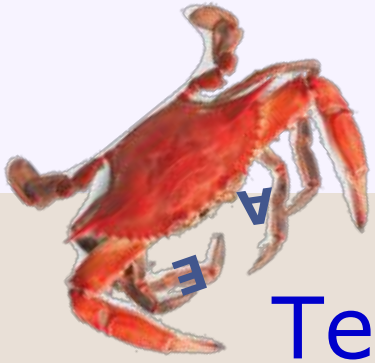
Annapolis

Carnegie Mellon

Designed-in Security:
Needs, successes, prospects

William L Scherlis
Professor and Director
Institute for Software Research
School of Computer Science
Carnegie Mellon University

May 2012



Technology Transition?

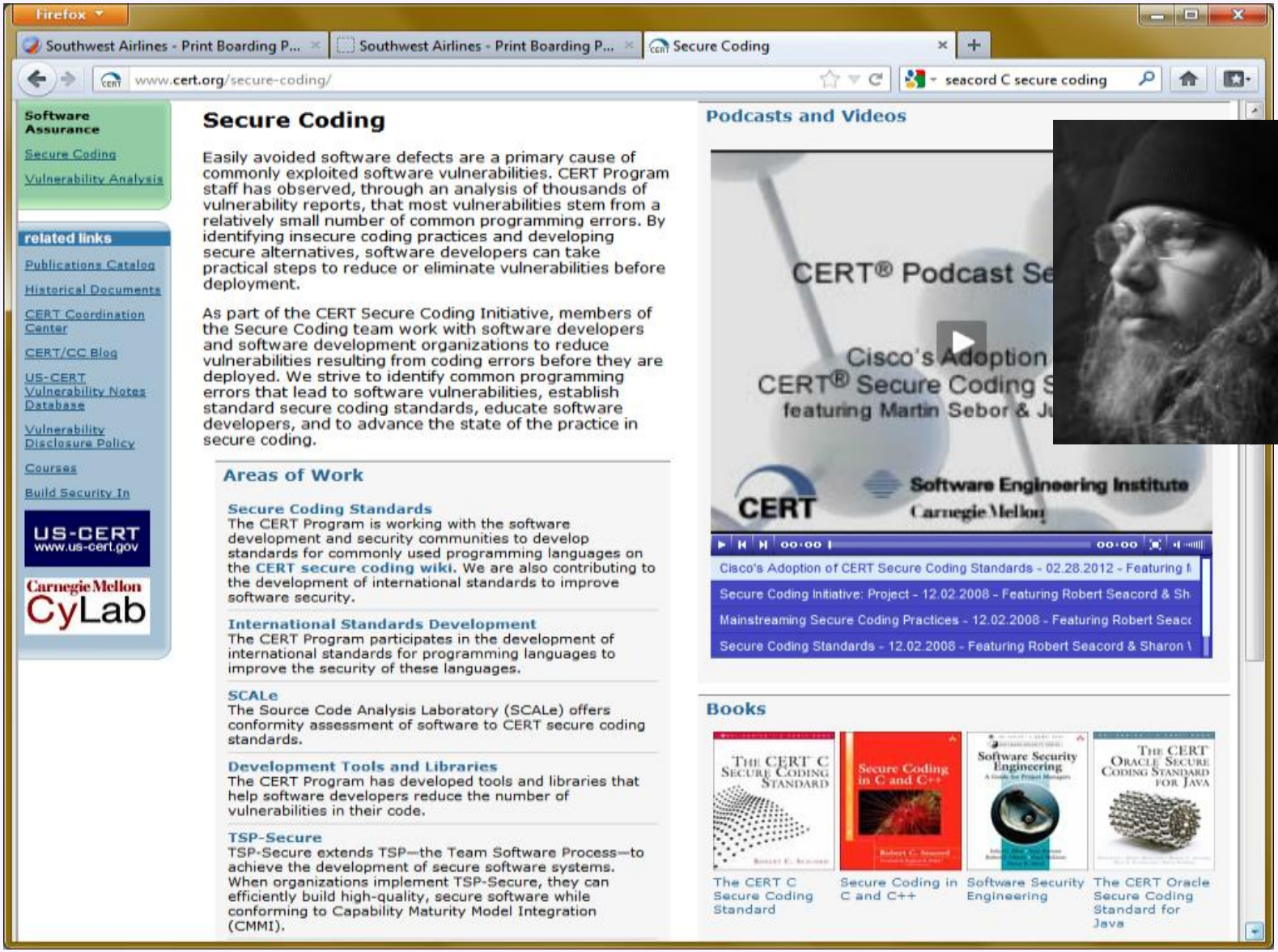
Transition-driven technical challenges

- Interplay of development and evidence production
  - A harmonized practice for development and evaluation teams
- Metrics
  - Towards ROI models for assurance-related investment
- Recertification
  - Necessary for SAAS and agile/IID
- Configurations and product families
  - Evidence of need: massive #ifdef combinatorics
- Component-based systems
  - Composition with a wide range of trust – attack surface is within
- Framework configurations
  - More than mobile

Patterns of transition success

Ex. 1: Microsoft

Ex. 2: Secure coding



Ex. 3: DSLs

- Cryptol
- *et al*

The following Cryptol code defines the core encryption routine for DES.

