Software Evolution & Technical Debt

December 12th 2012

Jurgen Vinju

Software Evolution

- Lehman: software goes bad eventually
- Standish: maintenance is the cost of software

Solutions?

Solutions

- Management
  - ISO/IEC 12207
  - ITIL
  - CMMI
- Mitigating the bad effects of evolution by managing people
- Ignores the cause?

Solutions

- Architecture
  - design for change
  - reversible design
- Platonic solution?
  - “if only we had...”

Solutions

- Automated testing
  - gives [sense of] security when modifying
- Who maintains the (complete?) tests?

Solutions

- Software renovation
  - Reverse engineering
  - Re-engineering
- Migrating old to new
- Tools mitigate investment cost
- Like starting from scratch, but with a burden?

Solutions

- Quality assessment & monitoring
  - Metrics, aggregation
  - Interpretation
- Preventing evolution issues
  - go/no-go
  - early focus on internal quality
- Assessing quality by measuring quantity?

Solutions

- Refactoring
  - continuous improvement
- Preventing degradation
- Waste of time? What to refactor to?

Solutions

- Design Patterns
  - Reserved names for [OO] solutions
    - Communicate to the reader
    - With documented internal quality attributes
- Do design patterns solve any problem?

Solutions

- Mining Software Repositories
  - predict where the bugs are
  - spot trends and sudden developments
  - observe what causes the problems
- So, now we know for sure that we have a problem?

Flipping the perspective

Flipping the perspective

- Previously:
  - Software maintenance should be done
  - Software evolution is problematic
  - Architecture, reverse engineering, metrics, … all fight against the problem
  - There is still a problem

Flipping the perspective

- Previously:
  - Software maintenance should be done
  - Software evolution is problematic
  - Architecture, reverse engineering, metrics, … all fight against the problem
  - There is still a problem
- Like:
  - We need to get to work
  - There is always a traffic jam
  - Share a car, widen the road, change working hours, take a train, ...
  - There is still a traffic jam

Flipping the perspective

- If we would work [closer to] where we live...
- We wouldn't need a car to go to work
- And, traffic jams would be a non-issue

- We all decided to live far from our jobs, and now we pay for this every day (except in weekends) and forever. It's our "forensic debt".

Technical Debt

- Coined by Ward Cunningham
- Metaphor of the decade (IMNSHO)
- Debt is what you take for granted because you have a higher priority elsewhere, but eventually you must pay it off, with interest.
- So, what would "technical debt" be?
  - what is the currency?
  - what is the interest?

Today's Technical Debt Hypothesis:

If we make internal software quality our primary goal, software evolution becomes a non-issue, and external quality will follow automatically

A Technical Debt Hypothesis:
If we make internal quality our primary goal, software evolution becomes a non-issue

Rhetorics:

First invest, then harvest.

No pain, no gain

Let's bring Mohammed to the mountain

`Assignment:`

O really?

Why?

How?

