



HCSS

DIS

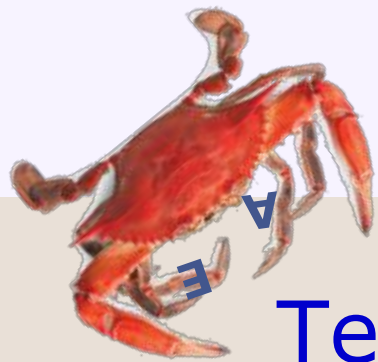


Annapolis

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

Software Assurance
[Secure Coding](#)
[Vulnerability Analysis](#)

related links
[Publications Catalog](#)
[Historical Documents](#)
[CERT Coordination Center](#)
[CERT/CC Blog](#)
[US-CERT Vulnerability Notes Database](#)
[Vulnerability Disclosure Policy](#)
[Courses](#)
[Build Security In](#)

US-CERT
www.us-cert.gov

Carnegie Mellon CyLab

Secure Coding

Easily avoided software defects are a primary cause of commonly exploited software vulnerabilities. CERT Program staff has observed, through an analysis of thousands of vulnerability reports, that most vulnerabilities stem from a relatively small number of common programming errors. By identifying insecure coding practices and developing secure alternatives, software developers can take practical steps to reduce or eliminate vulnerabilities before deployment.

As part of the CERT Secure Coding Initiative, members of the Secure Coding team work with software developers and software development organizations to reduce vulnerabilities resulting from coding errors before they are deployed. We strive to identify common programming errors that lead to software vulnerabilities, establish standard secure coding standards, educate software developers, and to advance the state of the practice in secure coding.

Areas of Work

Secure Coding Standards

The CERT Program is working with the software development and security communities to develop standards for commonly used programming languages on the [CERT secure coding wiki](#). We are also contributing to the development of international standards to improve software security.

International Standards Development

The CERT Program participates in the development of international standards for programming languages to improve the security of these languages.

SCALE

The Source Code Analysis Laboratory (SCALE) offers conformity assessment of software to CERT secure coding standards.

Development Tools and Libraries

The CERT Program has developed tools and libraries that help software developers reduce the number of vulnerabilities in their code.

TSP-Secure

TSP-Secure extends TSP—the Team Software Process—to achieve the development of secure software systems. When organizations implement TSP-Secure, they can efficiently build high-quality, secure software while conforming to Capability Maturity Model Integration (CMMI).

Podcasts and Videos

CERT® Podcast Series
Cisco's Adoption of CERT® Secure Coding Standards
featuring Martin Sebor & J...

CERT **Software Engineering Institute**
Carnegie Mellon

▶ 00:00 | 00:00

Cisco's Adoption of CERT Secure Coding Standards - 02.28.2012 - Featuring Martin Sebor & Sharon Lee
Secure Coding Initiative: Project - 12.02.2008 - Featuring Robert Seacord & Sharon Lee
Mainstreaming Secure Coding Practices - 12.02.2008 - Featuring Robert Seacord & Sharon Lee
Secure Coding Standards - 12.02.2008 - Featuring Robert Seacord & Sharon Lee

Books

THE CERT C SECURE CODING STANDARD
Robert C. Seacord

Secure Coding in C and C++
Robert C. Seacord

Software Security Engineering
A Guide for Project Managers
John Viega, Gary McGraw

THE CERT ORACLE SECURE CODING STANDARD FOR JAVA
Robert C. Seacord, Sharon Lee

The CERT C Secure Coding Standard | Secure Coding in C and C++ | Software Security Engineering | The CERT Oracle Secure Coding Standard for Java

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 ||;
```

Ex. 4: Sound static analysis

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	42 Results	Show	Java Class	1 Result	Show
Annotation	<input type="checkbox"/> Consistent 41	<input checked="" type="checkbox"/> Java Class	<input checked="" type="checkbox"/> SynchronizedLong	<input type="checkbox"/> Java Package	<input checked="" type="checkbox"/> Lock "<this>:V
Java Class	<input type="checkbox"/> Vouched 0	<input type="checkbox"/> Java Package		<input type="checkbox"/> Project	
Java Package	<input type="checkbox"/> Inconsistent 1	<input type="checkbox"/> Project			
Project	<input checked="" type="checkbox"/> filter the list above 1				

(no saved searches)