EXAMPLE 8-1 *DES Encryption*

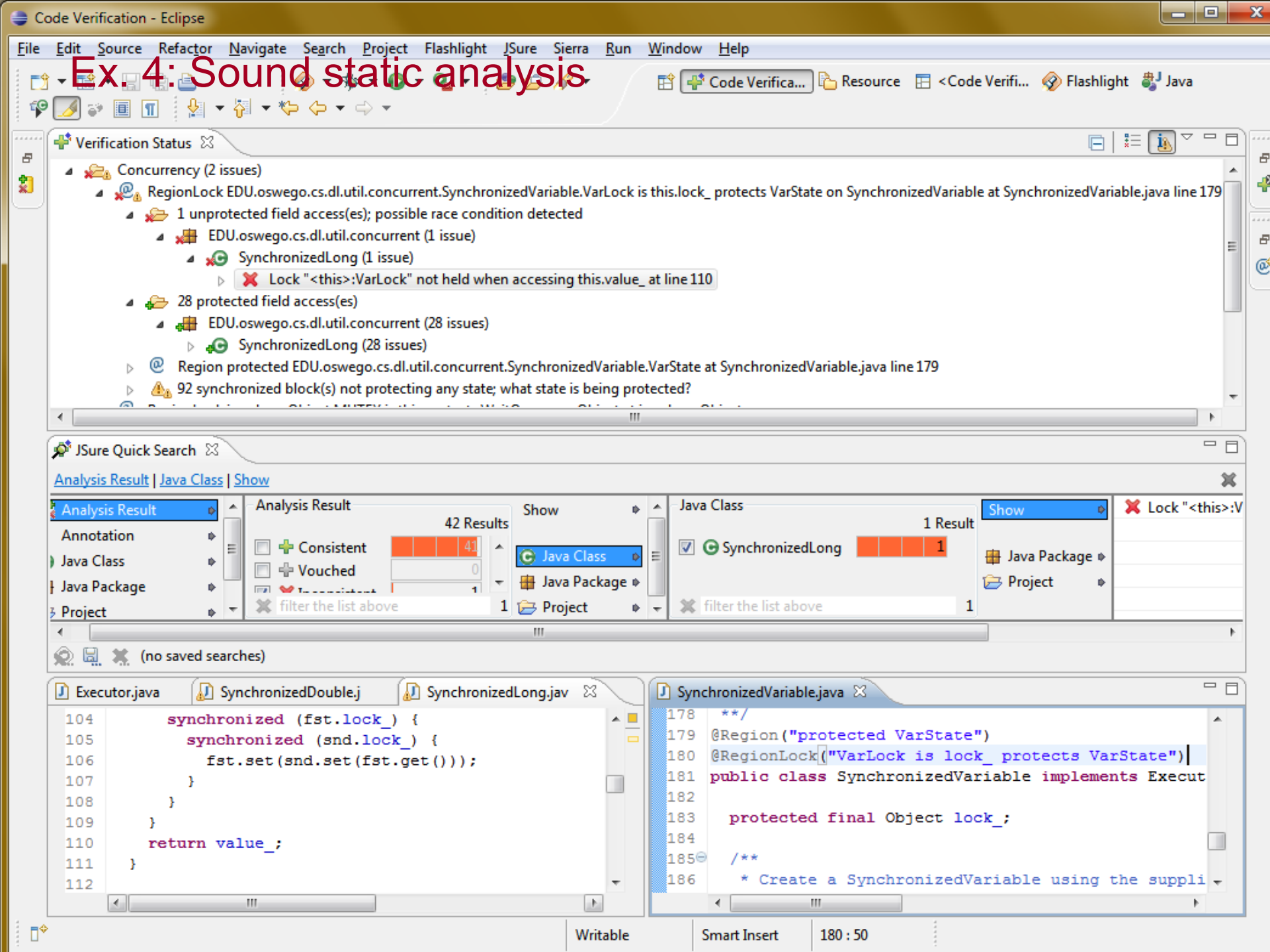
```
des : {a b} (a >= 7) => ([2**(a- 1)],[b][48]) -> [64];
des (pt, keys) = permute (FP, swap (split last))
  where { pt' = permute (IP, pt);
          iv = [| round (k, split lr)
                | k <- keys
                | lr <- [pt'] # iv
                |];
          last = iv @ (width keys - 1);
         };

round (k, [l r]) = r # (l ^ f (r, k));

f (r, k) = permute(PP, SBox(k ^ permute(EP, r)));

swap [a b] = b # a;

permute : {a b} (b >= 1) => ([a][b], [2**(b - 1)]) -> [a];
permute (p, m) = [| m @ (i - 1) || i <- p |];
```


