COMP314:

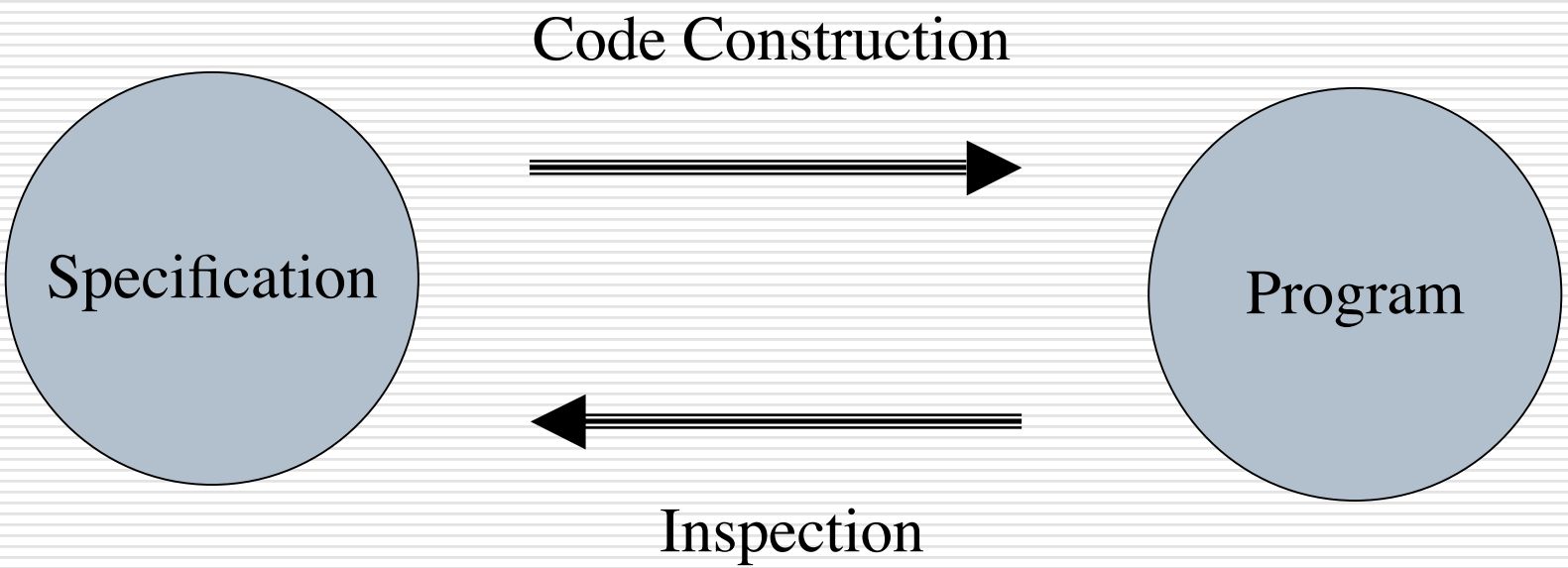
Virtuous Cycle

Cycles of activity and software projects

Automation and tools in a software company

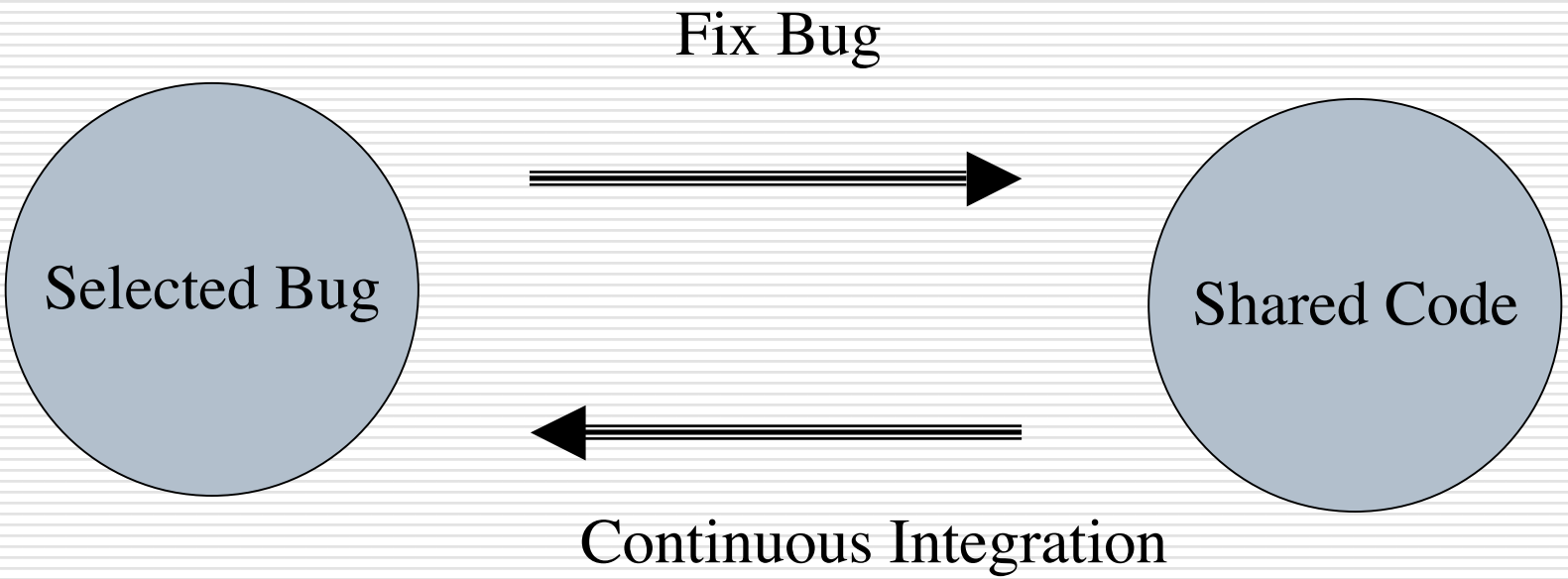
Testing as part of feedback cycles

Programming Cycle



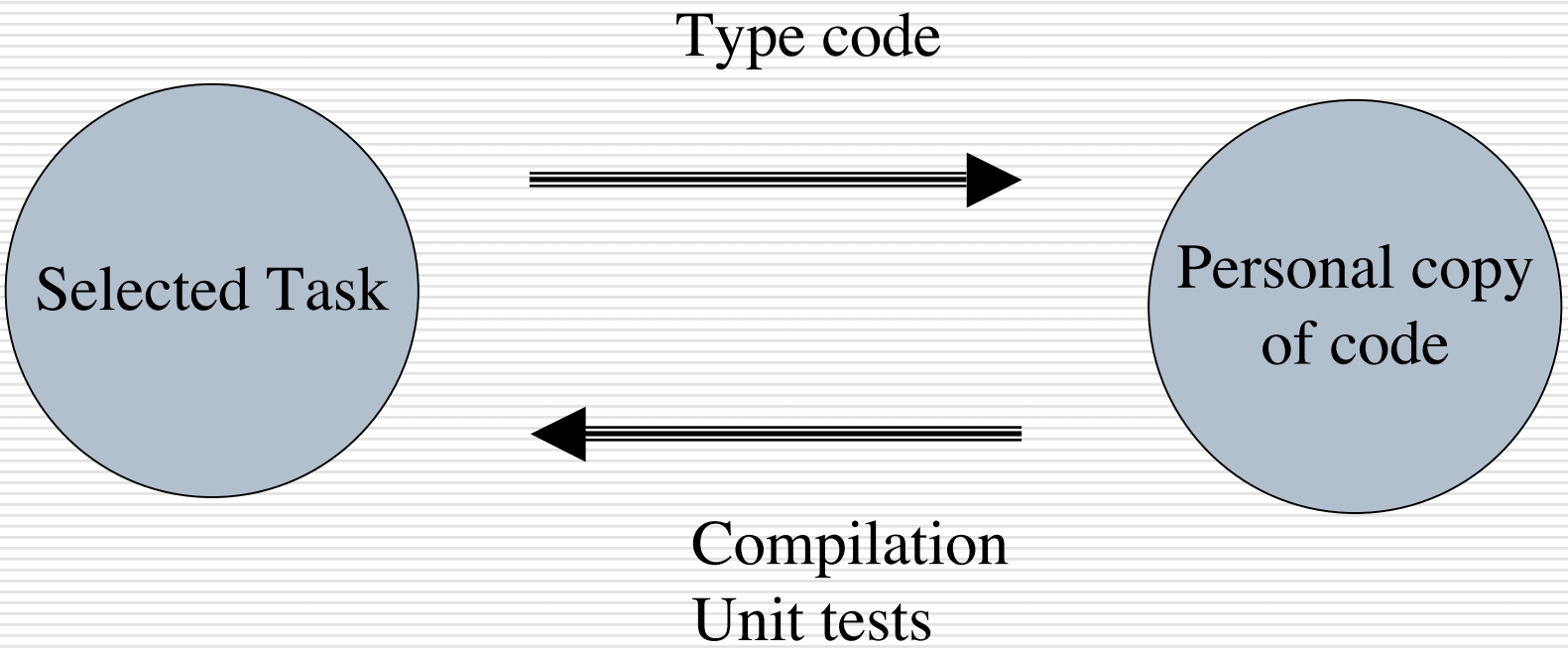
Programming Cycle - single bug

~8 hours

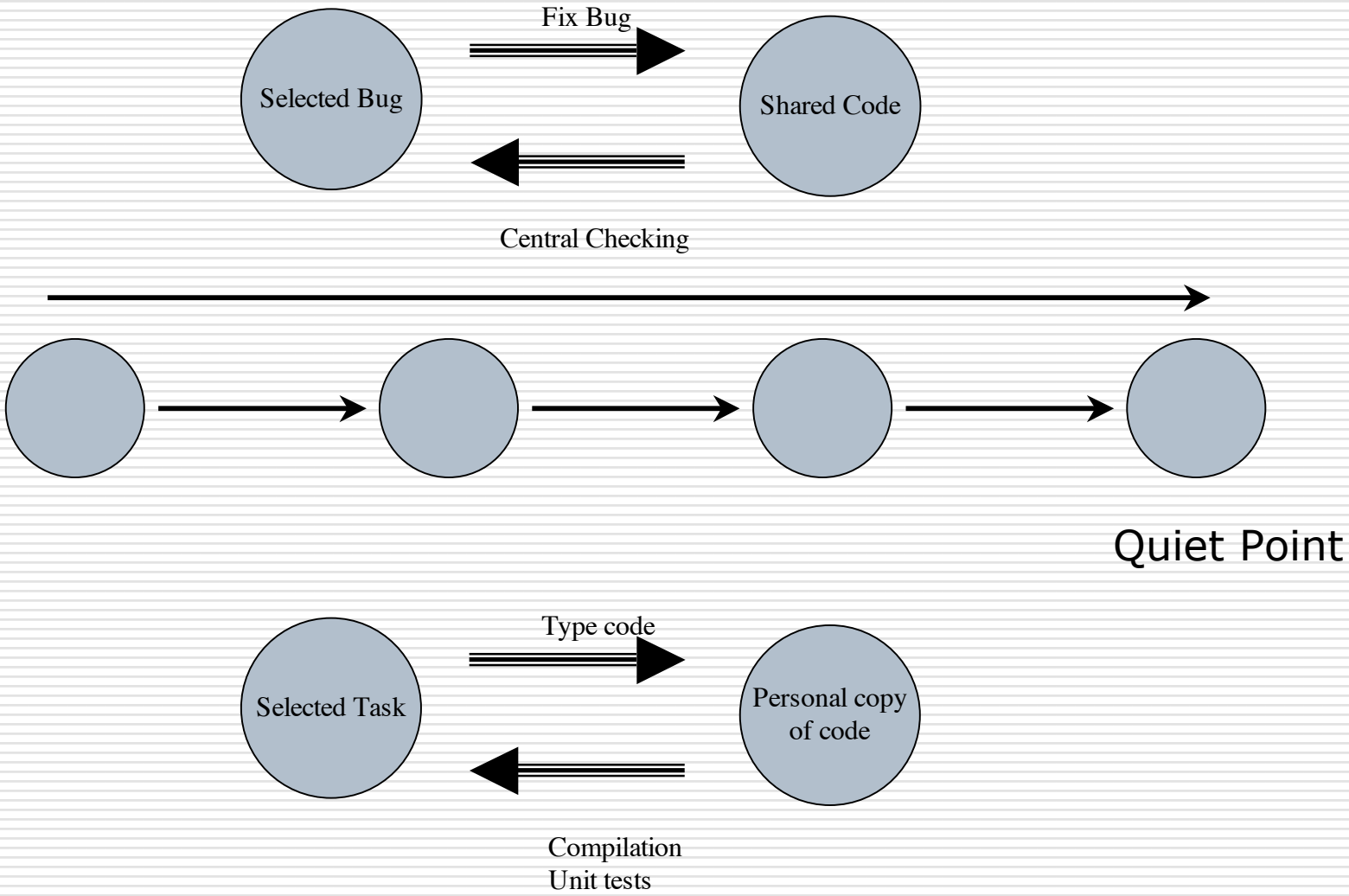


Programming Cycle - 10s lines of code (personal task)

~10s minutes



Multiple levels of cycles



A Virtuous Cycle

- High bandwidth – get as much feedback as possible
- Low Latency – get feedback as soon as possible

Examples of interesting cycles

- Innovation cycle - development of new products and ideas
- Science - selection of ideas and deliberate testing of them

A Very Short History of Software Engineering

- **October 1968**

  NATO Workshop term "Software Engineering" coined
- **1960's - now**

  Lots of fumbling around in the dark
