top-10 essential practices in software engineering

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

software engineering

- it's all about matching process, tools, technology, and architecture to your situation.
  - 40 line throwaway python script for your own use
    - only you will use it
    - only you will contribute to it
    - you will use it for the next 30 minutes and never again
  - a software product you are building a company around
    - 10's of thousands of paying customers will use it
    - eventually a large team will collaborate on it
    - it will survive for > 10 years
  - and everything in between
- there is no one "right way" for any situation

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

new vs. established product

- new product
  - 1 yr. to develop
  - 3 coders, 1 tester, 1 documenter
  - cost = 1 x 5 x $100,000 = $500,000
- established product
  - 5 years later
  - 20 coders, 10 testers/build, 5 documenters
  - cost to date = $10,000,000
  - ongoing cost = $3,500,000 / year
- improve productivity by 10%
  - new product: save $50,000
  - Established product: save $1,000,000 to date, $350,000/year

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

new vs. established (2)

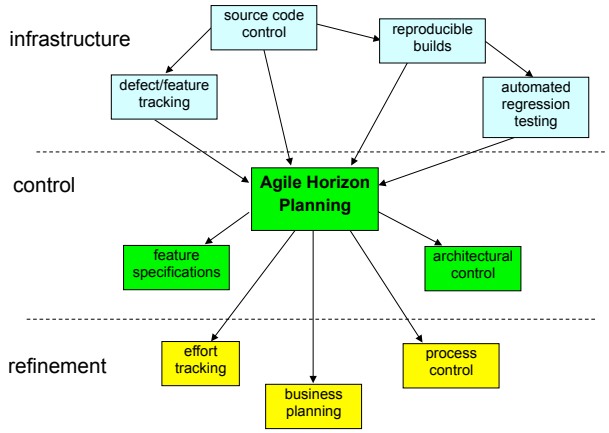
- next release development is more economically important.
- learn how 'next release' is done to setup initial release properly

top-10 essential practices

- crystallized for me whenever I enter into a new engagement.
- if any of these are missing, I know I have something to fix.
- these are all important
- it will take more than this course to cover them all
- you will agree that all suggestions are sensible and will probably vow to carry them out
  - on your first job, you'll focus on code and test and forget most of them
  - you'll be bitten in the ass
  - you'll re-commit to the ideas (if you're good)
- simple but hard
  - trust me: make sure these things are done and everything will go ok
  - very hard to change behaviour
  - need to be dogged and determined and tricky
- geared more towards 'next release' than 'new release'

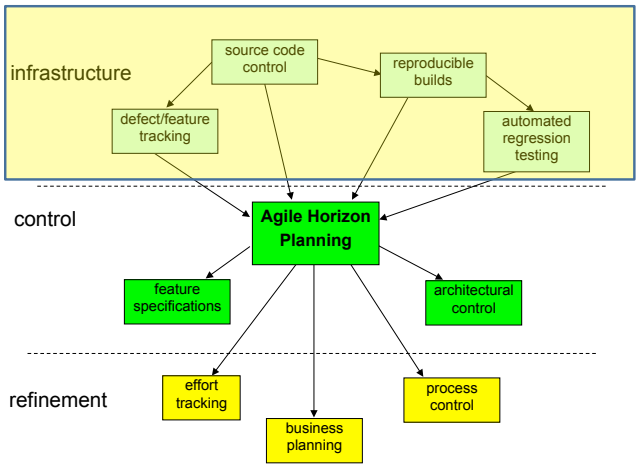
The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

top-10 (2)



The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

top-10 (3)



The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

1. source control

- central repository
  - everybody knows where to find what they are looking for
  - secure, backed-up storage
- defines module architectural structure
  - Hierarchy
- complete change history
  - can back up and find where problems are first introduced
- multiple maintenance streams
  - work on next release while maintaining previous releases
- patches
  - Can go back and patch any release in the field
- enables team development
- "interface" to coordinate dev and QA/build
- "guard" against bad changes

2. issue tracking

- keeps track of all defects found or new features desired
  - won't forget any
- coordinates a workflow for writing / fixing them
  - won't skip steps
- provides management visibility into progress and enables metrics to be gathered
- enables effective prioritization

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

3. reproducable builds

- check out of source control and one command to build the product
- required for a consistent experience across all developers, QA/Build, customers
- dev builds
  - for coding and testing
- production builds
  - includes creation of install image
  - and creation of ISO-Image (if still shipping on disks)
  - should also be fully automated

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

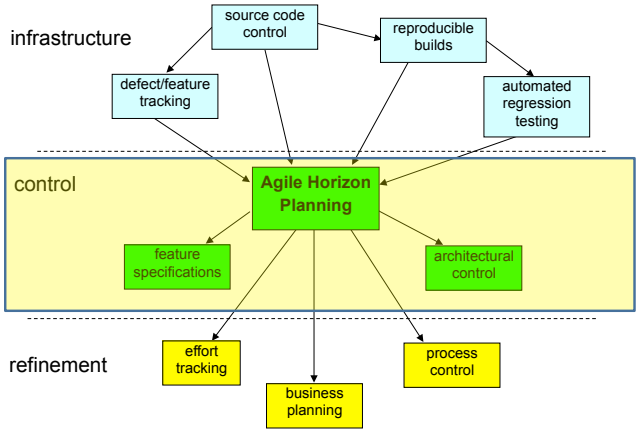
4. automated regression testing

- scripts that run after every QA/Build dev build to test as much functionality as possible
- critical to improving software quality
- prevents errors with previously seen symptoms from recurring
  - a very common thing to happen
- enables coders to change tricky bits with confidence
- enables finding problems closer to their injection
  - earlier you can find an issue the less costly it is to fix.