Code Verification - Eclipse
File Edit Source Refactor Navigate Search Project Flashlight JSure Sierra Run Window Help
Ex. 4: Sound static analysis
Code Verifica... Resource <Code Verifi... Flashlight Java
Verification Status
Concurrency (2 issues)
RegionLock EDU.oswego.cs.dl.util.concurrent.SynchronizedVariable.VarLock is this.lock_ protects VarState on SynchronizedVariable at SynchronizedVariable.java line 179
1 unprotected field access(es); possible race condition detected
EDU.oswego.cs.dl.util.concurrent (1 issue)
SynchronizedLong (1 issue)
Lock "<this>:VarLock" not held when accessing this.value_ at line 110
28 protected field access(es)
EDU.oswego.cs.dl.util.concurrent (28 issues)
SynchronizedLong (28 issues)
Region protected EDU.oswego.cs.dl.util.concurrent.SynchronizedVariable.VarState at SynchronizedVariable.java line 179
92 synchronized block(s) not protecting any state; what state is being protected?
JSure Quick Search
Analysis Result | Java Class | Show
Analysis Result
Annotation
Java Class
Java Package
Project
42 Results
Consistent
Vouched
filter the list above
Show
Java Class
Java Package
Project
1 Result
SynchronizedLong
Java Package
Project
(no saved searches)
Executor.java
SynchronizedDouble.j
SynchronizedLong.jav
104 synchronized (fst.lock_) {
105 synchronized (snd.lock_) {
106 fst.set(snd.set(fst.get()));
107 }
108 }
109 }
110 return value_;
111 }
112
SynchronizedVariable.java
178 **/
179 @Region("protected VarState")
180 @RegionLock("VarLock is lock_ protects VarState")
181 public class SynchronizedVariable implements Execut
182
183 protected final Object lock_;
184
185 /**
186 * Create a SynchronizedVariable using the suppli
Writable
Smart Insert
180 : 50

Patterns

- Structure
  - Support composability
  - Use cutpoints and specifications
- Models and analysis
  - Acknowledge attribute specificity
  - Employ diverse analytics: MC, SwA, TP, verification, etc.
- Tooling and practice
  - Integrate with widely used IDEs and team tools
  - Provide ongoing traceability support
  - Guide developers to errors; guide them to the fixes
  - Support proof management and truth maintenance (***examples***)
  - Deliver useful metrics of progress
- Adoptability and business case
  - Hide the cool math – focus on usability for developers/evaluators
  - Offer heuristic assist
  - Deliver early and ongoing gratification for verification effort
  - Manifest ROI models for each of developers, teams, enterprise

- Scale and complexity
- Value on simplicity/exposure
- Incrementality wrt change
- Incrementality wrt assurance

