Feedback Loops

in Practice

Marcus Denker

Talk held at ESUG2017
Two talks
(do not expect too much)
ESUG14
Scaffolding
Perfection
Feedback
Smalltalk can be feedback loop
Smalltalk should be feedback loop
TODAY
What does that mean in reality?
Examples for Pharo
What we do / what are the challenges
Goal: Getting feedback and ideas!
Trivial Changes
Every improvement has an effect
The only valid measurement of code quality: WTFs/minute.
A small change fed back will have huge payout
A tiny linear change now would be a **huge** change some iterations ago.
Trivial Change

- Issue tracker
  - Make it easy to contribute
  - Do not ignore contributions
Issue Tracker

- Fun: It was once thought as not needed
- Record issues people have
- Record contributions, too!
- Open: 651 Closed: 17459 (!!)
Challenges

• Work needed to keep clean
  • Duplicates, already fixed, non-actionable…
• Reviewing / Getting in good state
• Actually fixing reported bugs
Solutions

• Automatic close after 1 year inactivity
  • Has to be automatic else people get upset
• Some people look every to keep things in check
• Regular Sprints: every last Friday the month
  • More fun to do together then alone!
Make it easy

• It should be very easy to contribute

• We are not there yet!
Large(r) Change
Scaffolding 2014
Todays system is scaffolding for tomorrow
Elephant in the Room
Backward Compatibility
Example: Bloc/Bric
You can not stay 100% compatible to Morphic and do something better
Morphic is scaffolding to develop the next step
Nomadic Solution

- Do not build infrastructure
- Use resources until depleted
- Move on
Backward Compatibility

• Especially problematic for portable projects

• Why improve the Platform if projects can only use a 100% backward compatible subset?

• Is that a good situation?
Backward Compatibility

• We need better tools and structures to support evolution of client code

• Some experiments: rewriting deprecations (fun!)
Accept Imperfection
DONE IS BETTER THAN PERFECT
True for both small and big small
Good enough to integrate

- Deciding to integrate is very very hard
- You do not want to reject everything
- But accepting blindly is wrong, too

- Lots of work!
Involve the community

• Make it easy to review and test

• Delegate reviewing to subsystem maintainers

• Accept that nothing is perfect and mistakes can happen.
Accept Chaos

- You can not control everything. There is not enough time in the day.

- Things can get to be a bit chaotic at times

- Yet better than limiting activity to what is controllable
Feedback + Chaos

• Many examples where systems got into loop based exponential growth are examples lack of control:

• The web vs. online services

• Many examples at companies

• e.g Unics vs Multics, even X86 (often examples of perfect vs. DONE).
Release != Perfect

• Until Pharo 6: Lots of critic from outsiders about releasing something not perfect.

• But: Releases are done every year, not when everything is perfect.
Learned helplessness

- Smalltalk is open, can be changed
- Clients are programmers
- People do change tools/environment
- But “Smalltalk, the system” did not learn
Structure for Feedback
Structure for Feedback

- Example: GT Inspector
  - Extending the inspector is easy
  - There are lots of examples
  - It can be done in a modular way
Structure for Feedback

• Future Example: Sista
  • Implement Optimizer of the VM in the Image
    • Makes it easier (hopefully) for Smalltalkers to contribute
... round 1,000 times the global production of rice in 2010 (464,000,000 metric tons)
If it really works...
... we have a problem
Growth: everything gets “more”
Challenge: Growth

• More Boring tasks

• More complex tasks
  • Require full time, long term attention
Solution for Scaling

• Technical
  • e.g. Git for reviews and submissions, more people can get involved.

• Community Structure
  • Example: Consortium

• Even better solutions can be invented!
Input needed!

- Now
- At the Conference
- Later by Mail or in Discord
Twitter: @marcusdenkenker
http://marcusdenkenker.de