eXtreme Programming, Open Source, and Geographically Dispersed teams

Michael Richardson
<mcr@xelerance.com>
Xelerance Corporation
http://www.xelerance.com/
me:
http://www.sandelman.ca/mcr/

talk: http://www.xelerance.com/talks/bsdcan2004/

Overview of talk

- Introduction to speaker
- Introduction to eXtreme Programming
- The victim project
- Challenges of XP and Open Source
- Challenges of XP and telecommutting
- Tools and experiences
- Conclusions

Introduction to speaker

- Michael Richardson <mcr@xelerance.com>
- Long time Unix, BSD, Linux user.
- I write network protocol stuff: IPsec, radius, telnet, ssh, etc.
- history of security stuff: milkyway.com (firewalls), solidum.com (L2-L7 policy-based classification), SSH (IPsec-Express), other stuff.
- VP R&D at Xelerance Corporation
- Xelerance.com is providing 3rd level defect support for Openswan.
- AKA Sandelman Software Works, also tcpdump.org maintainer.
- This talk at http://www.xelerance.com/talks/bsdcan2004/

Introduction to eXtreme Programming

- no revolution here – just evolution
  - turns out I've been doing 70% of it all along
  - I'll bet you do that much already
  - nice manager friendly names for things we already knew
- some books by X
- websites: http://www.extremeprogramming.org/

10 key elements of eXtreme Programming

- user stories
- iterations
- unit tests
- refactoring
- pair programming
- acceptance tests
- daily stand up meeting
- integrate often
- release often
- fix XP when it breaks

User stories

- the user/customer defines what is important.
- they define this via a “story”, or scenario of some kind.
- e.g. “I want to be able to type

  `pkg_info –vuln-check apache2”`

Iterative Development

- features are added one at a time
- user/customer is allowed to provide feedback
- entire system is always useable in some fashion at end of iteration
- short iterations. 2 weeks to 2 months.

Unit tests

- all code has a unit test
- unit test confirms that one function/program/subsystem works
- provides for typical inputs/outputs
- is easy to run
- used sanity check
- added when bugs are found

Refactoring

- code is rewritten/broken up as needed
- test cases confirm that code is correct
-

Pair Programming

- two brains, one keyboard/screen
- allows for strategic/tactical thinking
- spreads out the knowledge
- makes new experts
- natural mentoring
- teaches old dogs new tricks

Acceptance Tests

- a test case which the user/customer uses to determine if feature is correct
- the acceptance test tells the programmer if they are finished yet
- explains the requirements in a brief fashion

Daily stand up meetings

- communicates current status quickly
- replaces long status meetings
- provides for quick response when people are stuck
-

Integrate Often

- merge all code into HEAD regularly
- refactored code has unit tests – they pass first.
- systems have acceptance tests

Release Often

- the customer never has to wait long
- fast response to customer need
- no custom/one-off releases for one-customer
- increased enthusiasm, feeling of success

fix XP when it breaks

- none of this is written in stone
- fix the process when it is wrong
- evolution – not revolution.

Victim Project

- Linux FreeS/WAN IPsec project
- hard to test stuff
- initially 3 in Ottawa, 3 in Toronto, 1 “manager” on Earth, 1 “funder” in San Francisco
  - US export/influence situation
- lots of security for privacy (STU-III “batphones” initially)
- many different networks
  - mine is 70% NetBSD, 30% Linux. All critical boxes are NetBSD. Similar mixes elsewhere.

Why talk about this here?

- Linux experience at BSDcan?
- FreeS/WAN project was closer to *BSD than Linux kernel in philosophy
  - project leads trying to bring BSD quality to Linux networking
  - structure of *BSD is simpler than Linux – one project, not hundreds of distros, vs kernel.
- (Net)BSD project(s) have frequent issues/debates with quality/scope/schedule.

Testing infrastructure

- I spent 2001/2002 was spent working on testing infrastructure. (In advance of switch to XP)
- host based simulation via User-Mode-Linux rather than 5 box system
- tests are repeatable, self-contained and portable
- strong preference for “dog-food” - we transitioned in 2001/2002 to running our code everywhere – including CVS access.

Transition to XP

- started monthly telephone conferences via multi-line POTS+batphone
- switched to H.323 (“GnomeMeeting”) in Winter 2002. Over IPsec!
- switched to SIP in June 2003. Over IPsec.
- team-only IRC channel (Over IPsec)
- public IRC channel on irc.openprojects.net -> ir-c.freenode.net.

