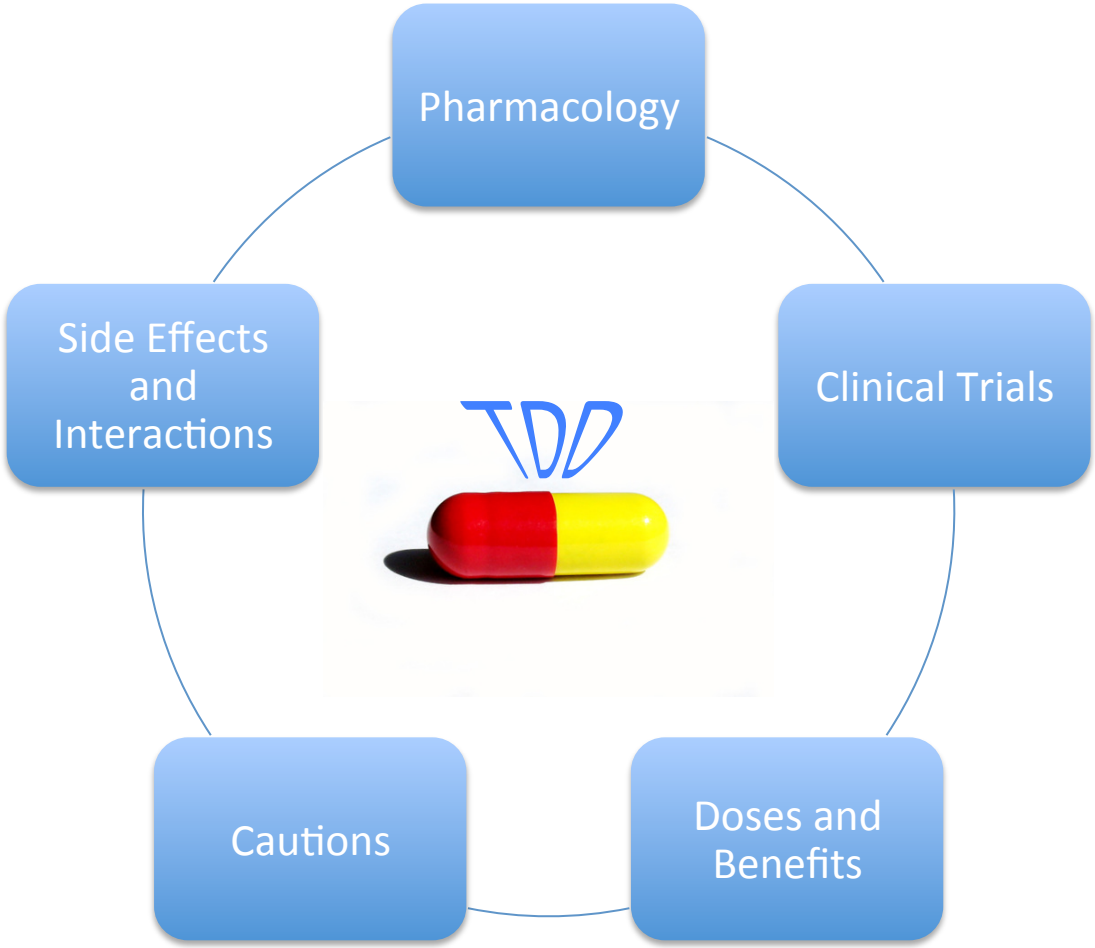
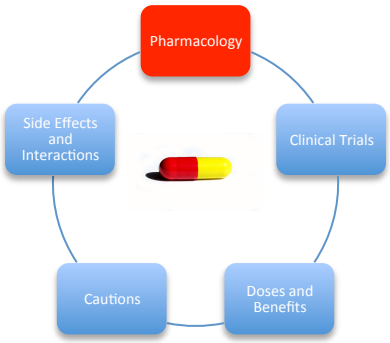
Test-Driven Development: Red-Green-Refactor

Burak Turhan
burak.turhan@oulu.fi

Outline



What's TDD?

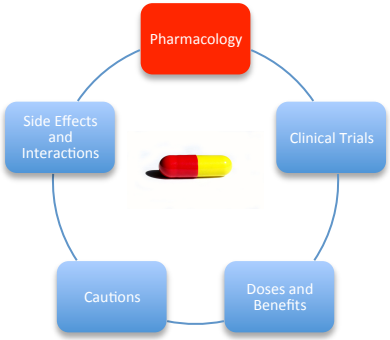


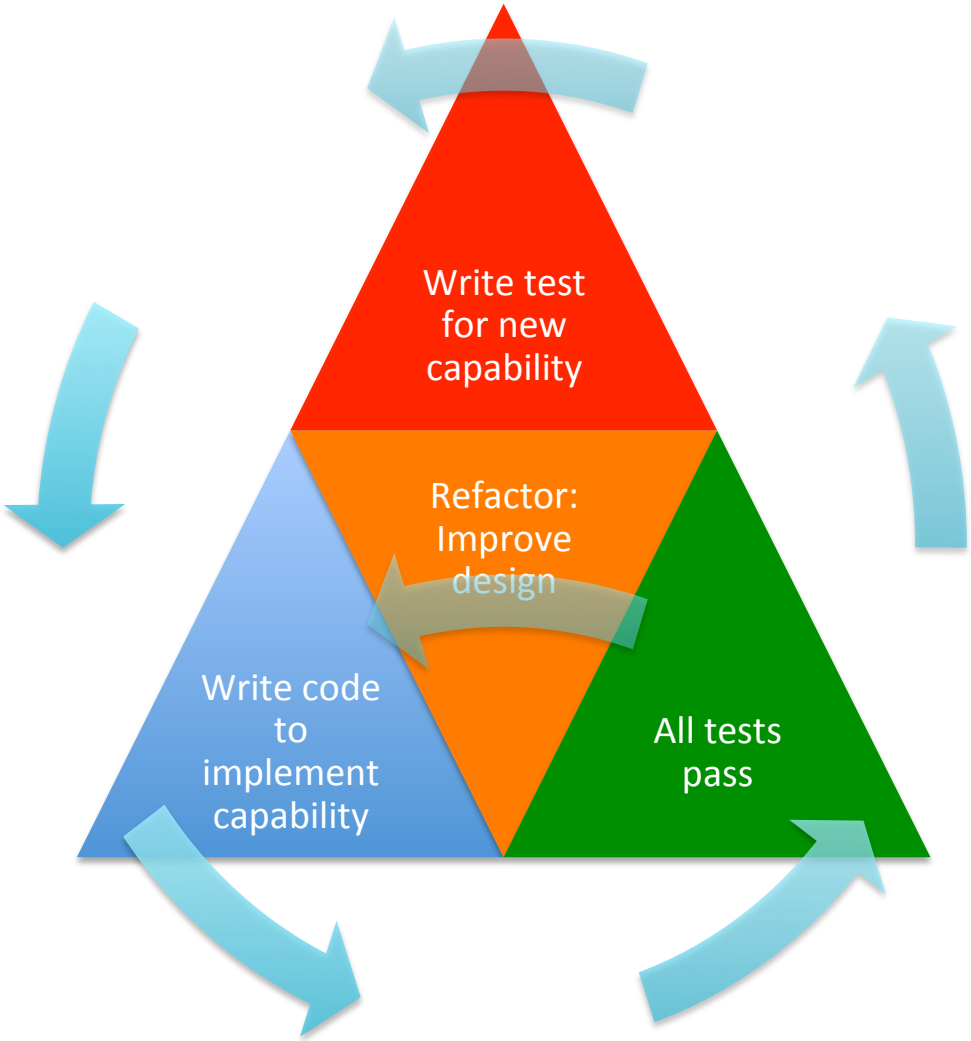
An incremental software development approach
…that relies on automated regression tests
…to ***steer*** development activities

Can be used at different levels:
System/Team: Customers' **acceptance tests** drive **development of new features**
Module/Personal: Programmers' **unit tests** drive **coding**

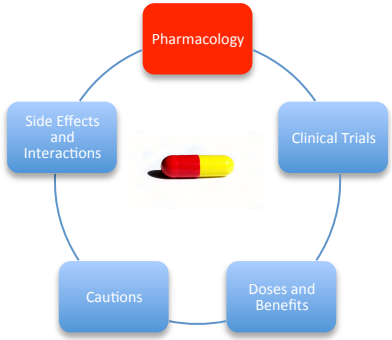
It is a development practice: Improved quality (less errors) and productivity.
It is a design practice: Leads to simpler design
~~It is a testing practice~~: Growing library of regression tests