- **1990s - now**

  "Extreme programming" or "Agile programming" – keep it light, simple, fast, only do what is needed

NetValue Development

- Tools that are used as part of the development cycle
  - cvscheck
    - compilation
    - style checking
    - testing
  - javadocs
    - documentation
  - jumble
    - quality of unit testing

NetValue Development

Continuous Integration

- cvscheck - Source Code Control and Build
  - Shared
  - Centralized
  - Automatic

cvscheck_cartesian Code Monitor

The system currently checks the code out of version control, compiles the src and test hierarchies separately, runs the unit tests under various jvms, generates javadocs for the src and test hierarchies. It performs additional checking that all test classes are linked into the root tester, and that each class has at least rudimentary documentation.

Current Status: Problems on other platforms
Latest Check: Level 5 started at 2008-07-20 14:30

- No changes in CVS
- cartesian/src
  - Clean compile (all output)
  - Documentation OK (all output) -- View JavaDocs
- cartesian/internal
  - Clean compile (all output)
- cartesian/test
  - Clean compile (all output)
  - AllTests linked correctly
- Unit Test Results

Platform	Results	Comment
Linux x86 2.4.x	Tests run OK (1724 tests) (stderr)	
MacOS X 10.4.7 (G4 1.4GHz 1GB)	No test summary in results file (stderr)	Last updated: Sep 17 2007
MS HPC	Tests run: 1724, Failures: 4, Errors: 557 (stderr)	Last updated: Jul 19 16:44
Windows2008 Server x64	Tests run: 1724, Failures: 1, Errors: 0 (stderr)	Last updated: Jul 19 23:02

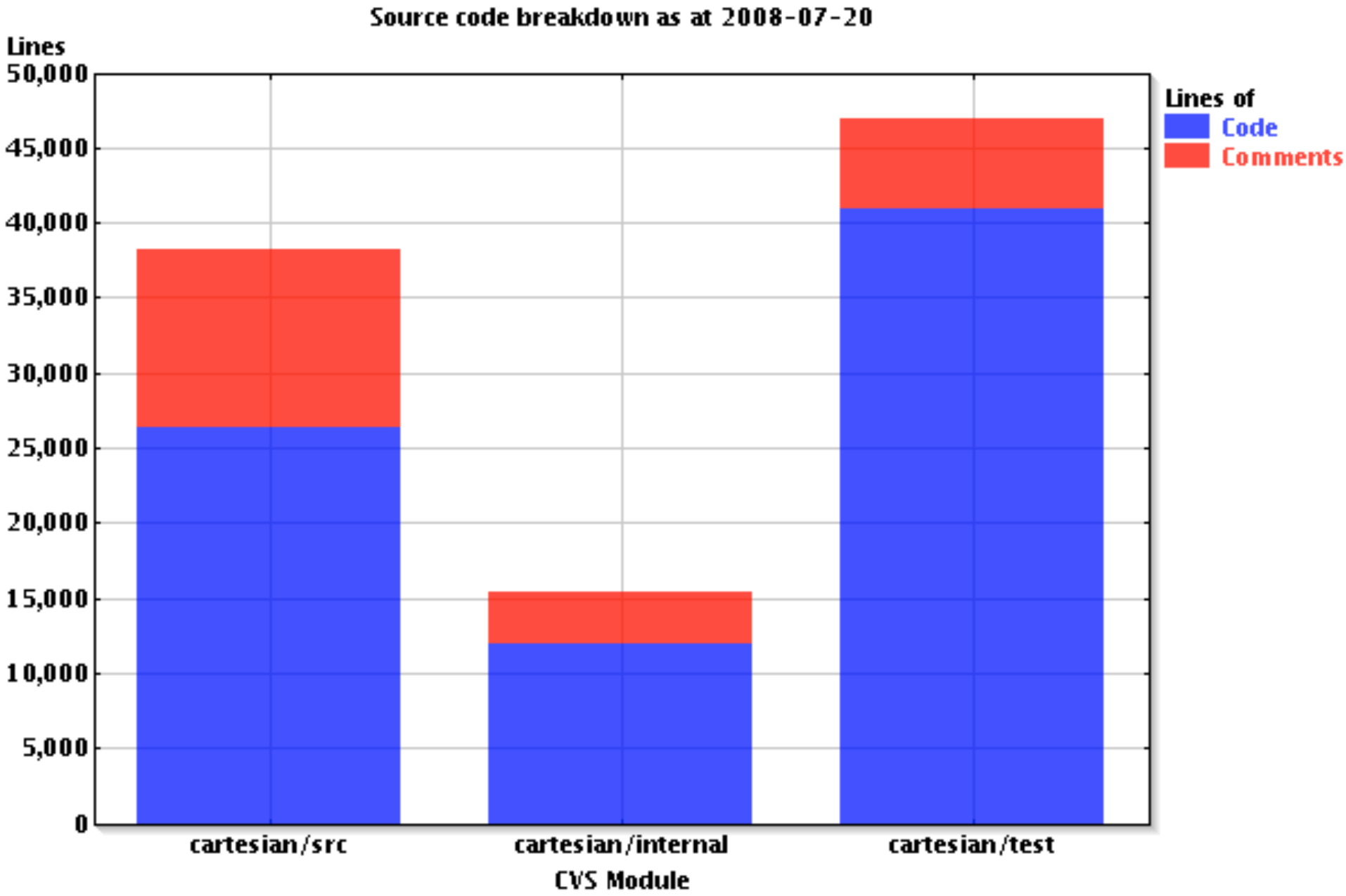
- Graphical status history
- Unit test history
- Source code stats
- Package dependence summary
- Jumble unit test quality analysis (log, queue:classes, packages, prequeue)
- Emma coverage results
- log from this run (previous log)

