CSC 5524 : Software quality, metrics, tests, processes

#### J <u>Paul</u> Gibson, D311

paul.gibson@telecom-sudparis.eu

http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC5524/

# **Software Quality - Tips Techniques & Tools**

http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC5524/ TipsTechniquesTools.pdf

# These are the following TTTs that all quality software engineers know about, and use as a matter of habit:

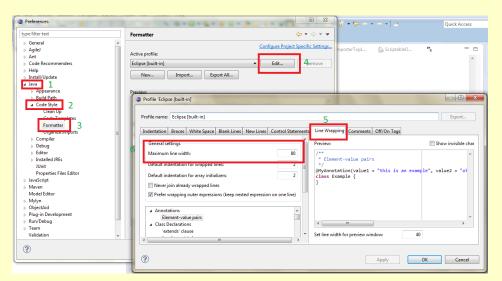
- Code style formatting
- Literate Programming
- Code Documentation Generators
- Code defect detection (bug finding)
- Design defect detection (bad smells), based on OO metrics
- Automated unit testing
- Code Coverage
- Version control
- Automated build: make, ant
- Continuous integration/delivery/testing
- Personal Software Process
- Software Process Improvement (SPI) & Capability Maturity Model (CMM)

#### **Code style formatting: indenting, whitespace**

All modern IDEs include tools for automatically formatting code to a predefined template/style/configuration

Many permit the definition of a library of these templates

TODO: Can you find and use this functionality in Eclipse (for Java)?



**READING** : *The elements of programming style*, Kernighan, Brian W., and Phillip James Plauger, 1979

2018: J Paul Gibson

#### **Literate Programming**

**READING:** Donald Knuth. "Literate Programming" (1984)

"Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do."

Useful Link: http://vasc.ri.cmu.edu/old\_help/Programming/Literate/ literate.html

#### **Code Documentation Generators**

All modern IDEs include tools support for automating documentation generation

Language dependent, e.g. JavaDoc

http://www.oracle.com/technetwork/java/javase/ documentation/index-137483.html

http://doclet.com

Language independent, e.g. doxygen

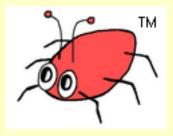
http://www.stack.nl/~dimitri/doxygen/index.html

### **Code defect detection (bug finding)**

Static analysis of code

Example, findbugs

http://findbugs.sourceforge.net



#### Design defect detection (bad smells), based on OO metrics

Example: JDeodorant is an Eclipse plugin that identifies bad designs and proposes fixes (refactorings)



http://users.encs.concordia.ca/~nikolaos/jdeodorant/

#### Automated unit testing

Nearly all modern programming languages have library/tool support for automated unit testing

For Java - JUnit



http://junit.org/junit4/

For C - CUnit

http://cunit.sourceforge.net

### **Code coverage**

When running tests, you need to know what percentage of code executed during the test runs.