```

104     synchronized (fst.lock_) {
105         synchronized (snd.lock_) {
106             fst.set (snd.set (fst.get ()));
107         }
108     }
109 }
110 return value_;
111 }
112
  
```

```

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
  
```


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

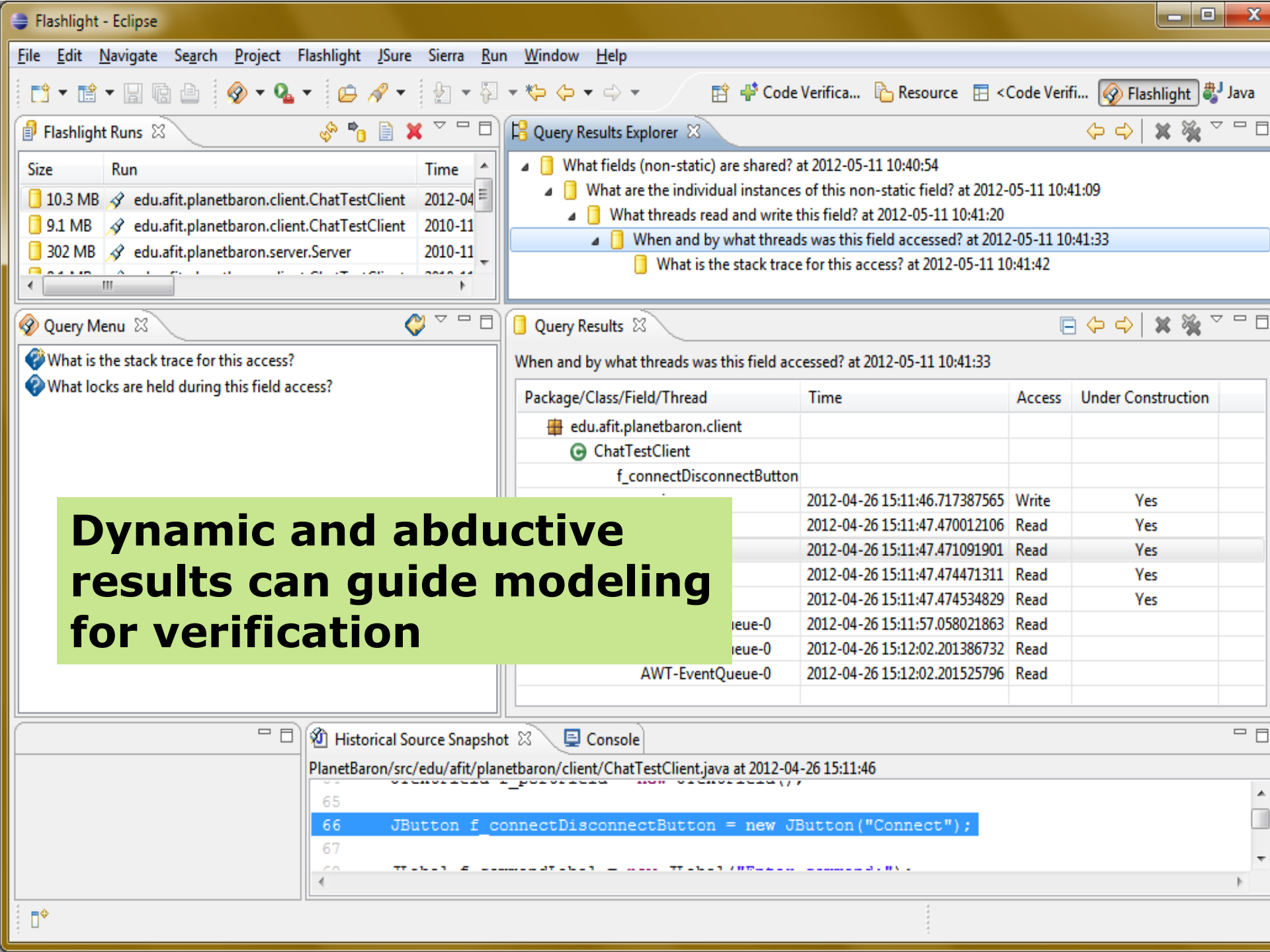
- Structure
 - Support composability
 - Use cutpoints and specifications
- Scale and complexity
- Value on simplicity/exposure
- Incrementality wrt change
- Incrementality wrt assurance

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

Size	Run	Time
10.3 MB	edu.afit.planetbaron.client.ChatTestClient	2012-04-26 15:11:46.717387565
9.1 MB	edu.afit.planetbaron.client.ChatTestClient	2010-11-11 15:11:47.470012106
302 MB	edu.afit.planetbaron.server.Server	2010-11-11 15:11:47.471091901

- What fields (non-static) are shared? at 2012-05-11 10:40:54
 - What are the individual instances of this non-static field? at 2012-05-11 10:41:09