Historical Table

date	CVS update	src compile	src docs	internal compile	test compile	test linkage	tests	ant build	Regression
2008-07-18 17:00	6 files changed						OK (1724 tests)	OK	ERROR
2008-07-18 16:30	18 files changed						OK (1724 tests)	OK	ERROR
2008-07-18 16:00	4 files changed				1 warnings		OK (1724 tests)	OK	ERROR
2008-07-18 15:30	3 files changed			417 warnings	15 warnings		OK (1724 tests)	OK	ERROR
2008-07-18 15:10	9 files changed			417 warnings	15 warnings		Tests run: 1724, Failures: 2, Errors: 0	OK	ERROR
2008-07-18 14:30	125 files changed			417 warnings	32 warnings	1 files unlinked	OK (1721 tests)	OK	ERROR
2008-07-18 13:50	164 files changed	1 warnings		417 warnings	31 warnings	2 files unlinked	OK (1721 tests)	No build status	ERROR
2008-07-18 13:20	3 files changed	1 warnings		444 warnings	947 warnings		OK (1721 tests)	No build status	ERROR
2008-07-18 13:10	1 files changed	5 warnings		432 warnings	895 warnings		Tests run: 1721, Failures: 2, Errors: 0	No build status	ERROR

cvscheck - 15mins

- Compilation
  with and without test code
- Running unit tests
  - full set of 1,700 tests every 15 mins
  - In jar file and locally
- Comments to html documentation


Source code breakdown as at 2008-07-20
Lines
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
Lines of
Code
Comments
cartesian/src
cartesian/internal
cartesian/test
CVS Module

cvscheck cont.

- Source code checks
  spelling, style, variable name conventions
- Full release build
  - Every 6 hours
  - Tested on four platforms (Linux, MacOSX, Windows HPC, Windows Server)

Testing – If you can't measure it why should you believe it?

Test in multiple ways:

- Unit tests
- "assert" statements
- Pre/post conditions
- Others …

Unit Tests

- Independent
  important when isolating problems
- Can be run automatically
  speeds process of integration and makes cycles faster

What are unit tests easiest for?

- “Small” self contained classes with clean interface and well defined state
- Facilities for constructing more complex environments
- Can easily set up many tests for different classes that implement the same interface

What are unit tests good for?

- Fast feedback on system wide errors in a shared environment
- Documenting usage
- New hires - what are the local coding standards?

What do unit tests find hard

- Non-deterministic code
- Timing dependent code
- Strongly system and environment dependent code
- GUI (human artistic judgment)

  although regression testing is possible

How good are our unit tests?

- Necessary to test coverage of tests.
- “Coverage” of lines not enough – just because a line was executed and didn’t throw an exception doesn’t tell us much
- Hence use mutation testing where code “mutated” and unit tests run to check if the mutation found.

jumble

- Jumble - Computes score for how well testing covers code - runs incrementally

Now an open source project.

Combination of work by Reel Two, student projects and department.

Module: cvscheck_cartesian
Package: com.reeltwo.cartesian.index (enqueue all classes in this package)

Average jumble score: 92%
Last updated: 2008-07-18 16:39:52

/com/reeltwo/cartesian/index/

93% collection
93% hash
96% params
60% similarity

100% [64] BitVector (2008-07-18)
100% [12] HashBitHandle (2008-07-18)
100% [17] HashBitVector (2008-07-18)
100% [0] IdPositionTranslation (2008-07-17)
100% [0] IdTranslation (2008-07-17)
88% [304] IndexImplementation (2008-07-18)
100% [27] IndexImplementationUtils (2008-07-18)
81% [93] QuickSort (2008-07-18)
85% [174] Utils (2008-07-18)