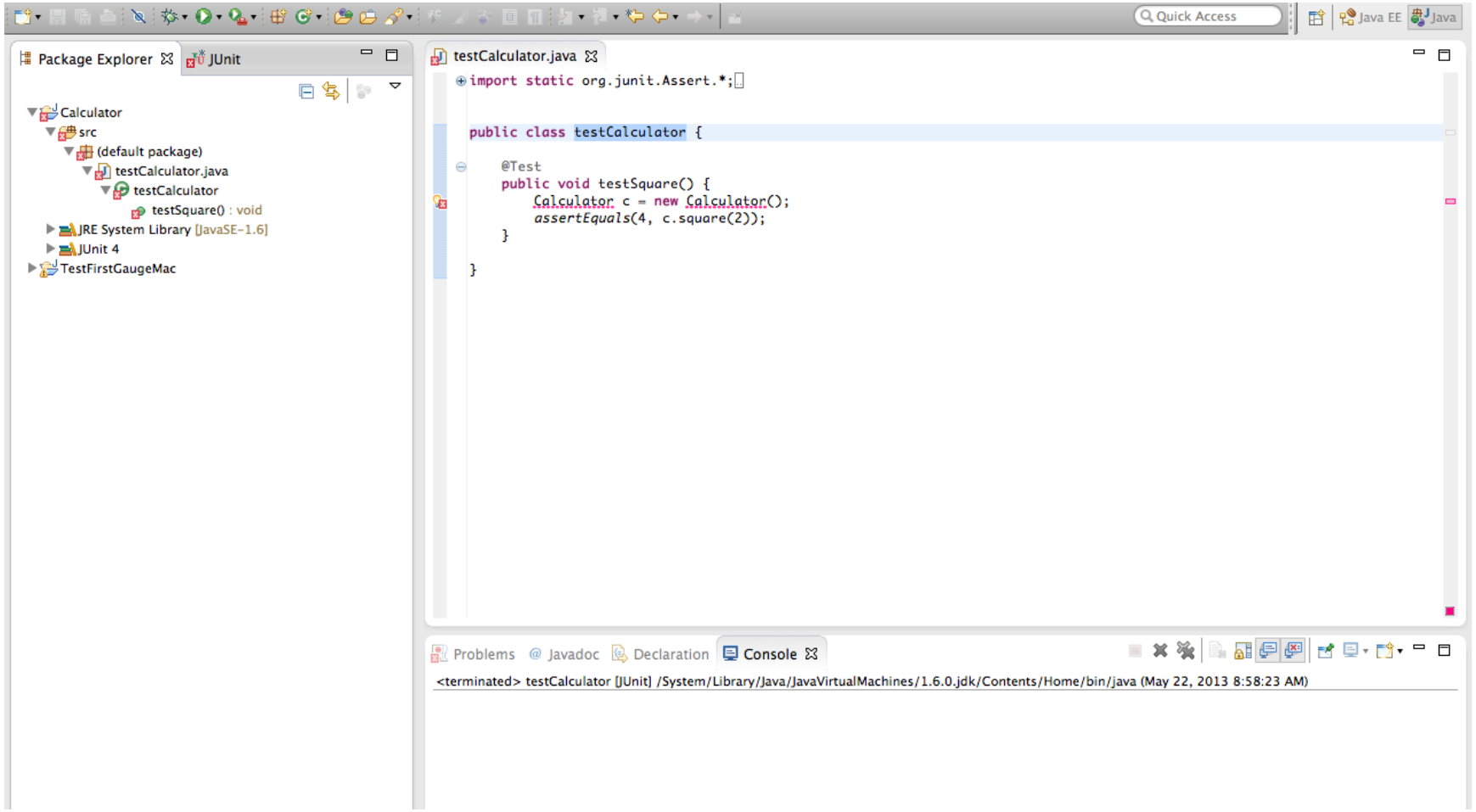
Why is TDD controversial?
Misunderstood -- "blame the name"
Incorporates & blends activities associated with traditionally distinct phases:
Effectiveness?
Too much work, too difficult to learn & apply?
Requires a certain way of reasoning and problem solving