Transition to XP (2)

- bought books in March 2002
  - some confusion as to whether we were “doing XP” yet. Transition to XP can be hard.
  - full blown by August 2002.
- all current tasks were broken up into stories.
- Developers were asked to provide estimates of effort required.
- “customer” was asked to rank things
- we wrote an iteration schedule, and stuck to it.

Velocity Measure

- velocity is a measure of “useful work time” to “real time”
- takes in account interruptions, email, being sick, re-installing Windows (!), etc.
- this is measured empircally

Challenges of XP and Open Source

- who is the customer?
- need them for acceptance tests/sign-off
- need them for ranking of work
- who is the responsible for testing?
- who is the “project manager”? (do we need one?) (do they have enough authority?)

Who is the customer

- we are the customers.
  - compiler people are customers of kernel peo-ple
  - kernel people are customers of compiler peo-ple
  - all of them work under X-windows
  - libc people are customers of kernel
- you make up the relations

Who is the customer

- actually points to a lack – we have developers and we organize them!
- but we don't have organizations of customers!
- Linux has this – Distro companies provide for the customers. Where are the equivalents for *BSD?
- there are some, but not well known.

Who is the customer

- FreeS/WAN project had political problems – the guy with the money felt he was not allowed to provide technical input!
- had to improvise a bit.

Challenges of XP and telecommuting

- stand-up meetings
- pair programming
- integrate often
- iterations
  - vastly different velocity measures

Stand-up Meetings

- despite being in the same timezone, we were almost al-ways in different phases.
  - worse when real time zones are involved as well
  - also worse when developers are not paid!
- agreed to “be around” at 13:30 every day.
  - At least in IRC.

Stand Up Meetings

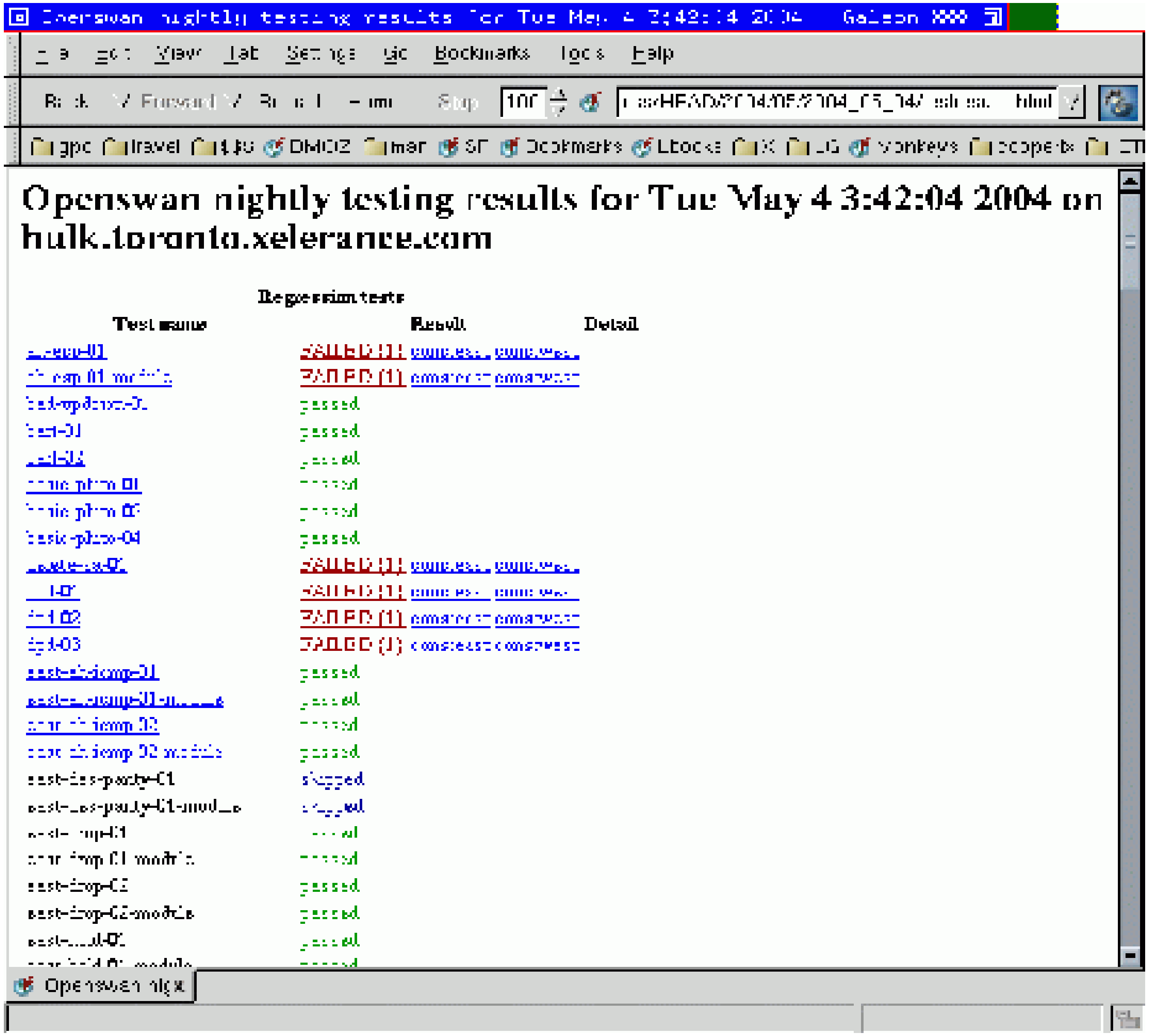
- VoIP when we need it.
- iteration meeting is well announced
  - often 3-4 hours long
  - once every three weeks (our iteration cycle)
  - near end of current iteration
- 45 minutes allocated to “technical” issues with meeting.
  - eat your own dog food
  - many rekey and DNS related bugs in IPsec found by using it ourselves