Most modern IDEs provide coverage tools/plugins

For Eclipse, with Java, a good example is EclEmma

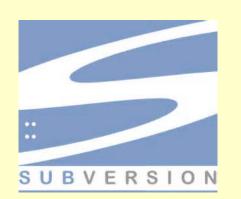
http://eclemma.org

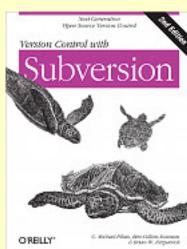
Name 22 3 4 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	≝∎G•]≜•§••∳•••]⊠.	ar 1 🥭 🖼		ti 😻 🏷	
Trained after 34,090 seconds	2 9 public boolean addAll int in		(		
	if [c.isEmpty]]] (	nder, collect	ion cl (	-	
Runsi 13009/13009 DEmors: D Defailures: 0	return false:				
	<pre>} else if   size == index</pre>	. II. aire an	01. (		
	return add&11[c];	11 D100	01.0		
Pakzes Catherardry	) else (				
- In junit framework. Test Sube	Listable succ = getl	Listableätlin	dex La		
junit.framework.TestSuite	Listable pred = [mal			mere i La	
TestBagLtils	Iterator it = c.iter				
B - in org. apache.commons.collections.TestClos	while [it.basNext]]				
	Image: A second se				
⊞ TestBufferUtils					
∃ TestEnumerationUtils	return true;				
B up apache commons collections. Testfact					
Testi.kti.kti     TestiMapi.tik					
Image of the second secon					
TestSetLitik			_	•	
	Problems Javadoc Dedaration Console Deverage	11			
org. apache.commons.collections.TestTrar     TestArrayStack					
H orp. apache .commons .collections. TestTrar     H orp. apache .commons .collections. TestTrar     H orp. TestBranNap	TestAlPackages (31.10.2006 15:04:14)	0 <sub>6</sub>   36 %	a 🗈 🗟 - 🔯		
⊞ ≣ TestArrayStadk ⊞ ≣ TestBeanMap	TestAlPackages (31.10.2006 15:D4:14) Element	Q 26 3 Coverage	CoveredLines	TotalLines	
⊞- I TestArrayStack	TestAlPachages (31,10,2006 15:04:14) Element 	Coverage 70,5 %	Covered Lines	Total Lines 13738	
TestArrayStack     TestBeanNap     Grg.apache.commons.collections.TestBina-	TestAlPackages (31.10.2006 15:04:14) Element G S java - commons-collections E) B org. spache science scientions	0, 36 3 Coverage 79,5 % 74,1 %	Covered Lines 10927 3842	TotalLines 13738 5183	
In TestArrayStad:     InstArrayStad:     InstBanMap     InstBanMap     InstBanMap     InstBandedTivBuffer	TestAlPackages (31.10.2006 15:04:14) Element 	0, 36 % Coverage 79,5 % 74,1 % 80,5 %	Covered Lines 10927 3842 32	TotalLines 13738 5183 37	
Trestmarking     T	TestAlPackages (31.10.2005 15:04:14) Benet:	0, 10 % Coverage 79,5 % 74,1 % 86,5 % 86,7 %	Covered Lines 10927 3842 32 13	TotalLines 13738 5183 37 15	
Fig. Technics Stack     Fig. Technics Stack     Fig. Technics Stack     Fig. Technics Collections Technics     Fig. Technics Collections     Fig. Technics And Provide Technics     Fig. TechCustor adds Index Air      Fig. TechCustor adds Index Air      Fig. TechCustor Collected Nap	TestAllhackages (31.10.2005 (5:04:14) Beneat () (2) yon- commons-collectors () (2) ArroyStackyene () (2) ArroyStackyene () (2) Bagtific jewa () (2) Bagtific jewa () (2) Bagtific jewa	Op         M         Op           Coverage         79,5 %         74,1 %           86,5 %         86,7 %         86,7 %           72,4 %         72,4 %         72,4 %	Covered Lines 10927 3842 32 13 155	TotalLines 13738 5183 37 15 214	
H = Trektras Stads     H = Trektras     H = Trektras Stads     H = Trektras     H = Trektras	TestAlPackages (21.10.2005 35:04:14) Benefit ■ ■ ■ aparties connects collectors ■ ■ Transition (pro- 0.1) Transition (pro- 0.1) Dearting (pro- 0.1) D	Op         M         M           Coverage         79,5 %           79,5 %         74,1 %           86,5 %         85,7 %           72,4 %         87,6 %	Covered Lines 10927 3842 32 13 155 127	TotalLines 13736 5183 37 15 214 145	
B T Technolitak     Techn	TestAll'schages (31.10.2005 15:04:14) Brevet	0, 10 0 Coverage 79,5 % 74,1 % 80,5 % 80,7 % 72,4 % 87,6 % 93,2 %	Covered Lines 10927 3842 32 13 155 127 82	TotalLines 13736 5183 37 15 214 145 88	
B T Technologian     Technologian	TestRifflexRages (21.10.2005 15:04:14) Bienet	0, 36 3 Coverage 79,5 % 70,1 % 80,5 % 72,4 % 87,6 % 93,2 % 93,2 % 93,2 %	CoveredLines 10927 3842 32 13 158 127 82 82 5	TotalLines 13738 5183 37 15 214 145 58 9	
Comparison of the second	TestABlackages (31:10:2005 15:04:14)           Brenet           ● ● Prove common-collectors	0, 36 9 Coverage 79,5 % 74,1 % 86,5 % 86,5 % 72,4 % 87,6 % 93,2 % 93,2 % 83,9 %	Covered Lines 10927 3842 32 13 155 127 82 5 8	TotalLines 13738 5183 37 15 214 145 88 9 9	
B      B      TestArmoltak     Signal TestArmoltak     Signal Second a     Signal Second a     Signal Second and a Second and American Second	TestRifflexRages (21.10.2005 15:04:14) Bienext	05 38 3 Coverage 79,5 % 79,5 % 79,5 % 70,5 % 80,5 % 80,7 % 72,4 % 93,2 % 93,2 % 93,2 % 93,9 % 93,9 %	Covered Lines 10927 3842 32 13 155 127 82 5 5 6 4	TotalLines 13738 5183 37 15 214 145 88 9 9 9 9	
Comparison of the second	TentHirachappe (31.0.2000 35:04:14) Bannet	0, 28 % Coverage 79,5 % 74,1 % 86,5 % 86,7 % 72,4 % 87,4 % 93,2 % 83,9 % 83,9 % 93,6 % 93,2 %	Covered Lines 10907 3842 32 13 155 127 82 5 8 4 31	TotalLines 13736 5183 37 15 214 145 88 9 9 9 13 33	
B      B      TestArmoltak     Signal TestArmoltak     Signal Second a     Signal Second a     Signal Second and a Second and American Second	TestRifflexRages (21.10.2005 15:04:14) Bienext	05 38 3 Coverage 79,5 % 79,5 % 79,5 % 70,5 % 80,5 % 80,7 % 72,4 % 93,2 % 93,2 % 93,2 % 93,9 % 93,9 %	Covered Lines 10927 3842 32 13 155 127 82 5 5 6 4	TotalLines 133738 5183 37 15 214 145 88 9 9 9 9 13	