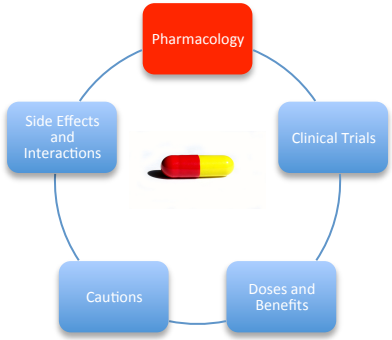


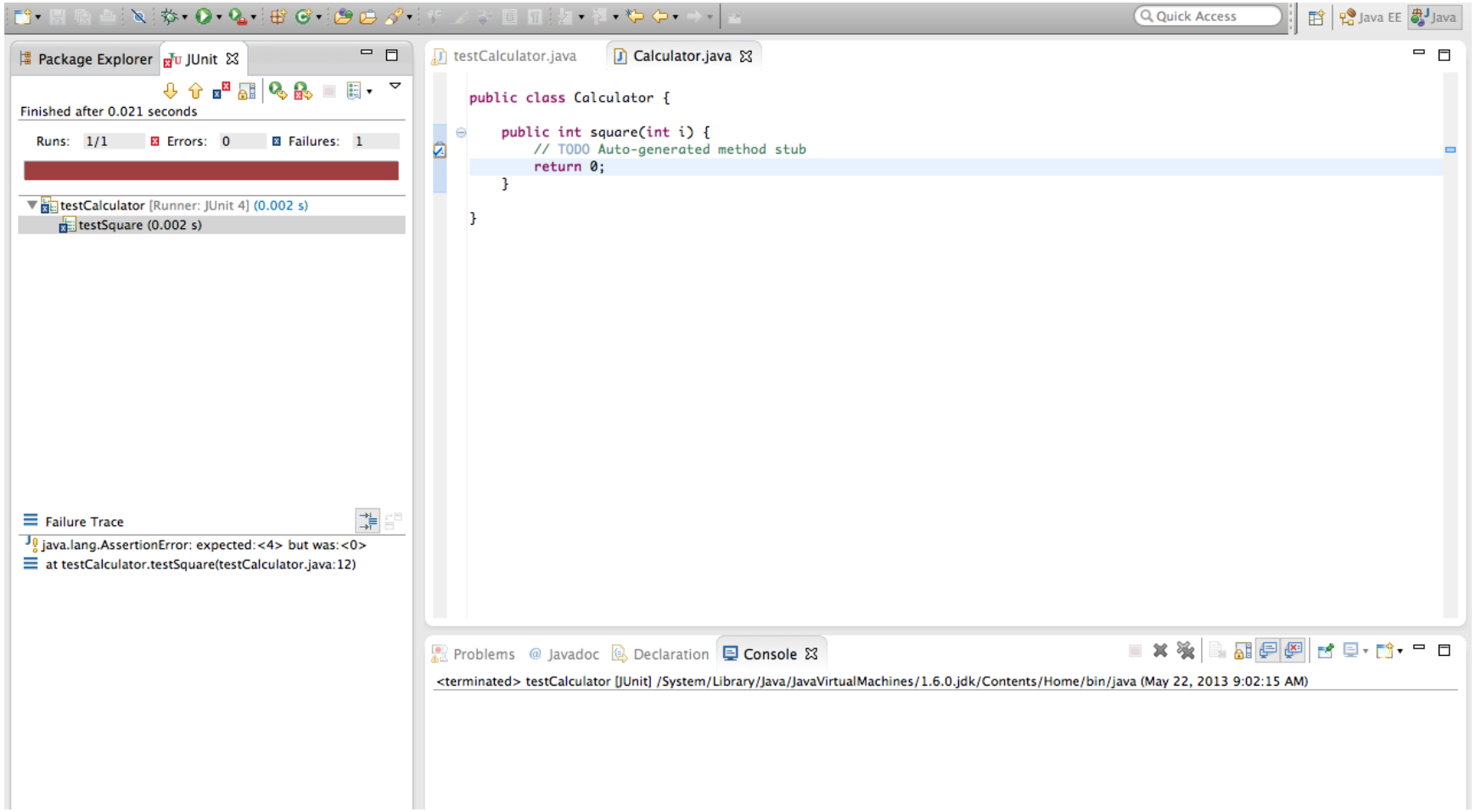
Write a test…



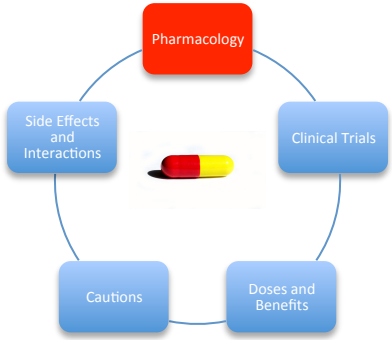


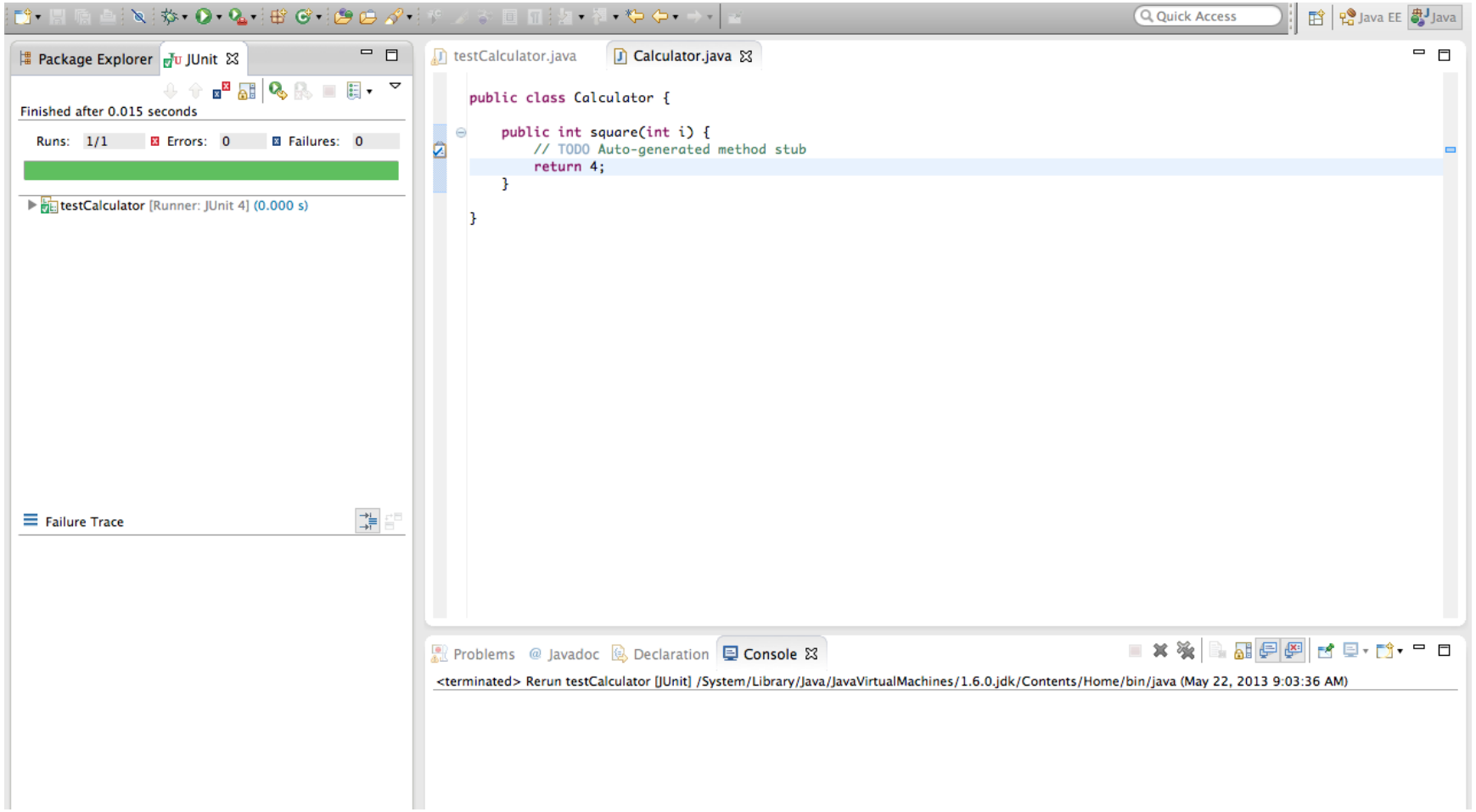
See the test fail (RED)



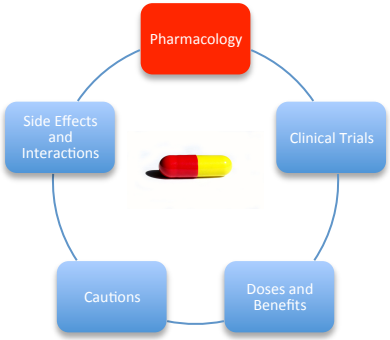


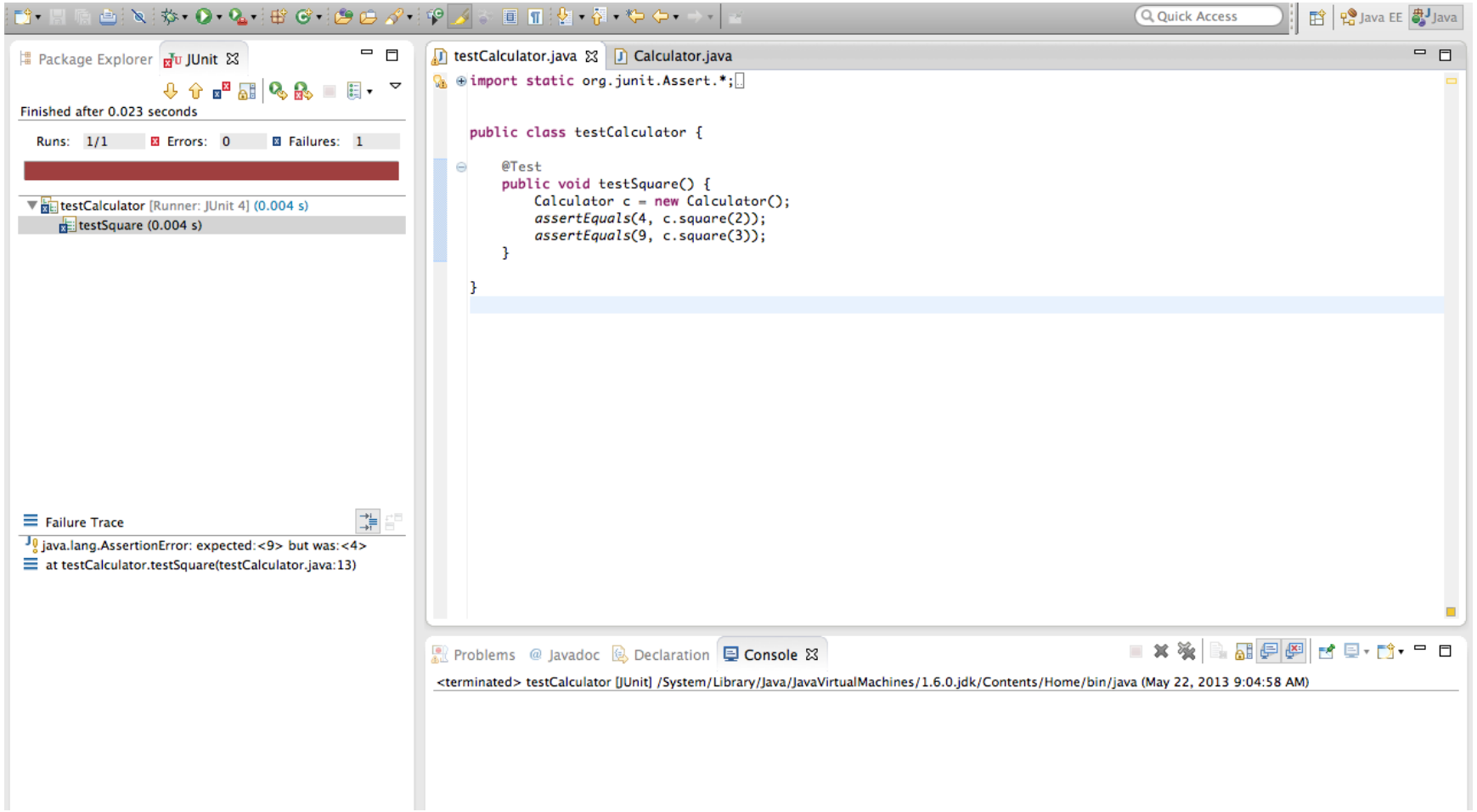
Make the test pass (GREEN)