testing – when to complain

- testing system issues daily email to all
- lists tests that succeed, those that fail, and permits deeper analysis
- often culprit would know they were responsible.
- but, sometimes didn't have time to fix.
- DERIVED RULE:

all test cases pass at end of iteration

testing – example output


Openswan nightly testing results for Tue May 4 3:42:04 2004 - Galeon
File Edit View Tab Settings Go Bookmarks Tools Help
Back Forward Reload Home Stop
100
gpc Travel DMOZ SF Bookmarks Monkeys
Openswan nig

Openswan nightly testing results for Tue May 4 3:42:04 2004 on hulk.toronto.xelerance.com

Regression tests

Test name	Result	Detail	
[illegible]-01	FAILED (1)	[illegible]	[illegible]
[illegible]-esp-01-[illegible]	FAILED (1)	[illegible]	[illegible]
[illegible]-01	passed		
[illegible]-01	passed		
[illegible]-02	passed		
basic-pluto-01	passed		
basic-pluto-02	passed		
basic-pluto-04	passed		
[illegible]-02	FAILED (1)	[illegible]	[illegible]
[illegible]-02	FAILED (1)	[illegible]	[illegible]
[illegible]-02	FAILED (1)	[illegible]	[illegible]
[illegible]-03	FAILED (1)	[illegible]	[illegible]
east-icmp-01	passed		
east-icmp-01-[illegible]	passed		
[illegible]-icmp-02	passed		
[illegible]-icmp-02-[illegible]	passed		
east-[illegible]-party-01	skipped		
east-[illegible]-party-01-module	skipped		
east-[illegible]-01	passed		
[illegible]-01-module	passed		
east-[illegible]-02	passed		
east-[illegible]-02-module	passed		
east-[illegible]-02	passed		
[illegible]-02-module	passed		

xelerance

how to do pair programming?

- screen!
- has multiuser mode
- two people SSH to same box, start screen.
- get on VoIP or POTS to talk
- get on IRC in case VoIP fails

Pair Programming (2)

- origin of Pair Programming from SmallTalk
- no compile cycle
- what to do while it compiles?
- answer: being in seperate places with two key-boards, two screens, means one can multitask.
- not clear if this is a good thing!

testing – getting everyone on the same page

- acceptance tests.
- we integrated too often!
- branches are hard. Maybe SubVersion will help.
-

release often

- we got better at this
- but we experienced “2.0 syndrome”, despite trying to avoid this.
- we did get better after .0

Conclusions

- eXtreme Programming can work
- open source projects can benefit from these methods
  - velocity is a problem
  - paid/unpaid is a problem
- geography is a challenge, but also an opportunity

Questions?

- me:
- http://www.sandelman.ca/mcr/
- talk: http://www.xelerance.com/talks/bsdcan2004/
-