Jumble results for com.reeltwo.cartesian.index.IndexImplementation

Enqueue this class for a jumble run
View JavaDoc

```
Jumble run at 2008-07-18 14:42:21
Mutating com.reeltwo.cartesian.index.IndexImplementation
Tests: com.reeltwo.cartesian.index.IndexImplementationTest
Mutation points = 304, unit test time limit 5.83s
.M FAIL: com.reeltwo.cartesian.index.IndexImplementation:159: CP[30] "Size must be positive:" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:162: CP[35] "Hash bits set to invalid value bits=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:164: CP[38] "Index_initialization" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:168: CP[41] 2147483647 -> -2147483648
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:171: CP[44] "Threshold must be positive:" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:189: CP[67] -9223372036854775808 -> -9223372036854775807
..M FAIL: com.reeltwo.cartesian.index.IndexImplementation:219: CP[80] "Index_sort" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:223: CP[82] "Index_overflow" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:226: CP[84] "Index_position" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:229: CP[86] "Index_bitVector" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:254: CP[90] -2147483648 -> -2147483647
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:317: CP[95] "count=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:317: CP[96] " i=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:352: CP[98] "i=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:352: CP[99] " low=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:352: CP[100] " high=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:364: CP[102] "k=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:364: CP[103] " hash=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:364: CP[104] " srbits=" -> "___jumble___"
.........................M FAIL: com.reeltwo.cartesian.index.IndexImplementation:539: CP[152] "Inconsistency in memory calculation bytes()=" -> "___jumble___"
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:539: CP[153] " total=" -> "___jumble___"
................................M FAIL: com.reeltwo.cartesian.index.IndexImplementation:161: 1 -> 0
.M FAIL: com.reeltwo.cartesian.index.IndexImplementation:161: 64 (@) -> 65 (A)
.....M FAIL: com.reeltwo.cartesian.index.IndexImplementation:186: 0 -> 1
............T.....................................
............M FAIL: com.reeltwo.cartesian.index.IndexImplementation:307: 0L -> 1L
.....M FAIL: com.reeltwo.cartesian.index.IndexImplementation:315: negated conditional
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:315: 0L -> 1L
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:315: negated conditional
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:316: 1L -> 0L
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:316: + -> -
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:318: + -> -
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:319: * -> /
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:319: + -> -
M FAIL: com.reeltwo.cartesian.index.IndexImplementation:320: 0L -> 1L
...T......T.........T............................
.............................................
T.............................
Score: 88%
```

Other testing

- "assert" statements at loop/statement level
- Integrity constraints (pre/post-conditions) at instance/class level

---

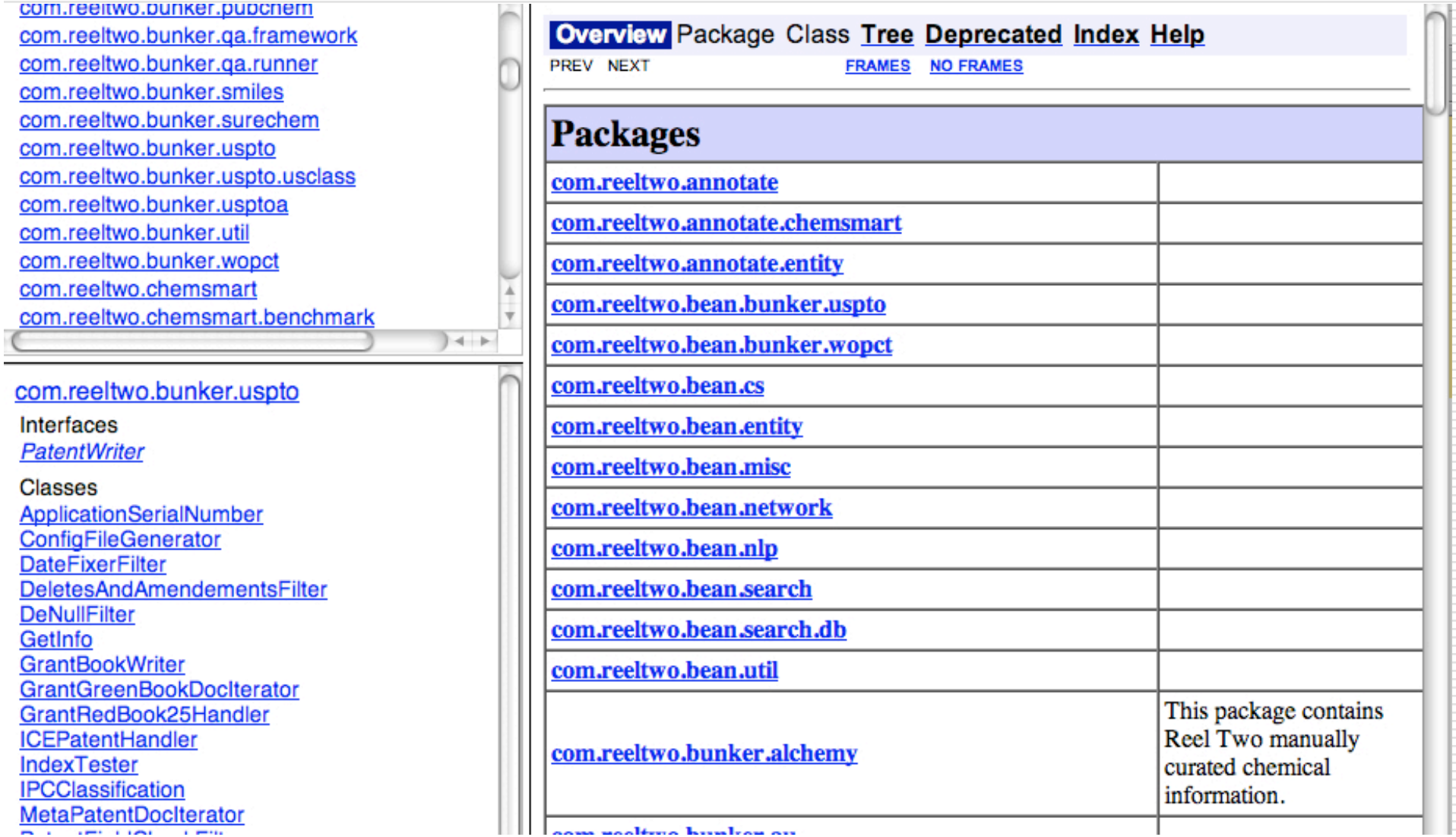
- Good techniques for non-deterministic code (early detection of failure)
- Interacts well with unit tests