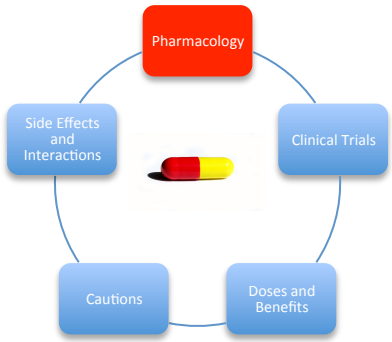


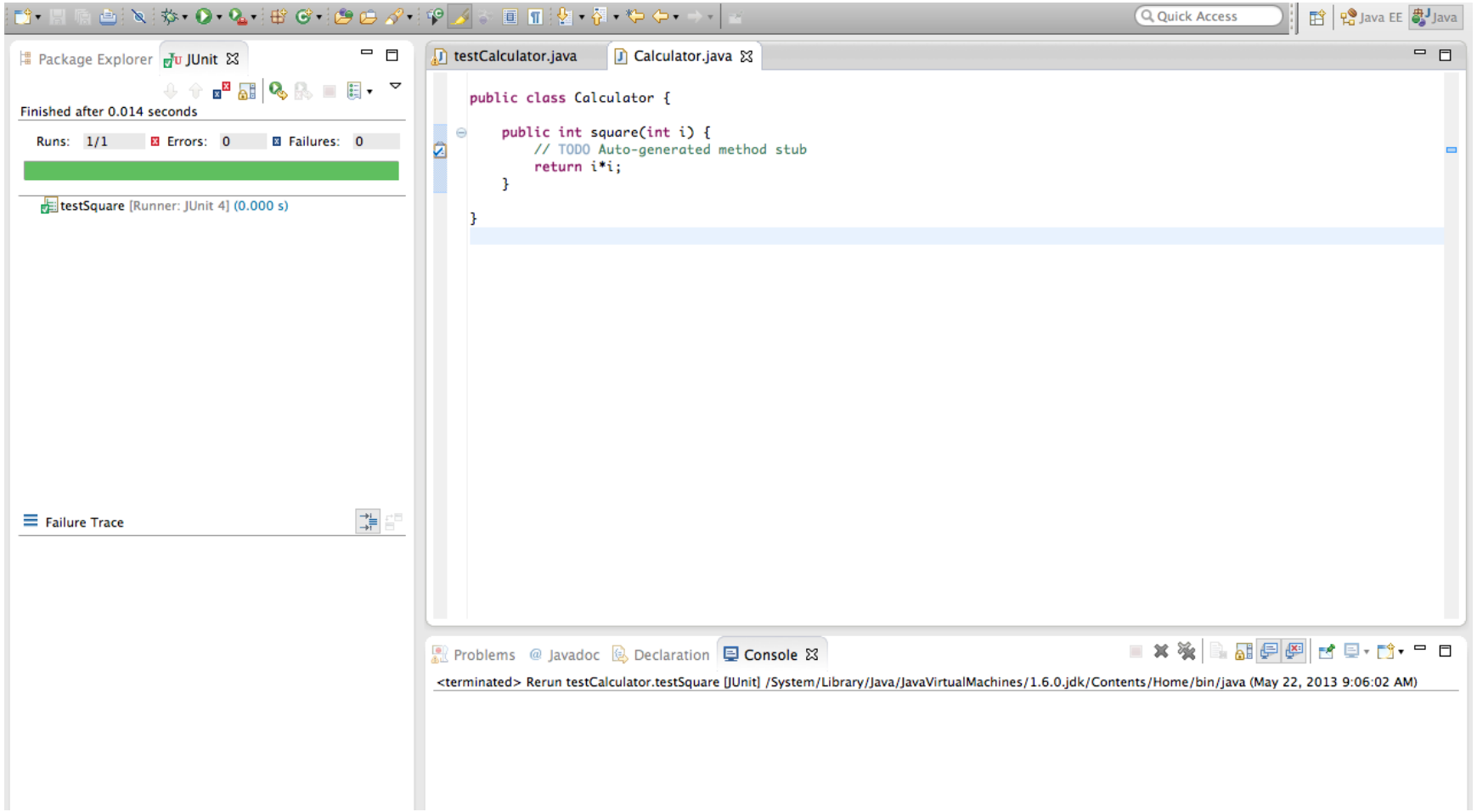
Refactor…



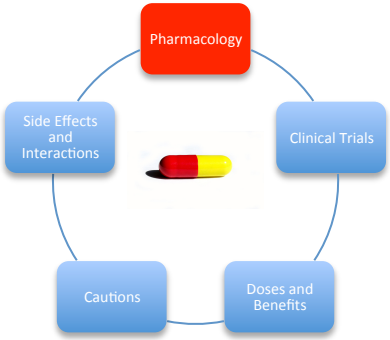


Refactor…

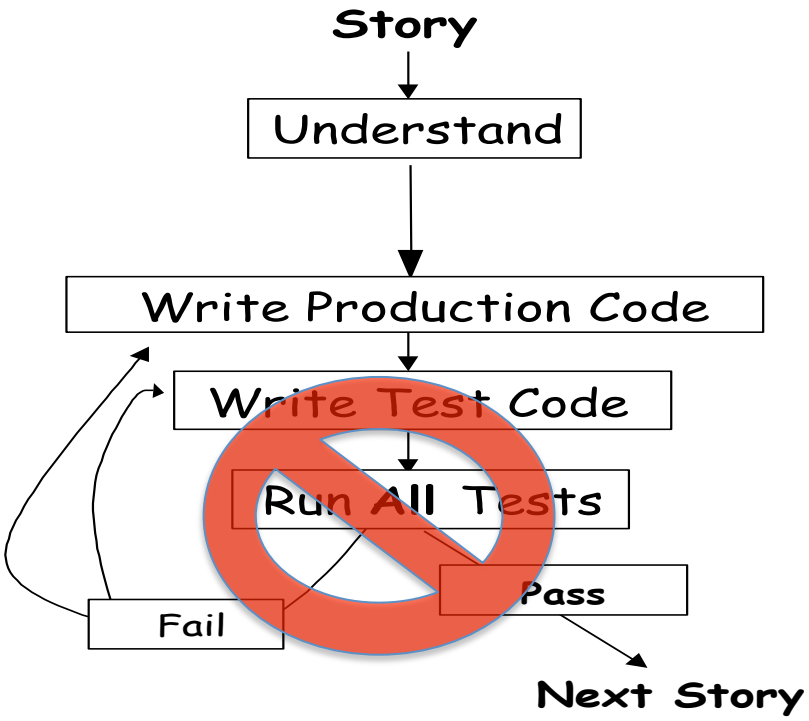




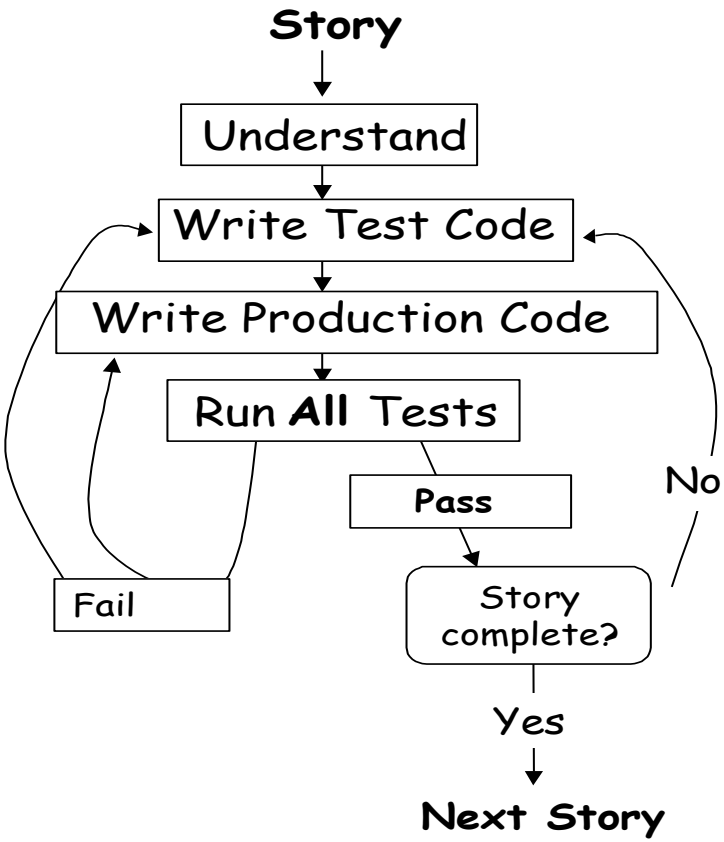
Comparison



Test-Last



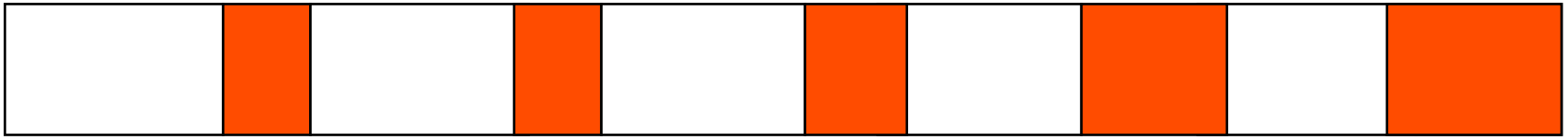
Test-First



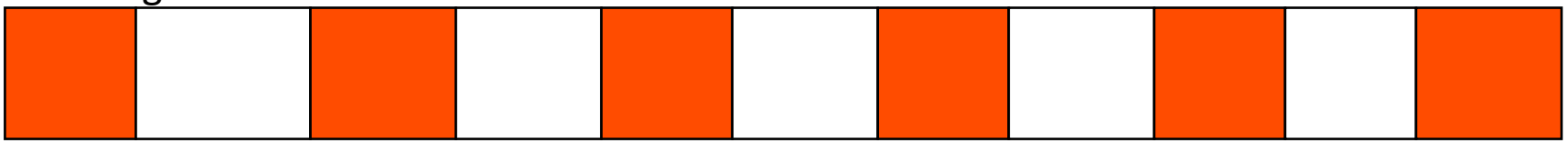
TDD dynamics

Traditional, phased test-last

Incremental test-last:
test-oriented, but non-TD

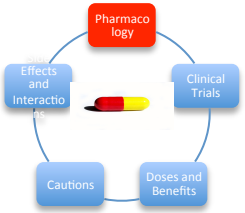


★ Coarse-grained test-first



★ Ideal test-first

Why should TDD work?



Test-first

Incremental

Regressive

Testability

Decomposition

Feedback

Formalization

Timely diagnostic

Visible progress

Test support

Focus

Refactoring

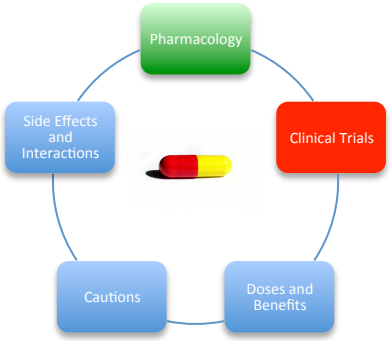
Comprehensibility

Overhead

Productivity

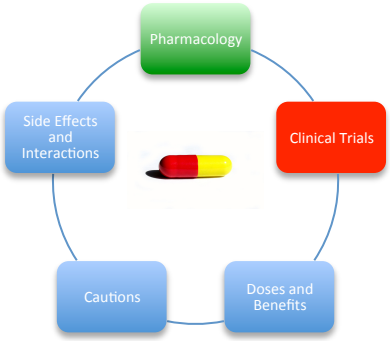
Ext. Quality

Int. Quality



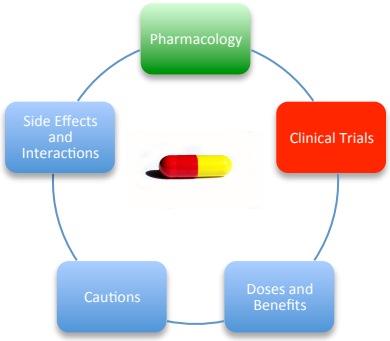
Question is…

Does TDD really improve software quality and/or programmer productivity *compared to* test-last development?