 - What threads read and write this field? at 2012-05-11 10:41:20
 - When and by what threads was this field accessed? at 2012-05-11 10:41:33**
 - What is the stack trace for this access? at 2012-05-11 10:41:42

- What is the stack trace for this access?
- What locks are held during this field access?

Package/Class/Field/Thread	Time	Access	Under Construction
edu.afit.planetbaron.client			
ChatTestClient			
f_connectDisconnectButton			
	2012-04-26 15:11:46.717387565	Write	Yes
	2012-04-26 15:11:47.470012106	Read	Yes
	2012-04-26 15:11:47.471091901	Read	Yes
	2012-04-26 15:11:47.474471311	Read	Yes
	2012-04-26 15:11:47.474534829	Read	Yes
AWT-EventQueue-0	2012-04-26 15:11:57.058021863	Read	
AWT-EventQueue-0	2012-04-26 15:12:02.201386732	Read	
AWT-EventQueue-0	2012-04-26 15:12:02.201525796	Read	

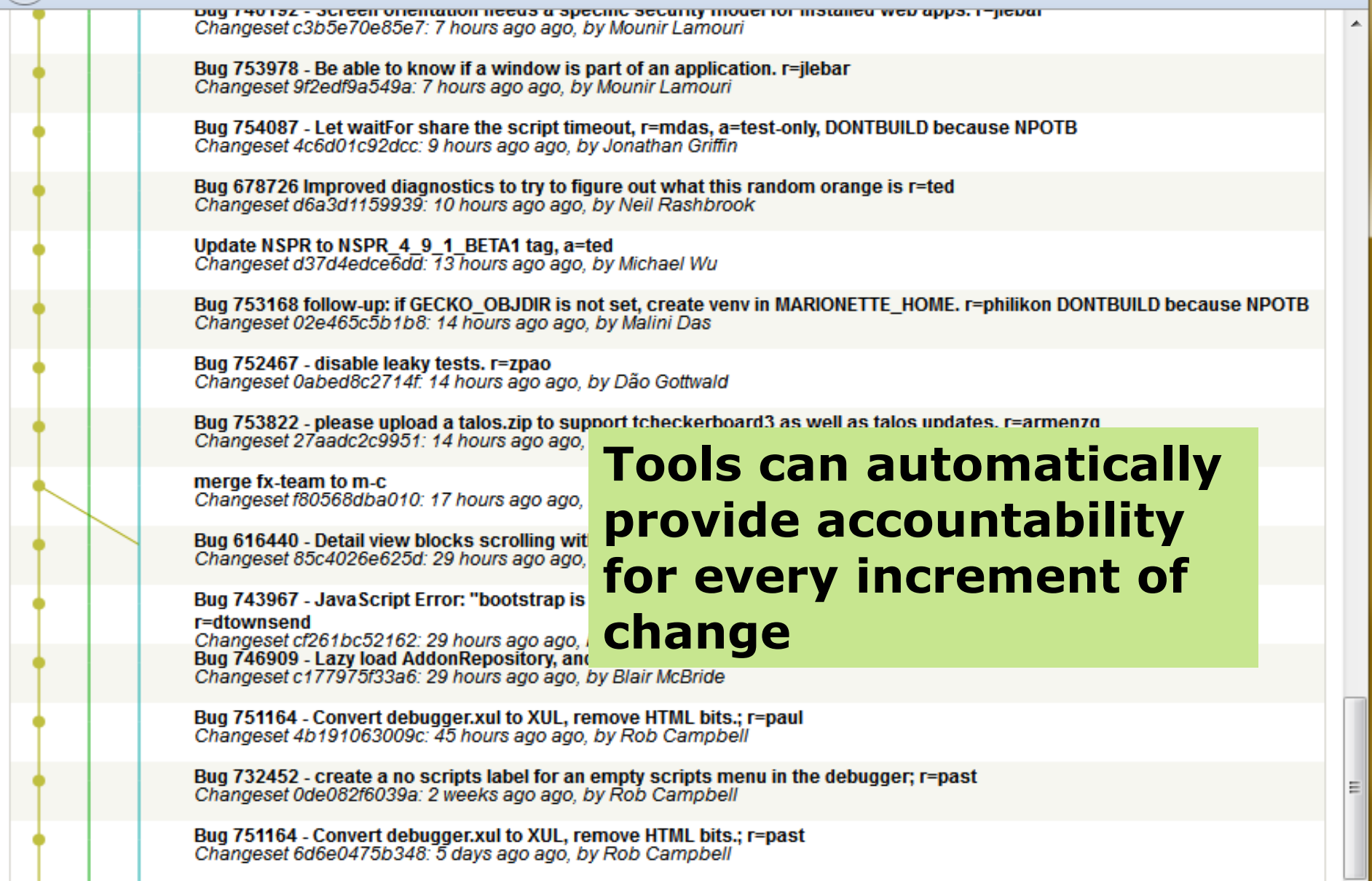
```
PlanetBaron/src/edu/afit/planetbaron/client/ChatTestClient.java at 2012-04-26 15:11:46
65
66 JButton f_connectDisconnectButton = new JButton("Connect");
67
68
```