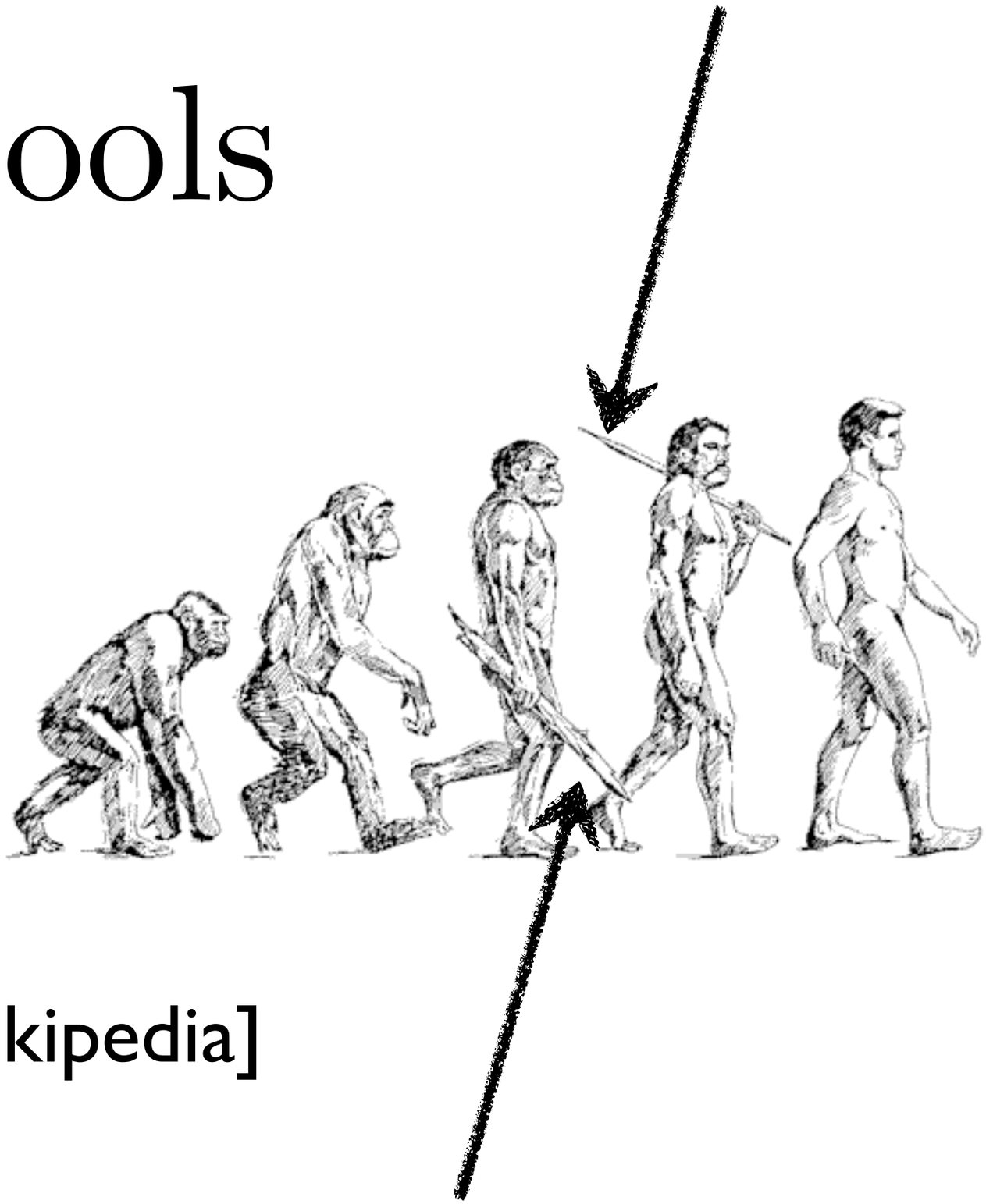
TDH Discussion

- O really? Why?
  - Tell us why this is not true
  - Tell us why this is true
  - What about “no silver bullet”? (Fred Brooks)
- How?
  - Tell us how this could be achieved?
  - What are critical success factors?

Real Solutions

- Write source code for readability
  - Code review
  - KISS - Keep it simple [&] stupid
- Monitor internal quality
  - Invest
  - Prioritize
  - Teach
- Tools: get help from the computer
  - search
  - summarize
  - refactor
  - test
  - generate

Tools



[wikipedia]

Rascal

- Manage source code diversity
  - languages, frameworks, libraries
  - reuse
- Manage tool multi-disciplinary nature
  - parse, model, analyze, generate, ...
  - integrate
- Build new tools to
  - search, summarize, test, generate, refactor, ...
  - and prevent or pay off technical debt

Your todos

- Transform argumentation scheme into text
  - revise motivation
  - revise argumentation
  - add conclusion
  - submit between dec 21 and dec 31 (2012)
- Finish series 2
  - demonstrate now or at least before dec 21 2012

Thanks!