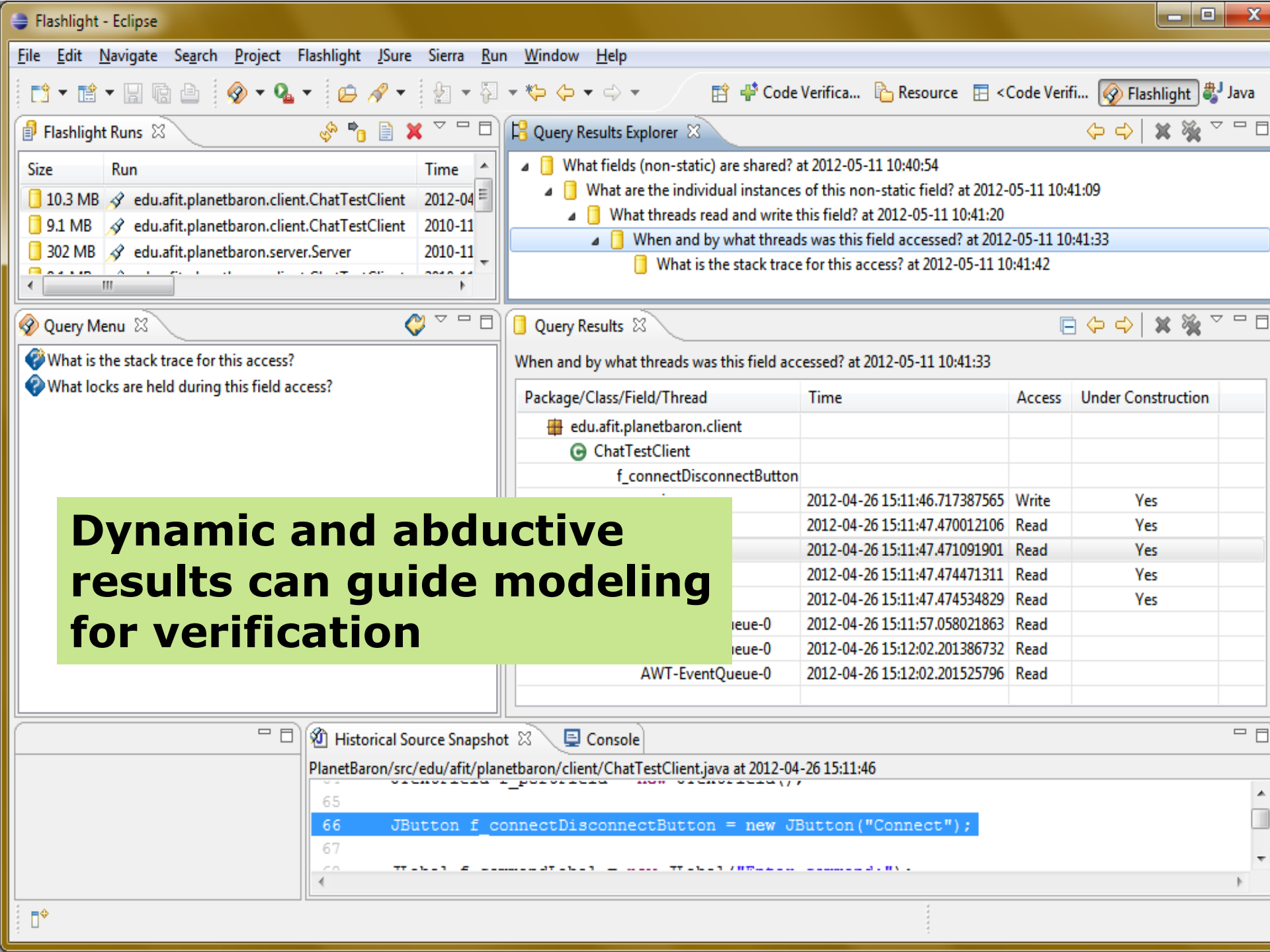
Patterns

- Structure
  - Support composability
  - Use cutpoints and specifications

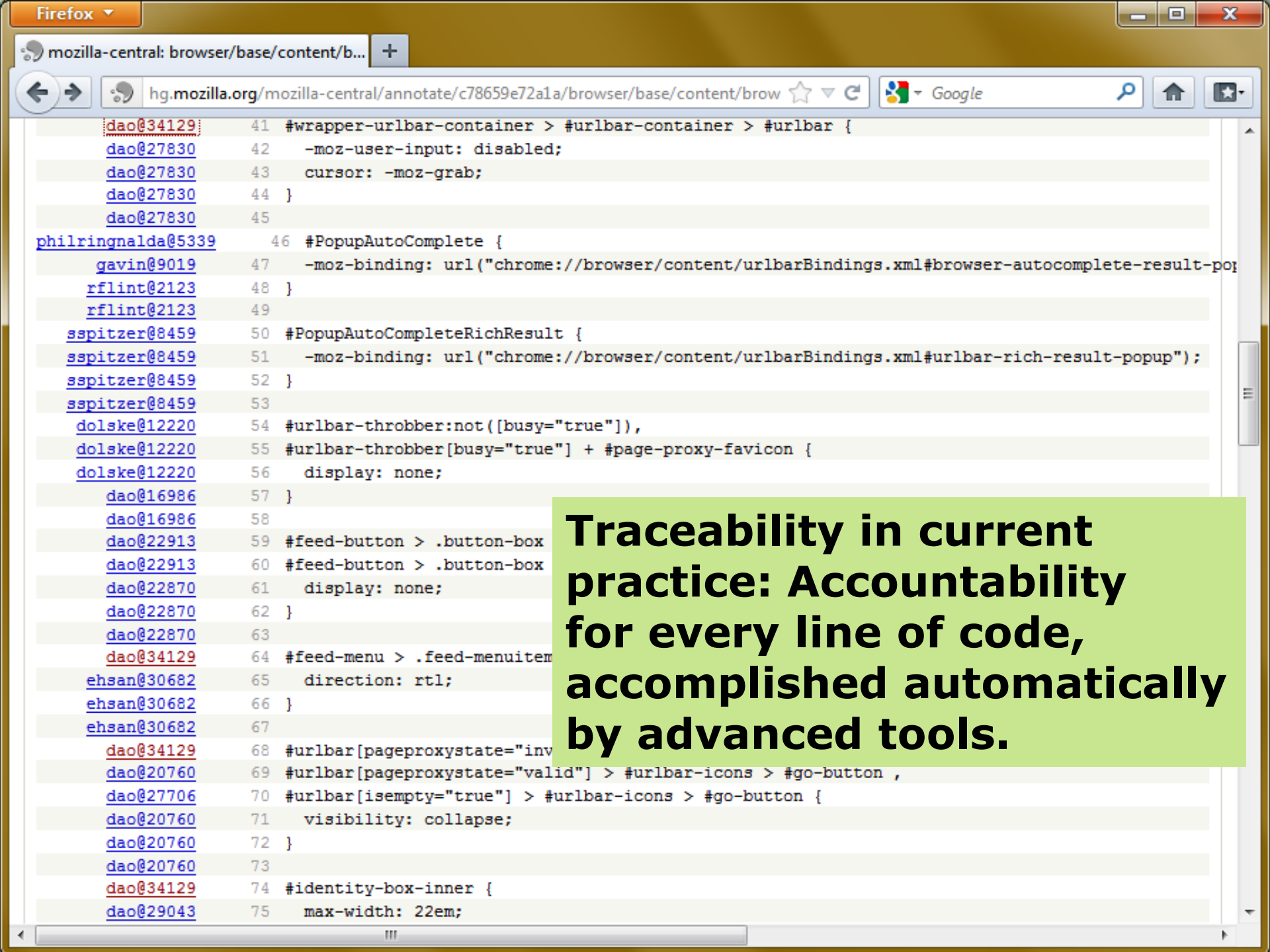
- Scale and complexity
- Value on simplicity/exposure
- Incrementality wrt change
- Incrementality wrt assurance