# **Version control**

You should know the basics of

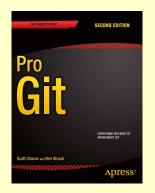
Subversion -svn https://subversion.apache.org



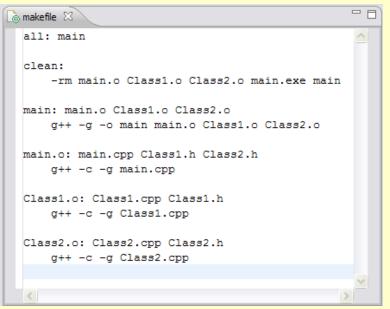


Git https://git-scm.com/





### Automated build: make, ant



http://www.gnu.org/software/make/manual/make.html



### **Continuous integration/delivery/testing**

- Maven https://maven.apache.org
  - http://www.sonarqube.org
- Jenkins https://jenkins.io

Sonar

Hudson http://hudson-ci.org

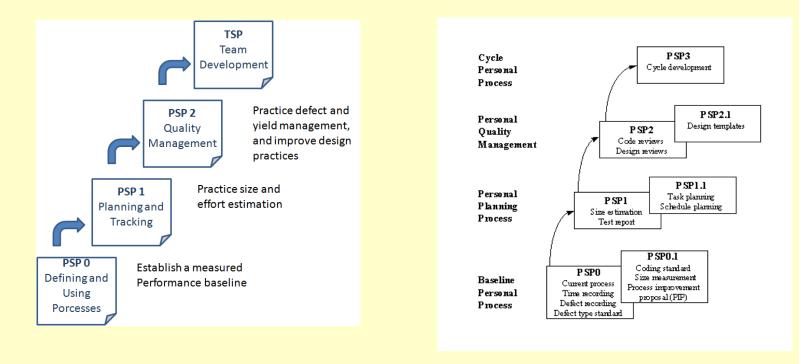






Make build easy Make build a uniform process Provide quality information feedback Support/enforce best practice for test/delivery Wrap up all this on an intreated service/server

# **Personal Software Process**

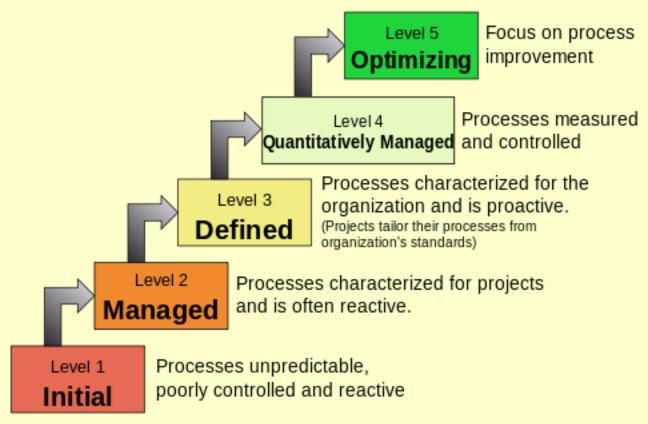


Using a defined and measured Personal Software Process, Watts S. Humphrey, 1996

The Personal Software Process: A Cautionary Case Study, Philip M. Johnson, Anne M. Disney, 1998

#### Software Process Improvement (SPI) & Capability Maturity Model (CMM)

#### **Characteristics of the Maturity levels**



#### http://www.sei.cmu.edu/cmmi/