Documentation of code

- Use the standard “javadoc” system
- checkstyle will insist on all public methods and classes being documented.
- I will insist that the documentation be good quality

Good documentation

- Documenting data structures is more important than operations on them
- Too much documentation is bad
- Document *why* a method exists rather than what it does


com.reeltwo.bunker.pubchem
com.reeltwo.bunker.qa.framework
com.reeltwo.bunker.qa.runner
com.reeltwo.bunker.smiles
com.reeltwo.bunker.surechem
com.reeltwo.bunker.uspto
com.reeltwo.bunker.uspto.usclass
com.reeltwo.bunker.usptoa
com.reeltwo.bunker.util
com.reeltwo.bunker.wopct
com.reeltwo.chemsmart
com.reeltwo.chemsmart.benchmark
com.reeltwo.bunker.uspto
Interfaces
PatentWriter
Classes
ApplicationSerialNumber
ConfigFileGenerator
DateFixerFilter
DeletesAndAmendementsFilter
DeNullFilter
GetInfo
GrantBookWriter
GrantGreenBookDocIterator
GrantRedBook25Handler
ICEPatentHandler
IndexTester
IPCClassification
MetaPatentDocIterator
Overview Package Class Tree Deprecated Index Help
PREV NEXT
FRAMES NO FRAMES
Packages
com.reeltwo.annotate
com.reeltwo.annotate.chemsmart
com.reeltwo.annotate.entity
com.reeltwo.bean.bunker.uspto
com.reeltwo.bean.bunker.wopct
com.reeltwo.bean.cs
com.reeltwo.bean.entity
com.reeltwo.bean.misc
com.reeltwo.bean.network
com.reeltwo.bean.nlp
com.reeltwo.bean.search
com.reeltwo.bean.search.db
com.reeltwo.bean.util
com.reeltwo.bunker.alchemy
This package contains Reel Two manually curated chemical information.

Class GrantRedBook25Handler

```
java.lang.Object
  └ com.reeltwo.bunker.patent.AbstractPatentHandler
      └ com.reeltwo.bunker.uspto.USPTOPatentHandler
          └ com.reeltwo.bunker.uspto.USPTOXMLPatentHandler
              └ com.reeltwo.bunker.uspto.GrantRedBook25Handler
```

All Implemented Interfaces:

PatentHandler, XMLPatentHandler, org.xml.sax.ContentHandler, org.xml.sax.ErrorHandler

---

```
public class GrantRedBook25Handler
extends USPTOXMLPatentHandler
```

An XML Handler for the Grant Red Book 25 DTD.

Version:

$Revision$

Author:

Richard Allen

Field Summary

Fields inherited from class com.reeltwo.bunker.uspto.USPTOPatentHandler
RELATIONSHIP_ASSIGNEE, RELATIONSHIP_ASSISTANT_EXAMINER, RELATIONSHIP_ATTORNEY_OR_AGENT, RELATIONSHIP_INVENTOR, RELATIONSHIP_PRIMARY_EXAMINER

Constructor Summary

	GrantRedBook25Handler()

Method Summary

void	characters(char[] ch, int start, int length)
void	endElement(java.lang.String namespaceURI, java.lang.String localName, java.lang.String qName)

```java
/**
 * Ensure that the working directory for the environment exists and all
 * the remote contents have been copied over.
 * @param remote directory where the remote environment file is to be found.
 * @param local the local working directory.
 * @param name name of the environment.
 * @throws IOException
 */
private void getEnvironment(
    final RemoteFile remote, final File local, final String name)
throws IOException {
```

cvscheck cont.

Cycles and Times

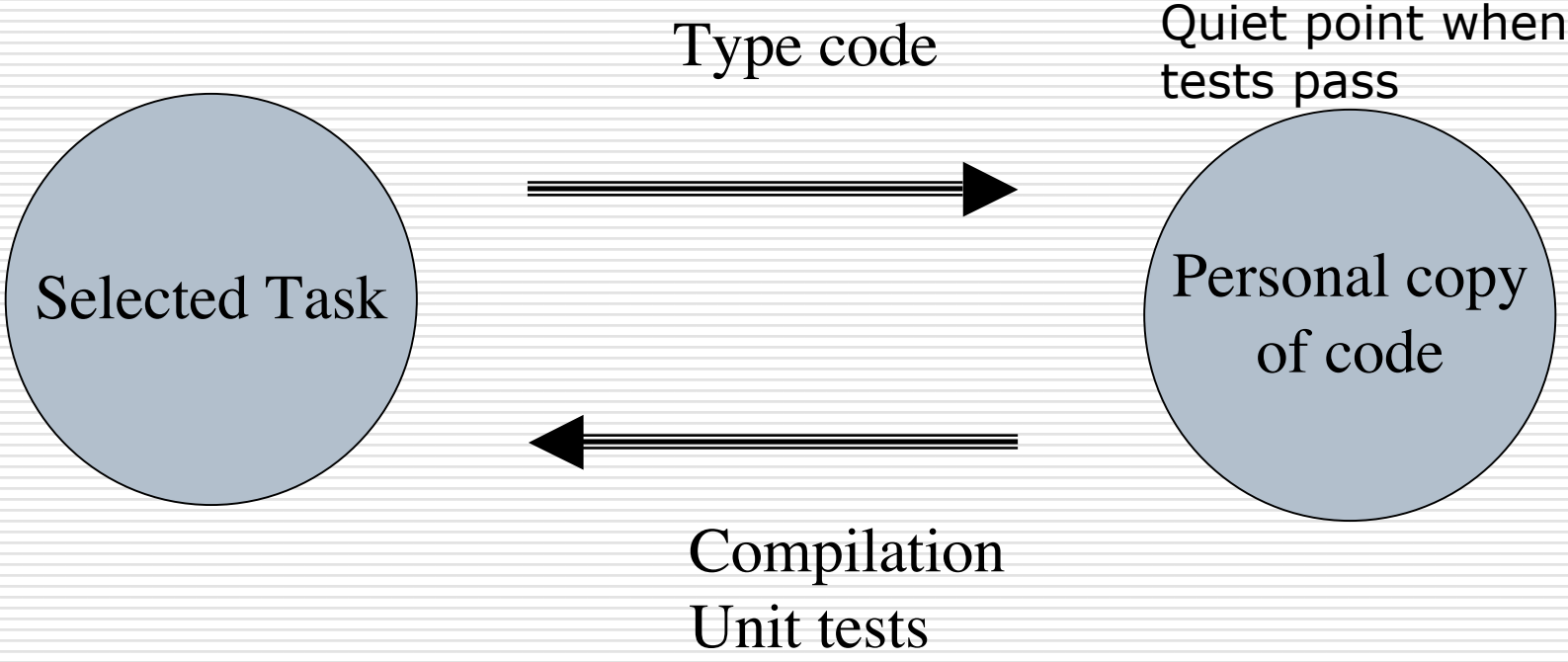
- Code construction unit and style checking - interactive to minutes
- cvscheck - 15mins to 6 hours
- Performance and extreme testing - days
- Release - weeks and months
- Product sales - months to years