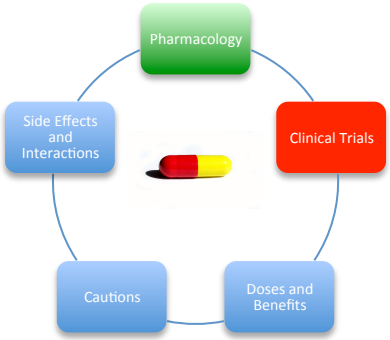
Question is…

Does TDD really improve software quality and/or programmer productivity *compared to* test-last development?



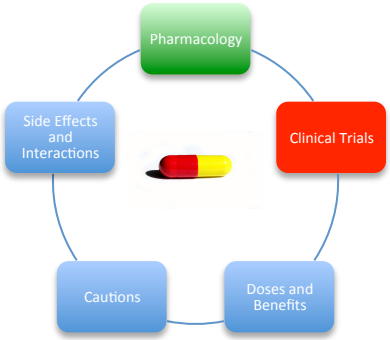
Question is…

Does TDD really improve software quality and/or programmer productivity *compared to* test-last development?



Question is…

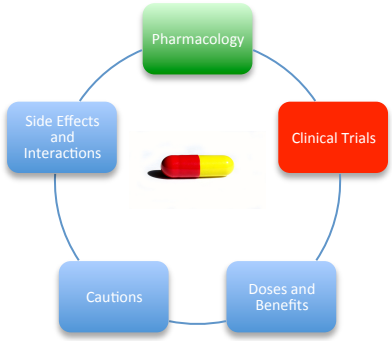
Does TDD really improve software quality and/or programmer productivity *compared to* test-last development?



Question is…

Does TDD really improve software quality and/or programmer productivity *compared to* test-last development?

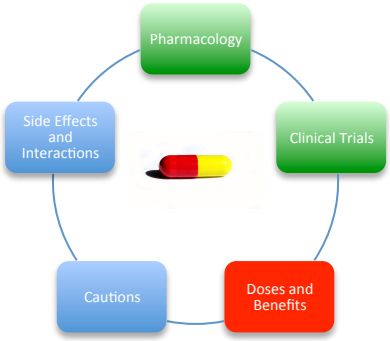
Sources of Evidence



Level	**L0**	**L1**	**L2**	**L3**	
Experience	**Any**	**Any**	**UgradStu**	**GradStu|Profes**	
Construct	**Any**	**Poor|Unknown**	**Adequate|Good**	**AdequateIGood**	
Type *Scale*	**Small**	**Medium|Large**	**Medium|Large**	**Medium|Large**	
Controlled Experiment	2	0	2	4	**8**
Pilot Study	2	0	5	7	**14**
Industrial Use	1	7	0	2	**10**
	5	**7**	**7**	**13**	**32**

- Wide range:
  - Scope: a few to 2600 hours of effort
  - Sample: 1 to 132 participants
  - Small scale: Scope < 170 hours & Sample < 11 participants
  - Medium scale: Scope > 700 hours | Sample > 13 participants
- Several studies with multiple measures

Productivity



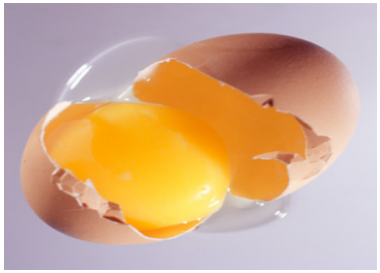
Type	BETTER	NO-DIFF	WORSE	
Controlled Experiment	✓ ✓ ✓	✓		4 2
Pilot Study	✓ ✓ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	14 12
Industrial Use	✓	✓	✓ ✓ ✓ ✓ ✓	7 1
ALL STUDIES:	10	6	9	25
HIGH-GRADE STUDIES:	**6**	**4**	**5**	**15**

Expectation: TDD incurs a productivity penalty

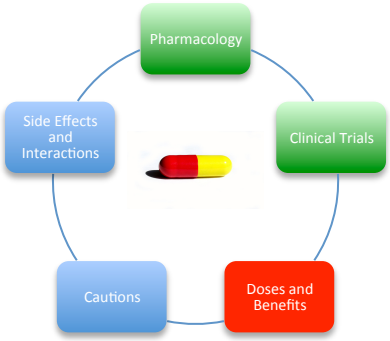
Intuition: Steep learning curve, overhead of writing and managing tests

- Experiments favor TDD, pilots mixed, and industrial studies do not favor TDD
- Same results regardless of study rigor

TDD doesn't have a consistent negative effect



Internal quality



Type	BETTER	NO-DIFF	WORSE	
Controlled Experiment	✓	✓ ✓		3 2
Pilot Study	✓ ~ ~ ~	✓ ✓	✓ ~ ~ ~	7 5
Industrial Use	✓ ✓ ✓		✓	4 2
ALL STUDIES:	6.5	4	3.5	14
HIGH-GRADE STUDIES:	**2.5**	**4**	**2.5**	**9**

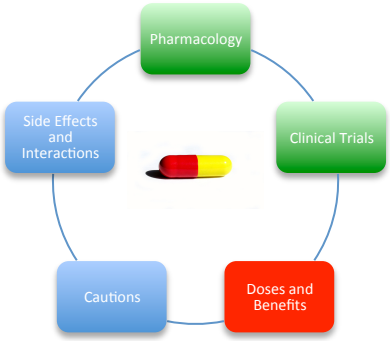
Expectation: TDD improves internal quality

Intuition: TDD should increase modularity and lead to frequent refactoring.

- Mixed results: TDD fared better for certain metrics and worse in others
- High-grade studies: report opposite effects

TDD has no consistent effect

External quality



Type	BETTER	NO-DIFF	WORSE	
Controlled Experiment	✓	✓ ✓ ✓	✓ ✓	6 ***5***
Pilot Study	✓ ✓ ✓ ✓ ✓ ✓	✓ ✓		8 ***7***
Industrial Use	✓ ✓ ✓ ✓ ✓ ✓	✓		7 ***1***
ALL STUDIES:	13	6	2	21
HIGH-GRADE STUDIES:	***5***	***6***	***2***	***13***

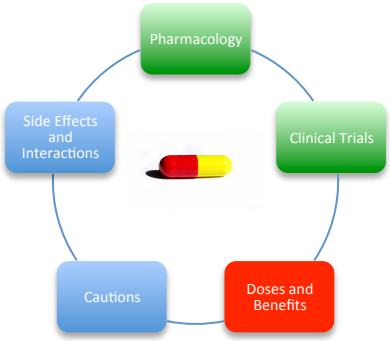
Expectation: TDD improves external quality

Intuition: Developers work on well-specified tasks, use frequent regression testing, and can quickly find errors due to changes.

- Overall, pilots & industrial data support the expectation (Experiments inconclusive)
- BUT evidence is contradictory when less rigorous studies are excluded.

Moderate evidence in favour of TDD

Test support



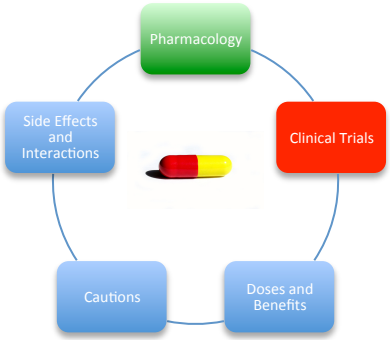
Type	BETTER	NO-DIFF	WORSE	
Controlled Experiment	✓ ✓	✓ ✓ ✓		5 4
Pilot Study	✓ ✓ ✓ ✓ ✓ ✓ ✓	✓	✓	9 6
Industrial Use	✓	✓	✓	3 1
ALL STUDIES:	10	5	2	17
HIGH-GRADE STUDIES:	**5**	**4**	**2**	**11**

Expectation: Testing process should be high quality and easy to maintain.

Intuition: Fine-grained regression tests are part of the process by definition.

- Strong evidence from pilots; experiments show it is no worse than the control.
- Would need to see more industrial evidence...
- Similar impression when low-rigor studies removed.

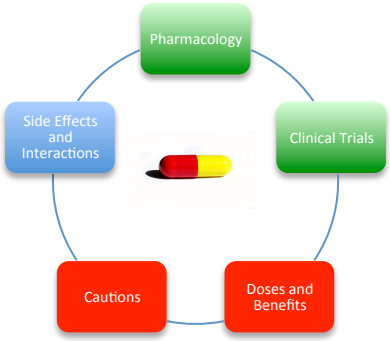
Some evidence that TDD improves test quality.



(OR)

If TDD was a pill,

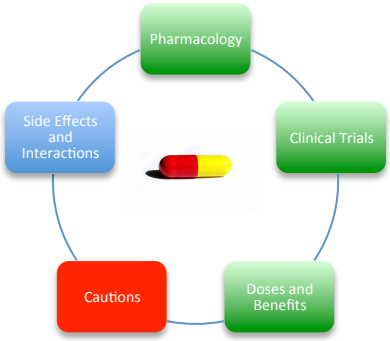
would you take it to improve your health?



Conformance

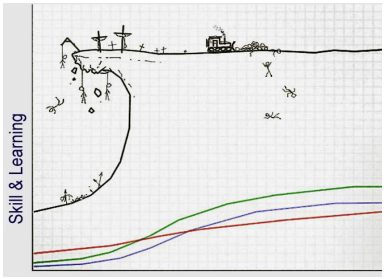
Is that really TDD that you are doing?

