repeatable)
- also, top-10 includes enough level 3 (defined) stuff to attain ISO 9000 certification
- and, top-10 even include some level-4 (quantitatively managed) stuff, where most useful
  - defect arrival/departure rates
  - estimate vs. actuals

planning

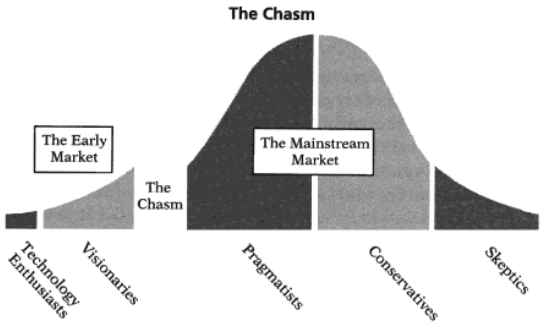
- planning is the most important aspect of CMM Level 2
- common flaws regarding planning
  - making no plans!
  - make a plan, but don't track it
  - attempt to track the plan with inadequate tools
    - Gantt charts
    - Microsoft Project

why plan?

- planning isn't *always* a good thing
  - release/expected date is not important
  - no expectations on new functionality
  - proof-of-concept (a.k.a. "spike")
- planning is required when external pressures come to bear on feature availability dates
- doesn't usually apply to first releases, but is necessary to "cross the chasm"

crossing the chasm

- book by Geoffrey Moore (1991)



planning essentials

What are we building?
By when will it be ready?
How many people do we have?

- answer these questions, and nothing more
  - not "who will be doing what?"
  - not "what are the detailed tasks required?"
  - not "in what order must the tasks be performed?"

Computer Science
UNIVERSITY OF TORONTO

implementation plans

- once initial planning is complete we can transition to a more detailed development plan
- this more detailed plan sorts out:
  - who is assigned to what
  - dependencies between features
  - etc.

Computer Science
UNIVERSITY OF TORONTO

of mice and men

"The best-laid schemes o' mice an' men Gang aft agley"
– Robbie Burns

- the essence of planning is uncertainty
  - plans never "according to plan"
  - must embrace change rather than resisting it
- how to make plans **and** embrace change?
  - track the plan constantly, not just at the start
  - react quickly & decisively to adverse situations
  - embrace a change in direction
  - re-plan quickly, can't be hard to deal with unexpected changes

Computer Science
UNIVERSITY OF TORONTO

internal changes

- estimation errors
  - initial estimates contain a significant (usually one-sided) margin of error
  - as plan progresses, and more information becomes available, variance in errors drops
- developer availability changes
  - illness, parental leave, resignations, cut backs, unexpected vacation plans, unexpectedly low hours of work, unexpected low productivity

Computer Science
UNIVERSITY OF TORONTO

external changes

- new (big) customer with specific demands
- pressure from competition
- collaboration opportunities
- acquisitions & mergers
- sudden changes in customer needs
  - ex. regulatory changes that affect them

Computer Science
UNIVERSITY OF TORONTO

the difficult question

- what are we building?
  - hard for 1st release, later ones have big wish list
  - marketing/product manager pick ones that will get most sales
- by when will it be ready?
  - too soon: customers won't be ready, won't want to learn, install, pay for it
  - too late: competition will pass you, customers will forget you == forgone revenue
- how many developers?
  - usually fixed for a given release, or planning horizon

Computer Science
UNIVERSITY OF TORONTO

the difficult question (2)

What are we building?
By when will it be ready?
How many people do we have?

the difficult question is:

can we do all 3 at once?

Computer Science
UNIVERSITY OF TORONTO

a common problem

- often organizations will answer all 3 questions, but not address the difficult one
- development mgmt. wants to please the rest of the company and agrees to too much – gung-ho spirit!
  - some actually believe in over-commitment to boost productivity – "it's a stretch, but we'll pull it off!"
- developers will say "it can't be done!" – but that's all those folks ever say, right?

Computer Science
UNIVERSITY OF TORONTO

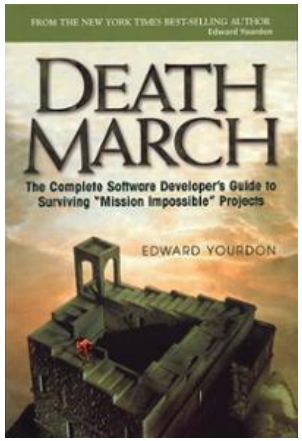
a common problem (2)

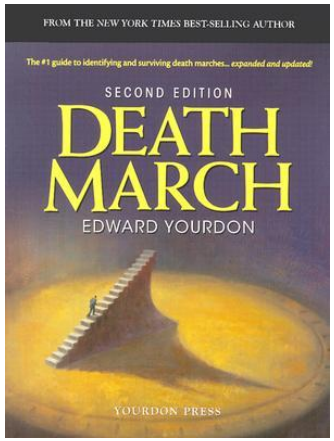
- major state of denial sets in…
  - or sometimes hopeless optimism
  - everybody is secretly hoping for a miracle
- nobody will accept any blame, and why should they?
  - dev. mgmt.: "we told you it was a stretch!"
  - developers: "we said it couldn't be done!"
  - marketing & sales: "R&D, should have said something earlier!"
  - CEO: "you all told me everything was fine!"
  - Yourdon's death march…

Computer Science
UNIVERSITY OF TORONTO

a common problem (3)

- *Death March* – Edward Yourdon





Computer Science
UNIVERSITY OF TORONTO

the solution – good planning

- the "*death march*" doesn't need to happen
- to avoid it we need some courage and conviction
- also need common sense:
  - is it even feasible to do what's asked by the date required?
  - don't give a quick (off-the-cuff) answer even if it's obviously impossible
  - put together a plan to demonstrate the facts.