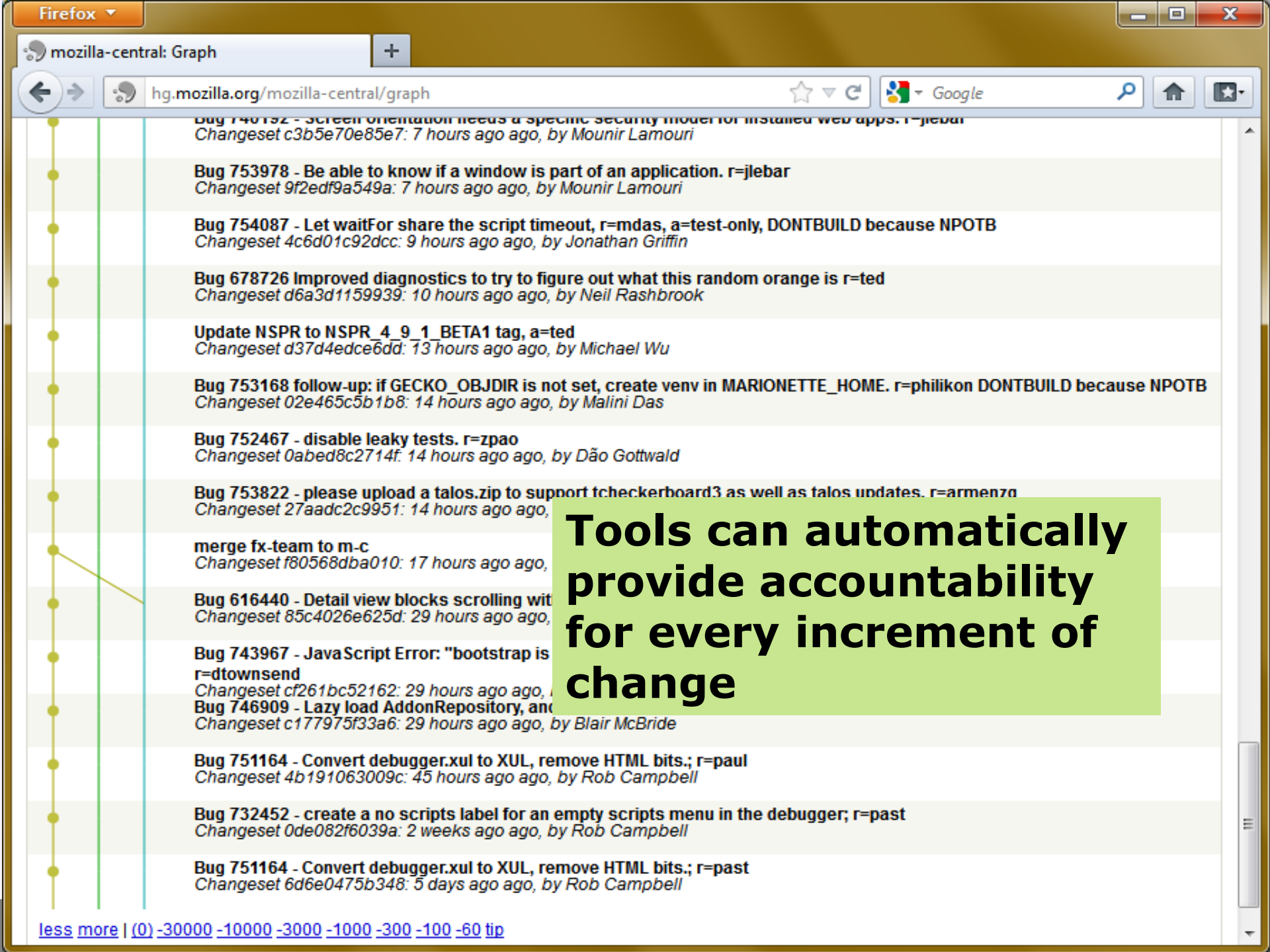
Interplay of development and assurance
- Code, models, proof structures
- Process and practice in development

Influence of success on devt infrastructure
- Types, storage, encap, parallelism, …