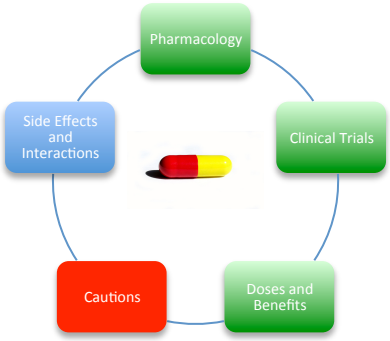
Do you really need all the time?



Domain/ Scope

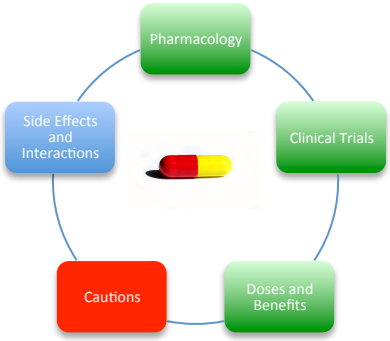
Is it applicable to all tasks?





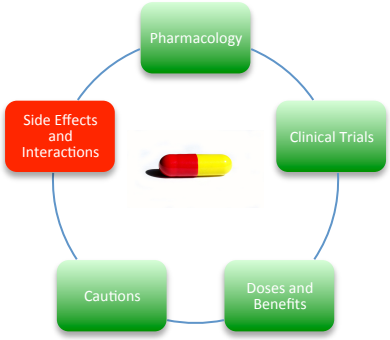
Adoption

Difficult



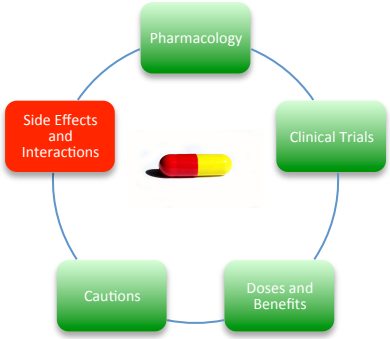
Long term effects

Unknown



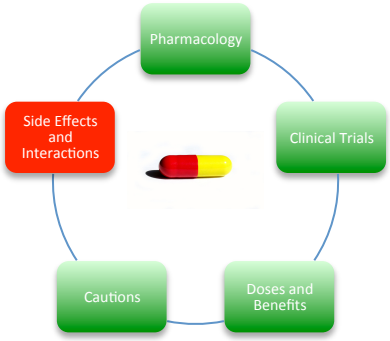
Complement with other practices

Ex: Pair Programming



Avoid with other practices

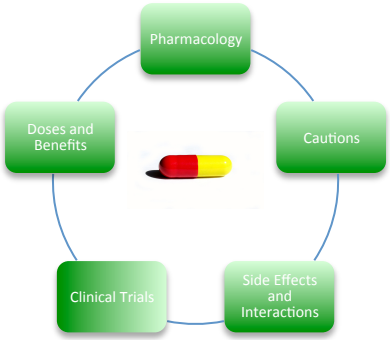
Ex: Big Up-Front Design



Addictive

Changes the way people think:

“It is hard to turn back and start coding from scratch without defining tests! How do you do it?”



If TDD was a pill, would you take it to improve your health?

Agile Project Manager

My thoughts on all things Agile, Project Management and Software Engineering

ABOUT

Categories: Professionalism Project Management Software Engineering

The evidence is in – TDD works!

In the November/December issue of IEEE Software the Voice of Evidence addresses Test Driven Development (TDD). Researcher Forrest Shull and several colleagues have reviewed the research literature on TDD. They posed three questions:

- Does TDD improve delivered quality?
- Does TDD improve internal code quality?
- Does TDD improve productivity?

They found 22 articles describing 33 individual studies. These studies were classified based on their "rigor". The conclusions below are based on *how many* high rigor studies supported it or opposed it. Lower rigor studies and inconclusive evidence is not considered.

TDD does *improve software quality* with 5 favorable studies and 2 negative. There may be some confounding effects of expertise here. In my experience, better programmers are also more likely to embrace TDD. Nevertheless, it is hard for me to see how more testing could lead to worse software.

There is no evidence (2.5 versus 2.5) that TDD leads to better internal code quality. This does not correspond to my personal experience. From what I have seen TDD will force higher modularity, testability and forcing the programmer to think first etc which will all favor better code.

TDD leads to *higher productivity* (6 versus 4). I have previously argued, in "Does unit testing pay off?", that just by reducing the number of bug fixes developers should be able to free up substantial productive hours.

Computing Education Blog

HOME ABOUT COMPUTING EDUCATION BLOG

Making Software: TDD doesn't work; Architecting is mixed; scripting is great

November 1, 2010

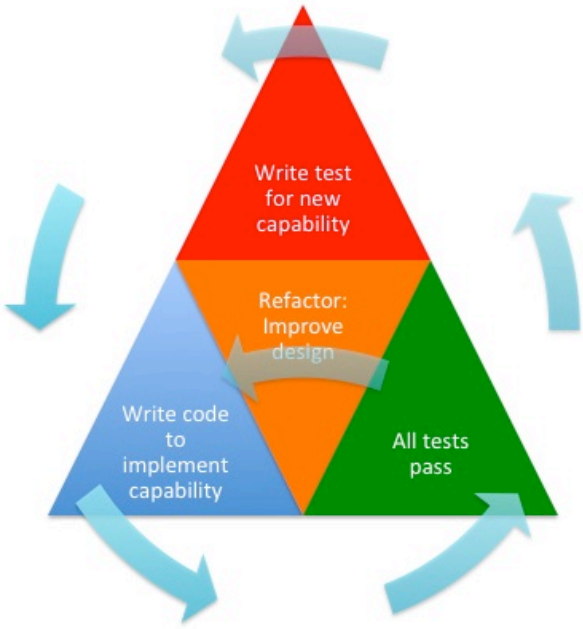
I got my copy of *Making Software* by Oram and Wilson on Friday, and have read just over half of it already. I'm really enjoying it! I've always enjoyed empirical data on programming (Whoa! My teenagers would be rolling their eyes at such a geeky statement!), and this book is chockfull of it. Some of the most interesting chapters for me so-far:

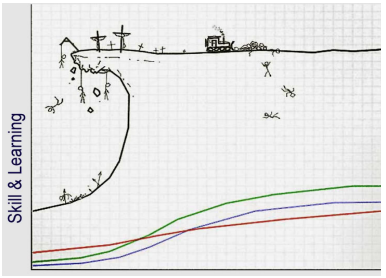
- Herraiz and Hassan have a great study comparing all the various complexity metrics to plain ole counting lines of code. The correlation is *astounding*. Sure, sure — counting lines of code is a rough measure. But when the correlation with McCabe's or Halstead's metrics are 0.97, why go to all that trouble?
- Barry Boehm does an even-handed (even if technically *WAY* complicated) analysis of whether up-front architecting is really worth it, or are agile methods (with architecting distributed across) good enough? The tradeoff is between having to do rework later, and spending way too much expensive effort up front than is worthwhile. His answer is that it depends on the size of the code and the riskiness of the effort. The cost of NOT architecting is between 18% and 92% of the total cost, depending on the project. I think he's trying to make the case that agile methods are too expensive and that Kent Beck's arguments for agile methods are wrong.
- The chapter on "How effective is Test-Driven Development?" was really enlightening for me. There are just no convincing data that building tests before building code is worth it! There aren't measurable benefits in quality, for a variety of measures.

“Would TDD work for me?”

- It depends…
  - Need to consider, what qualities are most important in your context (External quality, testability, ...)
- If intrigued:
  - Try it in small doses, stay alert for good and bad side effects
  - Recognize that there may be an *initial* productivity hit
  - Be open to making *educated* tweaks to the process.
  - Be aware it may be addictive!

m



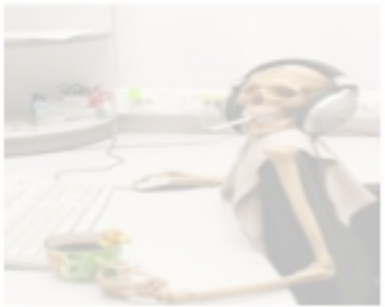


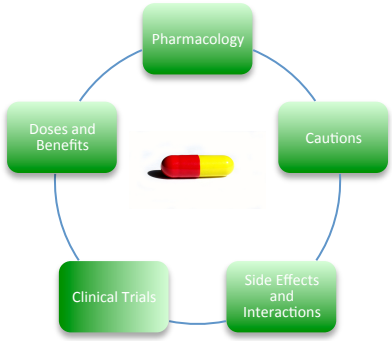




Any Questions?

turhanb@computer.org



Read!

-K. Beck, “Test-driven Development: by Example”, The Addison-Wesley signature series. Addison-Wesley, 2003

-B. Turhan, L. Layman, M. Diep, F. Shull and H. Erdogmus, “How Effective is Test Driven Development?”, in *Making Software: What Really Works, and Why We Believe It*, eds. Greg Wilson, Andy Orham, O'Reilly Press, 2010.