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

regression testing (2)

- enables fixing last problems prior to shipping with confidence
  - can release with fewer known defects
  - can release on time
- includes automated unit testing
  - developed while code is being written
  - tests classes and modules (collections of classes).
  - good design + dependency injection to replace surrounding infrastructure without recoding



5. horizon planning

- after the previous basics are in place this is the most important practice
  - will spend relatively more of the course on this
- determining
  - what goes into the software
  - by when will it will be done
  - using what resources
- tracking that throughout the time horizon
- adjusting as necessary

horizon planning (2)

- enables business side to do their jobs
  - good relationships
- enables quality
  - by maintaining necessary non-coding periods (e.g., stabilization sprints)
- provides elbow room
  - to improve productivity
- a weakness of many agile methods – end date prediction is somewhat an undefined thing!

release planning

- book used to refer to this as “release planning” – you may find older references to that.
- while the “big bang” release is still used and important, a lot of us are releasing software much more continuously.
  - used to be a horrible cowboy hacker sort of thing
  - if following good practices is now a preferred method, especially for Software-as-a-Service
- it is still critical to plan what features will be released by when over a convenient time horizon (e.g., 6 months, or quarterly)
  - when code gets pushed to production is a detail
  - how customers are presented with the new features is a detail
  - all the “release planning” principles used for big bang releases still apply

6. feature specifications

- complicated features require them
  - need to make this determination
- needed to keep release plan on track
  - better estimates if know what we are doing in more detail
- enables a better end-user feature
- eliminates unanticipated integration problems
- best place to introduce reviews

- The agile approach is to develop smaller units of user-visible functionality, and have constant user input.
  - somewhat suspect

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

7. architectural control

- must maintain a clean architecture even in the face of
  - many coders working on the code
  - frequent feature additions
    - that the software was not designed for initially
  - frequent defect corrections
    - by inexperienced coders who do not understand the architecture

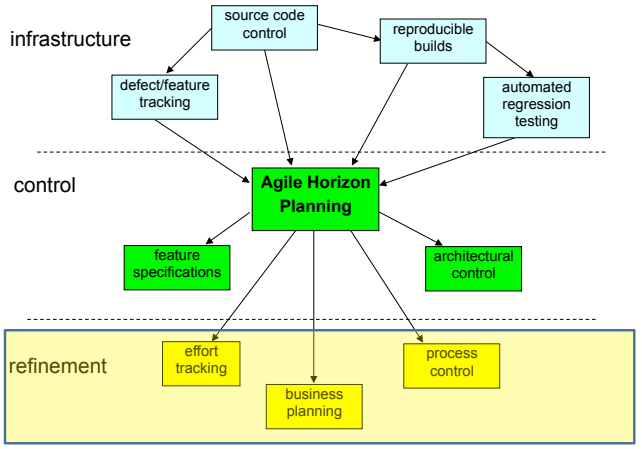
The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

architecture (2)

- architectural documentation
- review of designs and code for conformance
- chief architect (CSA)
- automated architectural checking tools

- agile approach is not to document the architecture. the code should be sufficiently well-designed that the architecture is clear.
  - somewhat suspect

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO



8. effort tracking

- need to know how much staff time is spent on
  - each new feature
  - correcting defects
  - other stuff
- can improve estimation accuracy
- can improve estimates of staff time available for next release
- can monitor effectiveness of initiatives to free up coder time for more coding

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

effort tracking (2)

- agile approach fixes the sprint length (e.g., 10 days), and looks at the number of "units" that were accomplished during that time. that establishes the number of "units" available for the next sprint of the same duration (velocity).
  - simple to implement
  - can't really say "why" and improve practices as a result
  - managers don't trust "units" (esp. non-R&D)

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

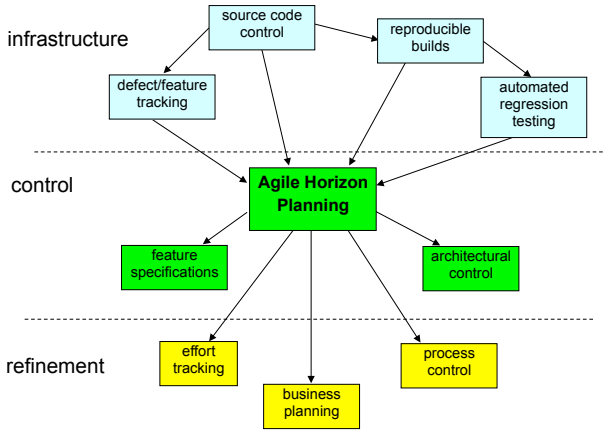
9. process control

- written process for the release cycle
- gets everybody on the same page
  - can train new staff
- enables systematic definition / collection of metrics
- can monitor process for compliance
- can consider changes to the process from a stable baseline

The Edward S. Rogers Sr. Department of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

10. business planning

- development occurs within a business context
- if not understood and managed, will sink the project more surely than technical shortcomings
- writing effective proposals
- integrating into the budget cycle.
- (may not have to cover this year)


infrastructure
source code control
reproducible builds
defect/feature tracking
automated regression testing
control
Agile Horizon Planning
feature specifications
architectural control
refinement
effort tracking
business planning
process control