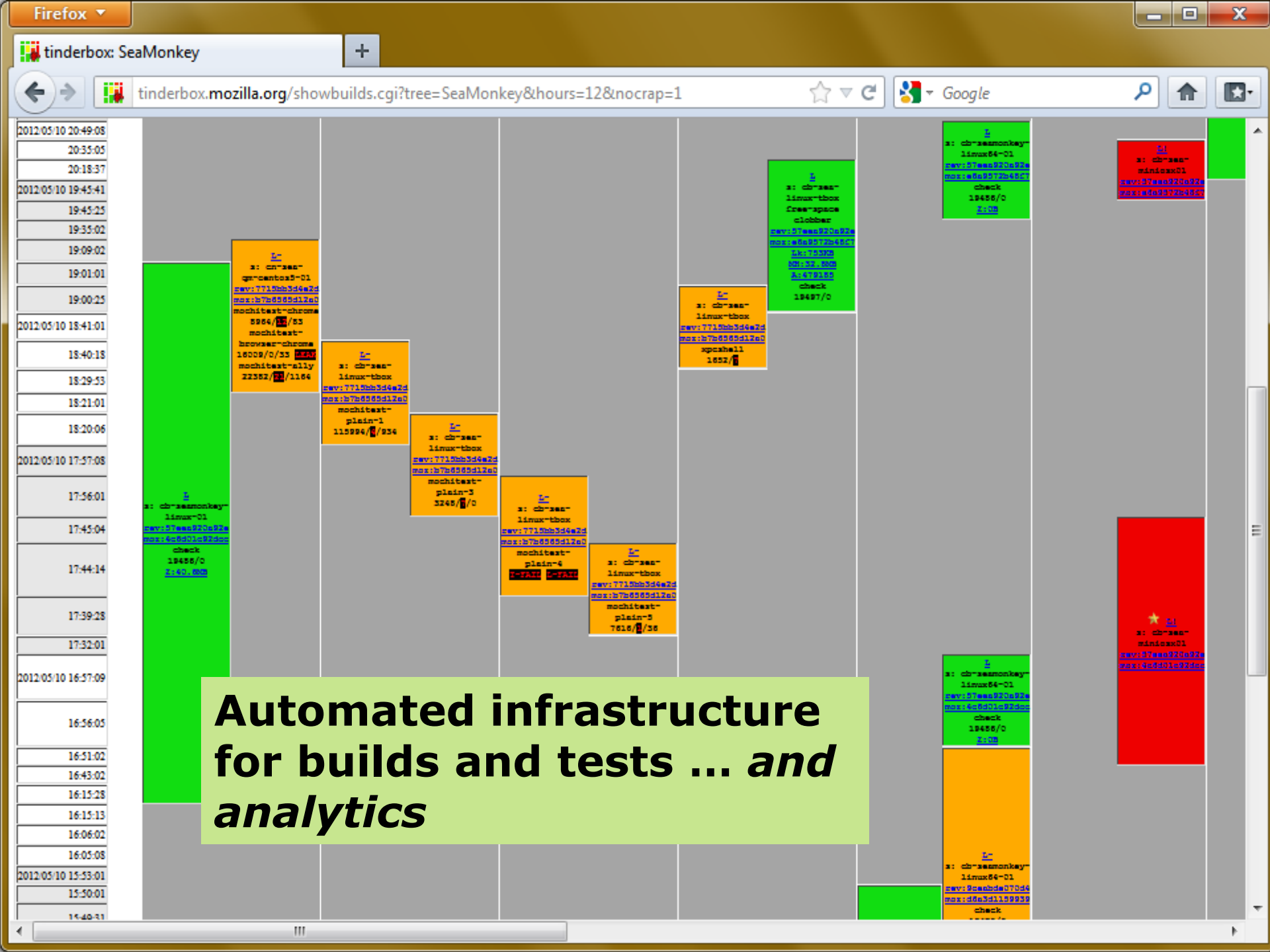


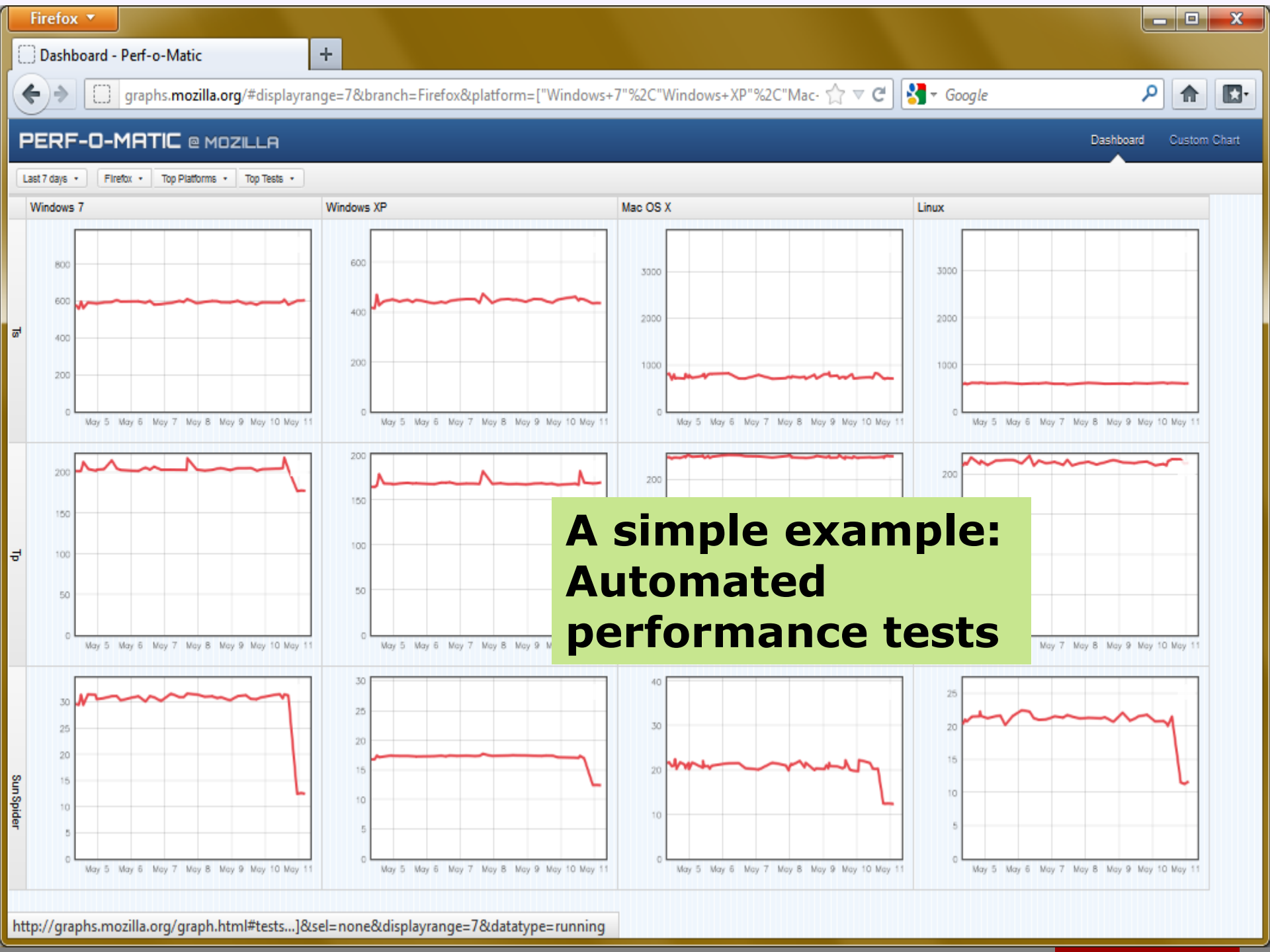
Dynamic and abductive results can guide modeling for verification