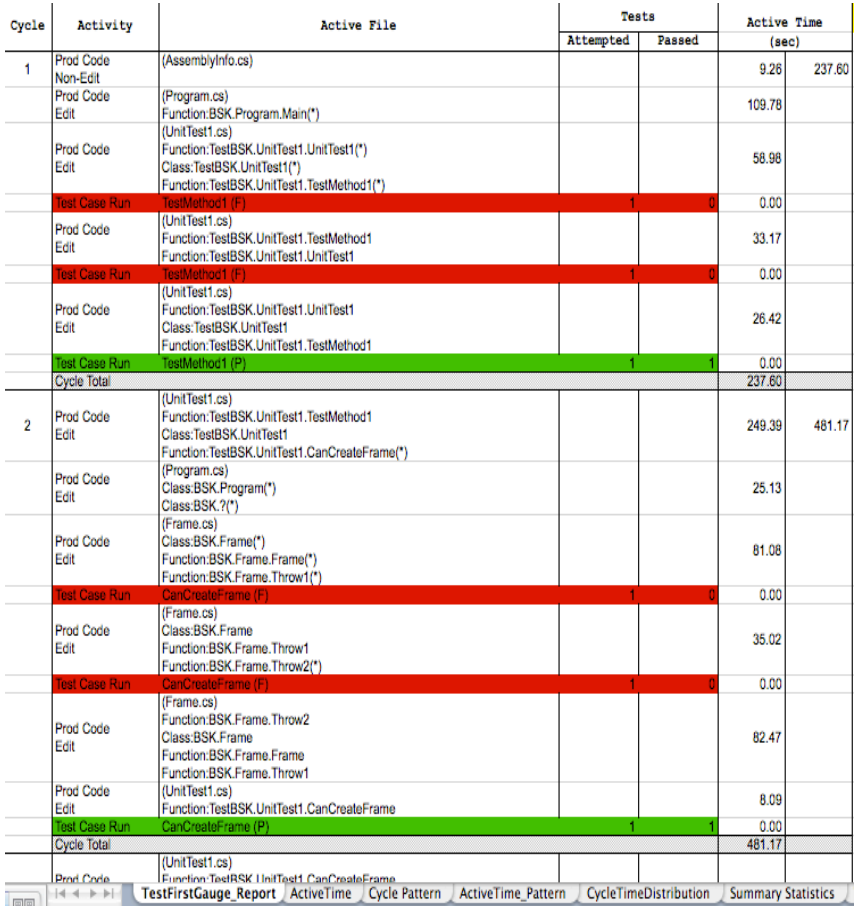
-F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep and H. Erdogmus, “What Do We Know About Test-Driven Development?”, Voice of Evidence in *IEEE Software*, 2010

-Y. Rafique and V. B. Misic, “The effects of test-driven development on external quality and productivity: A meta-analysis”, in *IEEE Transactions on Software Engineering,* 2012

Try!

http://eclipsetutorial.sourceforge.net/totalbeginner.html

TestFirstGauge

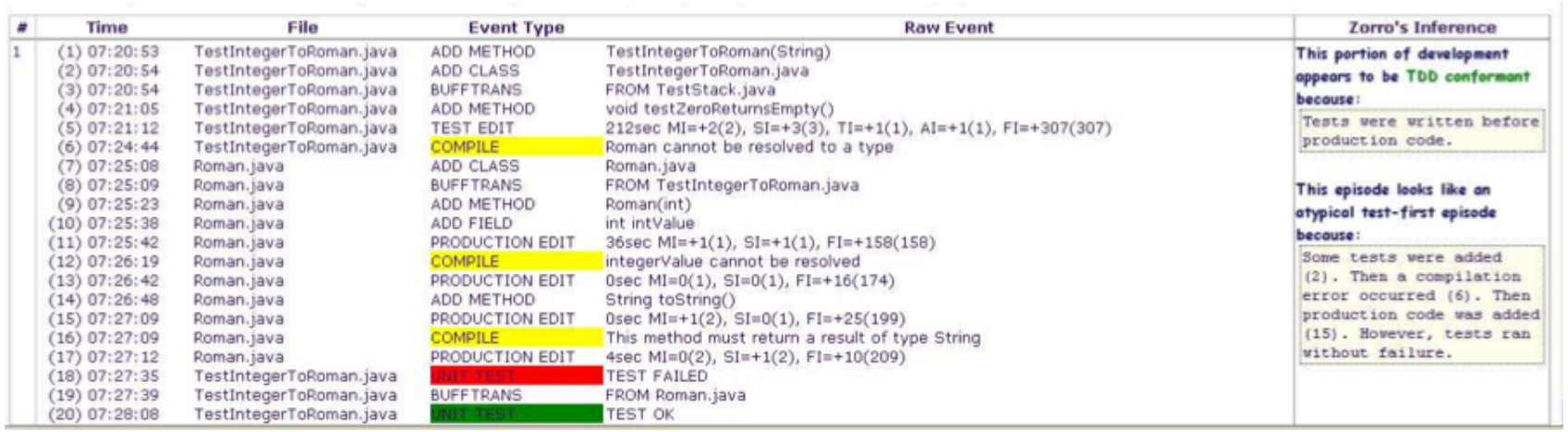


Cycle	Activity	Active File	Tests Attempted	Tests Passed	Active Time (sec)	
1	Prod Code Non-Edit	(AssemblyInfo.cs)			9.26	237.60
	Prod Code Edit	(Program.cs) Function:BSK.Program.Main(*)			109.78	
	Prod Code Edit	(UnitTest1.cs) Function:TestBSK.UnitTest1.UnitTest1(*) Class:TestBSK.UnitTest1(*) Function:TestBSK.UnitTest1.TestMethod1(*)			58.98	
	Test Case Run	TestMethod1 (F)	1	0	0.00	
	Prod Code Edit	(UnitTest1.cs) Function:TestBSK.UnitTest1.TestMethod1 Function:TestBSK.UnitTest1.UnitTest1			33.17	
	Test Case Run	TestMethod1 (F)	1	0	0.00	
	Prod Code Edit	(UnitTest1.cs) Function:TestBSK.UnitTest1.UnitTest1 Class:TestBSK.UnitTest1 Function:TestBSK.UnitTest1.TestMethod1			26.42	
	Test Case Run	TestMethod1 (P)	1	1	0.00	
	Cycle Total				237.60	
2	Prod Code Edit	(UnitTest1.cs) Function:TestBSK.UnitTest1.TestMethod1 Class:TestBSK.UnitTest1 Function:TestBSK.UnitTest1.CanCreateFrame(*)			249.39	481.17
	Prod Code Edit	(Program.cs) Class:BSK.Program(*) Class:BSK.?(*)			25.13	
	Prod Code Edit	(Frame.cs) Class:BSK.Frame(*) Function:BSK.Frame.Frame(*) Function:BSK.Frame.Throw1(*)			81.08	
	Test Case Run	CanCreateFrame (F)	1	0	0.00	
	Prod Code Edit	(Frame.cs) Class:BSK.Frame Function:BSK.Frame.Throw1 Function:BSK.Frame.Throw2(*)			35.02	
	Test Case Run	CanCreateFrame (F)	1	0	0.00	
	Prod Code Edit	(Frame.cs) Function:BSK.Frame.Throw2 Class:BSK.Frame Function:BSK.Frame.Frame Function:BSK.Frame.Throw1			82.47	
	Prod Code Edit	(UnitTest1.cs) Function:TestBSK.UnitTest1.CanCreateFrame			8.09	
	Test Case Run	CanCreateFrame (P)	1	1	0.00	
	Cycle Total				481.17	
	Prod Code	(UnitTest1.cs) Function:TestBSK.UnitTest1.CanCreateFrame				

TestFirstGauge_Report | ActiveTime | Cycle Pattern | ActiveTime_Pattern | CycleTimeDistribution | Summary Statistics

- Sensor data collection from IDE: development activities
- Offline analysis and reporting of the session
- Infrastructure to measure conformance