Programming Cycle - 10s lines of code (personal task)

~10s minutes

Speed up using IDE

Quiet point when local tests pass

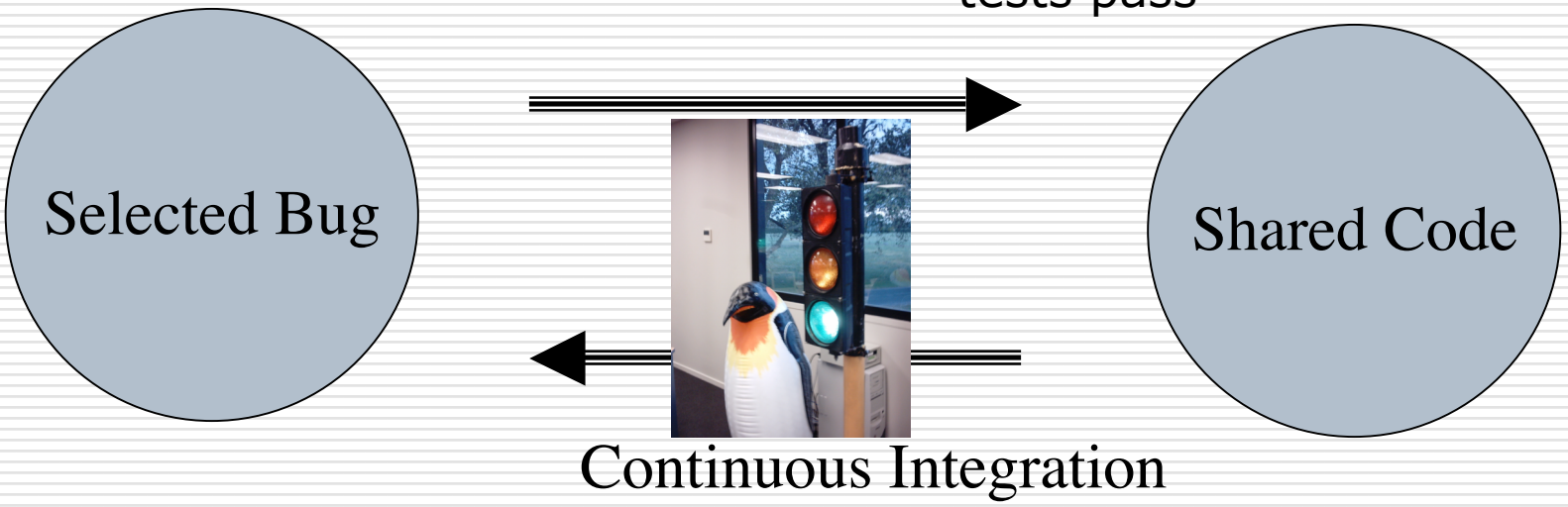


Programming Cycle - single bug

~8 hours

Speed up with svn and cvscheck

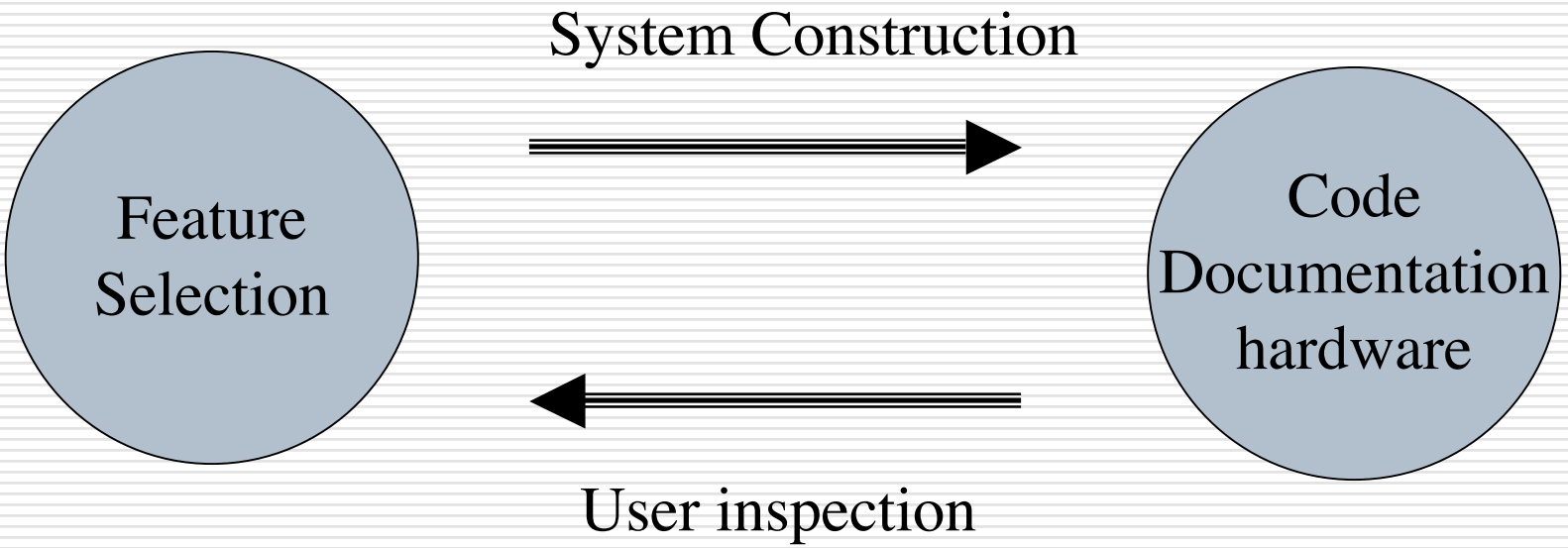
Quiet point when all central tests pass



Release Cycle

~weeks to months

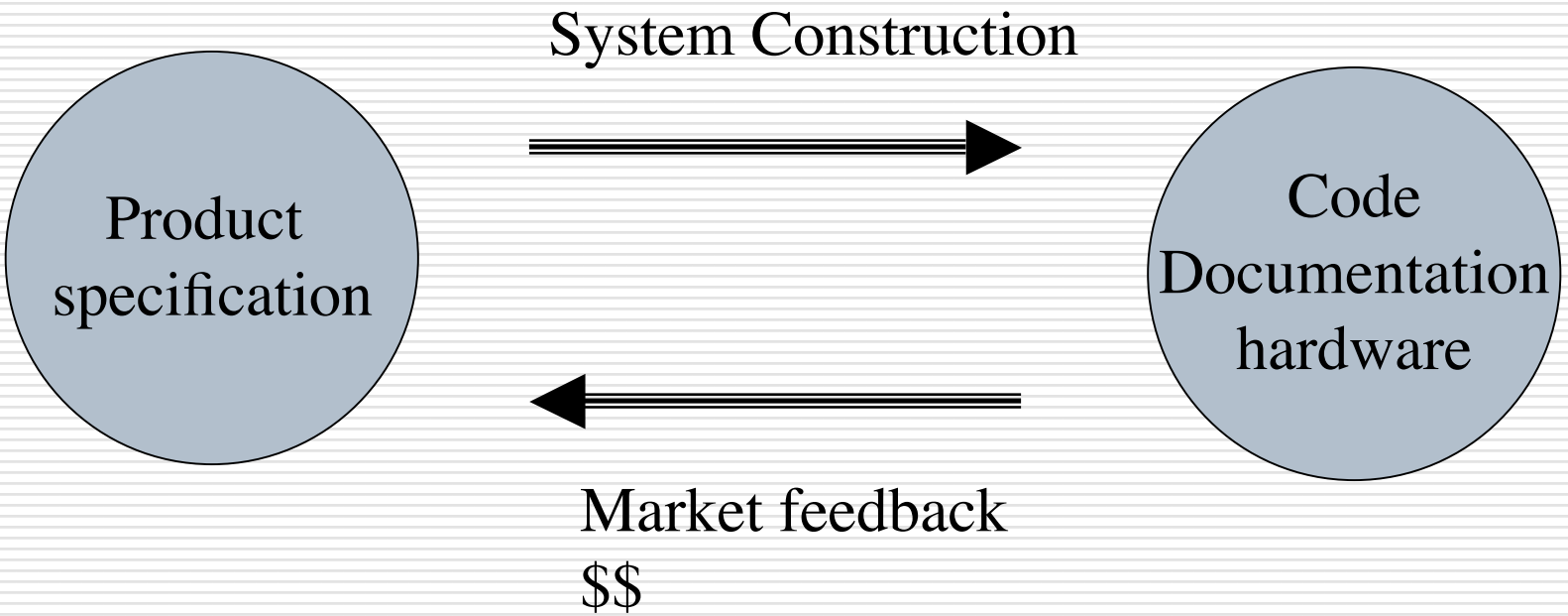
Quiet point when user accepts



Product Cycle

~months to years

Quiet point when getting
Revenue from product



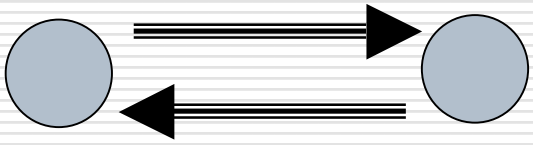
How to improve a cycle

make it fast

- remove unnecessary work
- automate where possible
- plan the quiet points

get as much feedback as possible

- look for “risks”



Programming Iterations

Following our principles above

- make each iteration as short as possible
- less than one week (8 hours programming)
- in some cases hours or minutes

Think about the iteration beforehand

- you may be surprised how small and short they can be

Terms introduced

Cycle - process of performing action and receiving **feedback** on result

Quiet Point - when a cycle has achieved its goal

Continuous integration - cycle where programmers contribute code to a shared repository and global testing is done

Unit tests - tests that are independent and automated

Summary

Can view software projects as cycles of activities

Modern “agile” programming emphasizes the value of short productive cycles

The goal is to deliver value - not to spend time doing useless make work

Automated tools are an important way of speeding and improving such cycles