Firefox
mozilla-central: browser/base/content/b...
hg.mozilla.org/mozilla-central/annotate/c78659e72a1a/browser/base/content/brow
Google
dao@34129 41 #wrapper-urlbar-container > #urlbar-container > #urlbar {
dao@27830 42 -moz-user-input: disabled;
dao@27830 43 cursor: -moz-grab;
dao@27830 44 }
dao@27830 45
philringnalda@5339 46 #PopupAutoComplete {
gavin@9019 47 -moz-binding: url("chrome://browser/content/urlbarBindings.xml#browser-autocomplete-result-pop
rflint@2123 48 }
rflint@2123 49
sspitzer@8459 50 #PopupAutoCompleteRichResult {
sspitzer@8459 51 -moz-binding: url("chrome://browser/content/urlbarBindings.xml#urlbar-rich-result-popup");
sspitzer@8459 52 }
sspitzer@8459 53
dolske@12220 54 #urlbar-throbber:not([busy="true"]),
dolske@12220 55 #urlbar-throbber[busy="true"] + #page-proxy-favicon {
dolske@12220 56 display: none;
dao@16986 57 }
dao@16986 58
dao@22913 59 #feed-button > .button-box
dao@22913 60 #feed-button > .button-box
dao@22870 61 display: none;
dao@22870 62 }
dao@22870 63
dao@34129 64 #feed-menu > .feed-menuitem
ehsan@30682 65 direction: rtl;
ehsan@30682 66 }
ehsan@30682 67
dao@34129 68 #urlbar[pageproxystate="inv
dao@20760 69 #urlbar[pageproxystate="valid"] > #urlbar-icons > #go-button ,
dao@27706 70 #urlbar[isempty="true"] > #urlbar-icons > #go-button {
dao@20760 71 visibility: collapse;
dao@20760 72 }
dao@20760 73
dao@34129 74 #identity-box-inner {
dao@29043 75 max-width: 22em;
Traceability in current practice: Accountability for every line of code, accomplished automatically by advanced tools.

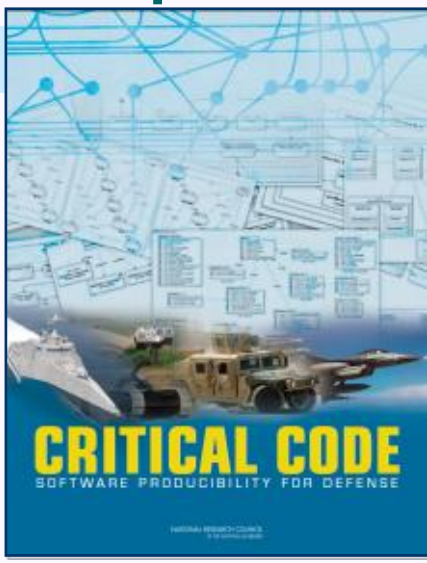


Tools can automatically provide accountability for every increment of change


Firefox
tinderbox: SeaMonkey
tinderbox.mozilla.org/showbuilds.cgi?tree=SeaMonkey&hours=12&nocrap=1
Google
2012/05/10 20:49:08
20:35:05
20:18:37
2012/05/10 19:45:41
19:45:25
19:35:02
19:09:02
19:01:01
19:00:25
2012/05/10 18:41:01
18:40:18
18:29:53
18:21:01
18:20:06
2012/05/10 17:57:08
17:56:01
17:45:04
17:44:14
17:39:28
17:32:01
2012/05/10 16:57:09
16:56:05
16:51:02
16:43:02
16:15:28
16:15:13
16:06:02
16:05:08
2012/05/10 15:53:01
15:50:01
Automated infrastructure for builds and tests ... and analytics