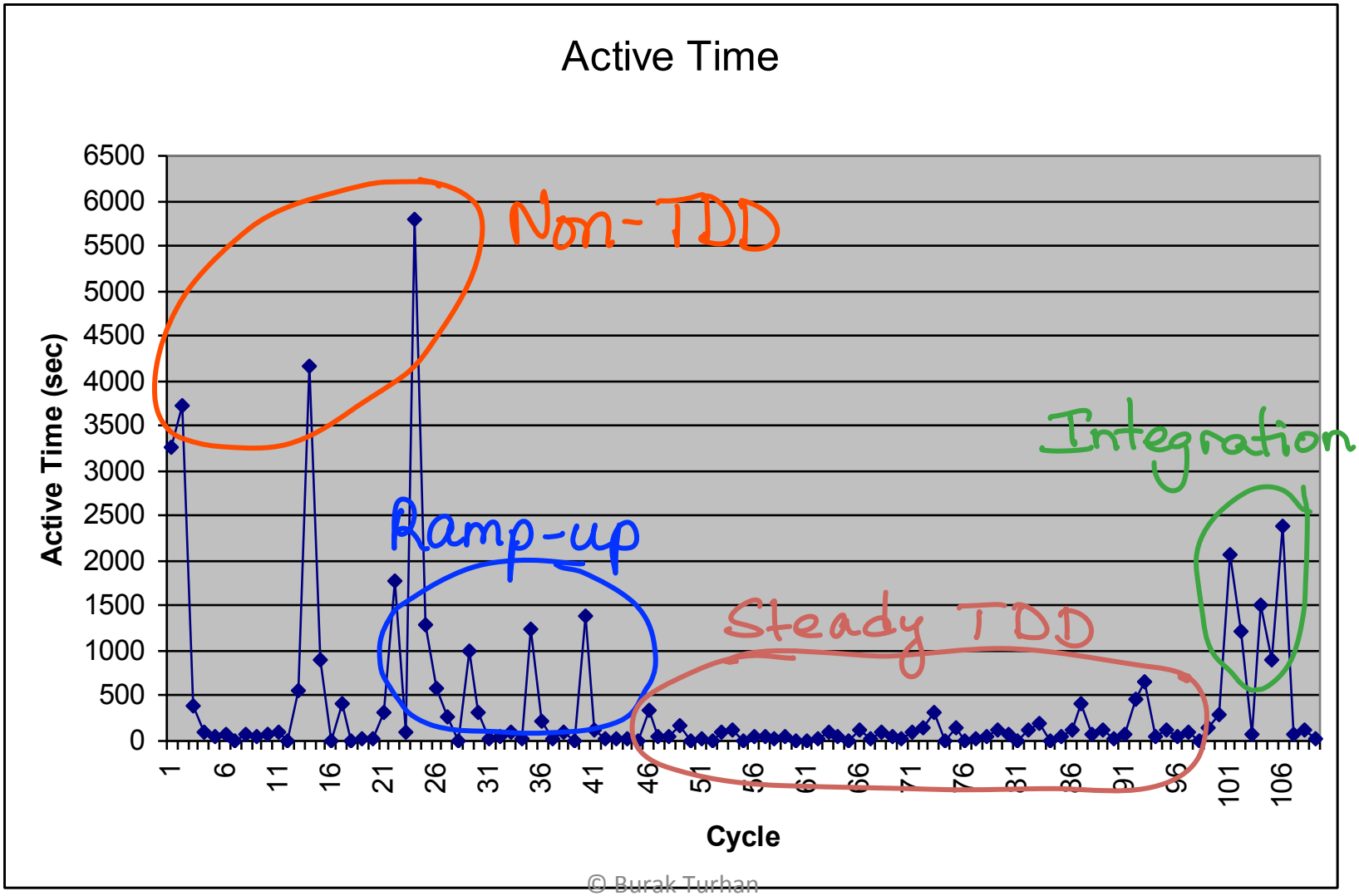
Zorro



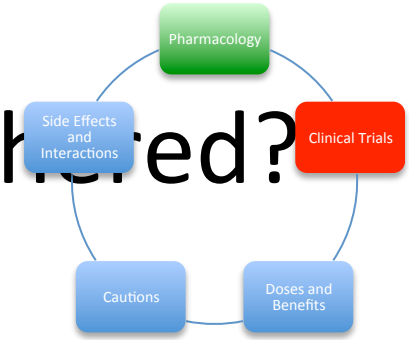
#	Time	File	Event Type	Raw Event	Zorro's Inference
1	(1) 07:20:53	TestIntegerToRoman.java	ADD METHOD	TestIntegerToRoman(String)	This portion of development appears to be TDD conformant because: Tests were written before production code.
	(2) 07:20:54	TestIntegerToRoman.java	ADD CLASS	TestIntegerToRoman.java	
	(3) 07:20:54	TestIntegerToRoman.java	BUFFTRANS	FROM TestStack.java	
	(4) 07:21:05	TestIntegerToRoman.java	ADD METHOD	void testZeroReturnsEmpty()	
	(5) 07:21:12	TestIntegerToRoman.java	TEST EDIT	212sec MI=+2(2), SI=+3(3), TI=+1(1), AI=+1(1), FI=+307(307)	
	(6) 07:24:44	TestIntegerToRoman.java	COMPILE	Roman cannot be resolved to a type	
	(7) 07:25:08	Roman.java	ADD CLASS	Roman.java	This episode looks like an atypical test-first episode because: Some tests were added (2). Then a compilation error occurred (6). Then production code was added (15). However, tests ran without failure.
	(8) 07:25:09	Roman.java	BUFFTRANS	FROM TestIntegerToRoman.java	
	(9) 07:25:23	Roman.java	ADD METHOD	Roman(int)	
	(10) 07:25:38	Roman.java	ADD FIELD	int intValue	
	(11) 07:25:42	Roman.java	PRODUCTION EDIT	36sec MI=+1(1), SI=+1(1), FI=+158(158)	
	(12) 07:26:19	Roman.java	COMPILE	integerValue cannot be resolved	
	(13) 07:26:42	Roman.java	PRODUCTION EDIT	0sec MI=0(1), SI=0(1), FI=+16(174)	
	(14) 07:26:48	Roman.java	ADD METHOD	String toString()	
	(15) 07:27:09	Roman.java	PRODUCTION EDIT	0sec MI=+1(2), SI=0(1), FI=+25(199)	
	(16) 07:27:09	Roman.java	COMPILE	This method must return a result of type String	
	(17) 07:27:12	Roman.java	PRODUCTION EDIT	4sec MI=0(2), SI=+1(2), FI=+10(209)	
	(18) 07:27:35	TestIntegerToRoman.java	UNIT TEST	TEST FAILED	
	(19) 07:27:39	TestIntegerToRoman.java	BUFFTRANS	FROM Roman.java	
	(20) 07:28:08	TestIntegerToRoman.java	UNIT TEST	TEST OK	

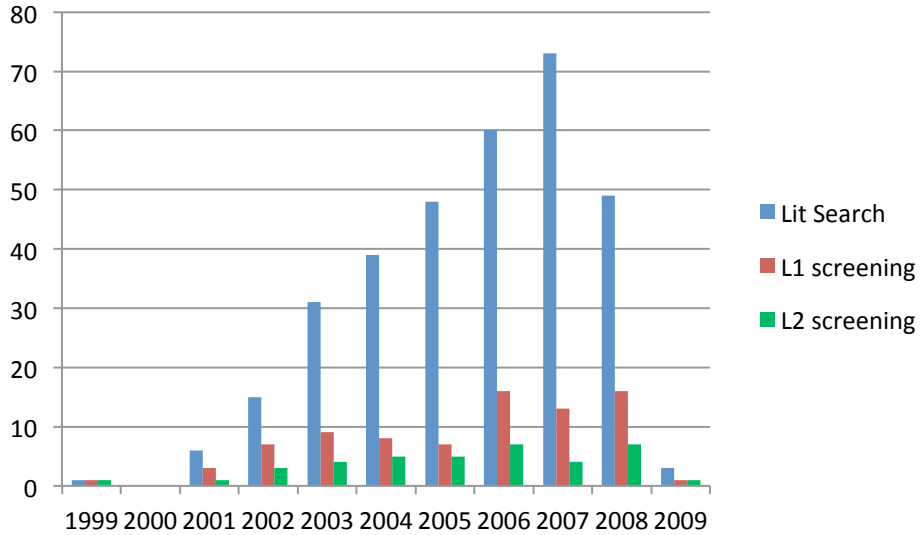
- Based on TFG, uses TFG inference algorithms.
- Online analysis and reporting
- Hackystat plugin developed by Hongbing Kou
- Currently discountinued…

TDD dynamics-1 (©Erdogmus)

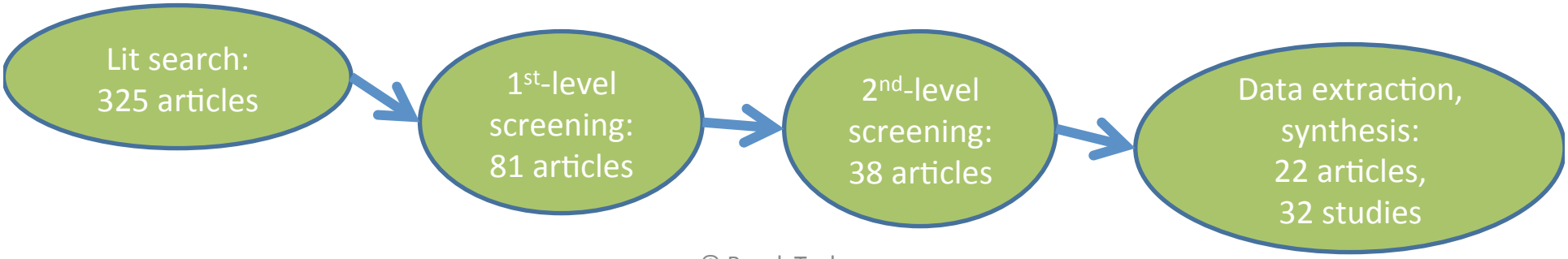


Systematic evidence: How gathered?

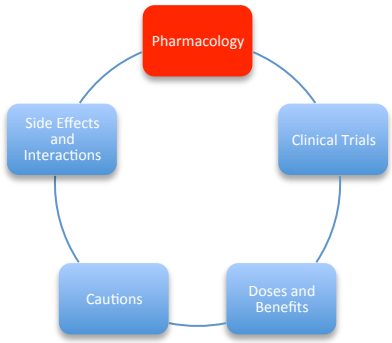




- Systematic literature review
- Exclusive focus on studies that evaluated TDD with respect to productivity, quality, and test support



TDD myths



Myth	Truth
• (Unit) testing = TDD	▸ (Unit) testing ≠ TDD
• It's a QA technique	▸ It's a development approach
• It's a substitute to design & modeling, traditional QA	▸ It's orthogonal to design & modeling, traditional QA
• It's for junior/disorganized people • It's for advanced developers	▸ It requires skill and discipline, but junior developers can still master it
• It curtails productivity	▸ Most who apply it consider it a productivity technique, but the jury is still out

How does TDD work?