```

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
philingnalda@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

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
 15:40:11

The chart displays various tasks such as 'cb-sea-linux-01', 'cb-sea-linux-tbox', 'cb-sea-linux-tbox-free-space', 'cb-sea-linux-tbox-clobber', 'cb-sea-linux-tbox-xpcshell', 'cb-sea-linux-tbox-plain-1' through 'plain-5', and 'cb-sea-linux64-01'. Each task includes a terminal icon and a text box with details like 'ENV: SeaMonkey', 'check', and '19458/0'.

**Automated infrastructure
 for builds and tests ... *and
 analytics***

Last 7 days Firefox Top Platforms Top Tests



**A simple example:
Automated
performance tests**

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*

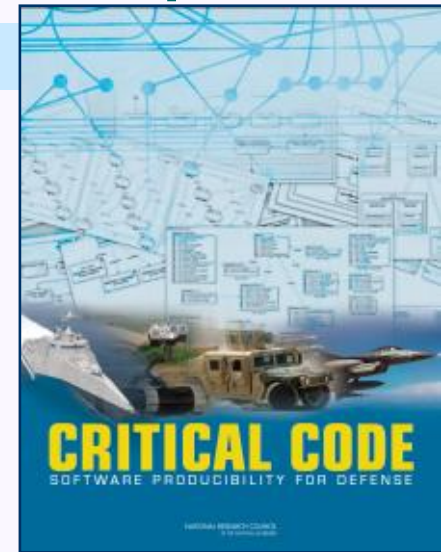
- Architecture modeling and architectural analysis
- Validation, verification, and analysis of design and code
- Process support and economic models for assurance
- Requirements
- Language, modeling, code, and tools
- Cyber-physical systems
- Human-system interaction

Challenge issues

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

3. **Leadership** – *Never relinquish the innovation lead*

- **Recognize the unboundedness of software**
- **Stay ahead in assurance** (cf. DSB’07)
- **Sustain innovation and ecosystem lead**



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