Firefox
Dashboard - Perf-o-Matic
graphs.mozilla.org/#displayrange=7&branch=Firefox&platform=["Windows+7"%2C"Windows+XP"%2C"Mac-
Google
PERF-O-MATIC @ MOZILLA
Dashboard
Custom Chart
Last 7 days
Firefox
Top Platforms
Top Tests
Windows 7
Windows XP
Mac OS X
Linux
Ts
Tp
SunSpider
May 5 May 6 May 7 May 8 May 9 May 10 May 11
A simple example: Automated performance tests
http://graphs.mozilla.org/graph.html#tests...]&sel=none&displayrange=7&datatype=running

Relevant material from the NRC *Critical Code* report



1. **Practice** – *Enhance mission capability, agility, assurance, linking*
   - **Enable incremental iterative development at arm's length**
     - Process and measurement – rethinking the practice
   - **Enable architecture leadership, interlinking, flexibility**
     - Architecture – "architecture ≈ strategy"
   - **Enable mission assurance at scale, with rich supply chains**
     - Assurance and security – evidence-based and preventive
2. **Research** – *Promote game-changers*
   - Architecture modeling and architectural analysis
   - Validation, verification, and analysis of design and code
   - Process support and economic models for assurance
   - Requirements
   - Language, modeling, code, and tools
   - Cyber-physical systems
   - Human-system interaction
3. **Leadership** – *Never relinquish the innovation lead*
   - **Recognize the unboundedness of software**
   - **Stay ahead in assurance** (cf. DSB'07)
   - **Sustain innovation and ecosystem lead**

Challenge issues
- **Technology leadership focal point**
- **Smart customer: inside expertise**
- **Accelerate the pipeline**

Adopt a strategic approach to software assurance

- **Current technical approaches to software assurance are inadequate**.
  - *Assurance*
    - *A human judgment regarding reliability, safety, security, etc.*
  - Current technical approaches need to be augmented
    - Costs range from 30-50% for typical major projects
    - Testing and inspection techniques are inadequate for modern software devt

- **Assurance conclusions are difficult to draw.**
  - Not analogous to reliability models for physical systems
  - Cannot be achieved entirely through *post hoc* acceptance evaluation
    - Quality and security are built in, not “tested in”

“Foreign influence” on software – DSB 2007

- *Provenance is a poor surrogate for direct evaluation*
- *We need to be better at understanding our own code*

RECOMMENDATION: DoD should provide incentives to industry to produce higher quality code.

There will never be a single set of best practices, testing tools, methodologies, or development process that works for all vendors -- nor is that desirable, since

Promote the Use of Automated Tools in Product Development

Tools for the detection of vulnerabilities continue to improve and proliferate. Different vendors will choose to use different tools; no one tool is right for every company. The DoD may well have an interest in determining how good the tools are that vendors use, but DoD should not be in the position of dictating particular

Report of the
Defense Science Board Task Force
on
Mission Impact of Foreign Influence on DoD Software

September 2007

Office of the Under Secretary of Defense
For Acquisition, Technology, and Logistics
Washington, D.C. 20301-3140

Adopt a strategic approach to software assurance

- **DoD faces particular challenges to assurance**.
  1. The **arms-length relationship** between a contractor development team and government stakeholders
  2. Modern systems of all kinds draw on components from **diverse sources**
     - This implies that **supply-chain attacks** must be contemplated, along with attack surfaces within the software application
       - There will necessarily be differences in the levels of trust conferred on components.
       - There may also be opacity in the supply chain for vendor and sub components
     - **Evaluative and preventive approaches** can be integrated to enhance assurance in complex supply chains with diverse sourcing.
  3. **High consequences** due to roles in war-fighting and protection of human lives and national assets
  4. Failure to maintain a lead in the ability to prevent and evaluate confers **advantage to adversaries** (*DSB2007, paraphrased*)
- Finding from DSB2007

  It is an essential requirement that the United States maintain advanced capability for "test and evaluation" of IT products. Reputation-based or trust-based credentialing of software ("provenance") needs to be augmented by direct, artifact-focused means to support acceptance evaluation.

Conclusions – patterns for progress in the mainstream

- Languages are improving
  - **L + M + A → L'**
- Enrich API focus
  - Enrich models at APIs
- Enhance architecture focus
  - Structure for trust localization/isolation
- Push further development of abstractions and modeling formalisms
  - With CPS and beyond CPS
- Tools are essential to support modeling and analysis
  - Already true for development: individuals, teams, enterprise
  - Proof management is a first-class activity
  - Heuristic assist (abductive, correlative, etc) pays off
  - Replace missionary work with metrics
- Adapt evaluation practices and policies
  - Support incrementality and continuous evolution – constant ROI
  - Don't require full-scope verification – tests and inspection results
  - Incent the interplay of development, evidence-building, assurance
  - Integrate with SDL-like processes