Empirical Software Engineering with Examples

is not a topic for examination

Empirical Software Engineering

a sub-domain of software engineering focusing on experiments on software systems

devise experiments on software, in collecting data from these experiments, and elaborate laws and theories from this data (wikipedia)

Software Quality

?

Defects (Bugs)

Classification

		Observed Defects: TRUE	Observed Defects: FALSE
Predicted Defects	TRUE	TP	FP
	FALSE	FN	TN

precision = TP/(TP + FP)

recall = TP/(TP + FN)

Some tests...

arithmetic mean

mean(2, 3, 4) = 3

mean(0, 3, 6) = 3

standard deviation

R

http://www.r-project.org/

T. Zimmermann, R. Premraj, and A. Zeller

Predicting defects for Eclipse

Third International Workshop on Predictor Models in Software Engineering (PROMISE)

IEEE Computer Society, 2007

Predicting Defects for Eclipse

Release	Number of	
	Files	Packages
2	6729	377
2,1	7888	434
3	10593	661

Used Metrics...

methods	FOUT	Number of method calls (FAN OUT)
	MLOC	Method Lines of Code
	NBD	Nested block depth
	PAR	Number of parameters
	VG	McCabe Cyclomatic Complexity
classes	NOF	Number of fields
	NOM	Number of methods
	NSF	Number of static fields
	NSM	Number of static methods
files	ACD	Number of anonymous type declarations
	NOI	Number of interfaces
	NOT	Number of classes
	TLOC	Total lines of code
packages	NOCU	Number of files

Spearman correlations

FOUT	0.400	0.319	0.537	0.523
MLOC	0.403	0.322	0.545	0.544
NBD	0.392	0.320	0.392	0.320
PAR	0.350	0.283	0.554	0.526
VG	0.389	0.315	0.546	0.538
NOF	0.260	0.204	0.507	0.480
NOM	0.319	0.268	0.502	0.491
NSF	0.186	0.170	0.459	0.414
NSM	0.202	0.179	0.448	0.371
ACD	0.258	0.180	0.442	0.414
NOI	0.160(-)	0.129(-)	0.129	0.110
NOT	0.160	0.129	0.518	0.470
TLOC	0.421	0.333	0.581	0.559
NOCU	-	-	0,514	0,461

Classification of files

Release	Testing	Precision	Recall
2	2.0	0.692	0.265
	2.1	0.478	0.191
	3.0	0.613	0.171
2,1	2.0	0.664	0.203
	2.1	0.668	0.160
	3.0	0.717	0.139
3	2.0	0.578	0.277
	2.1	0.528	0.220
	3.0	0.675	0.224

Foutse Khomh, Massimiliano Di Penta, Yann-Gaël Guéhéneuc, Giuliano Antoniol

An exploratory study of the impact of antipatterns on class change- and fault-proneness.

Empirical Software Engineering 17(3): 243-275 (2012)

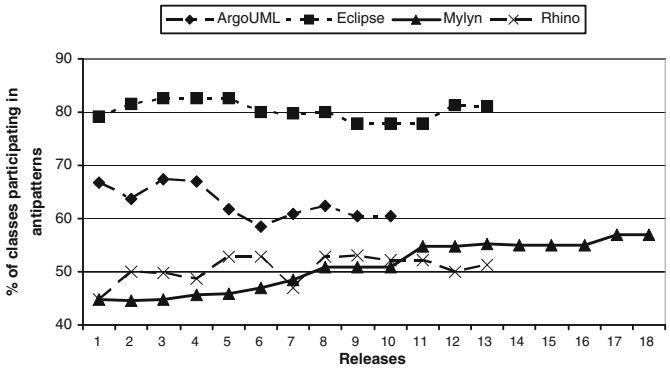


Fig. 1 Percentages of classes participating in antipatterns in the releases of the four systems

AntiSingleton

List of AntiPatterns...

ClassDataShouldBePrivate

SpeculativeGenerality

MessageChain

LongMethod

RefusedParentBequest

Blob

LazyClass

ComplexClass

LongParameterList

SwissArmyKnife

SpaghettiCode

LargeClass

Research Questions...

What is the relation between

anti-patterns

change- & defect- proneness

Research Questions...

What is the relation between

kinds

change- & defect- proneness

Research Questions...

Does the presence

relate to the size of these classes

Research Questions...

What kind of changes

participating or not in anti-patterns

Change proneness

ArgoUML		Eclipse		Mylyn		Rhino	
Releases	Odds ratios	Releases	Odds ratios	Releases	Odds ratios	Releases	Odds ratios
0.10.1	4.17	1.0	1.13	1.0.1	10.51	1.4R3	10.41
0.12	7.16	2.0	0.75	2.0M1	10.37	1.5R1	17.98
0.14	6.22	2.1.1	2.59	2.0M2	7.38	1.5R2	17.37
0.16	15.84	2.1.2	1.42	2.0M3	206.60	1.5R3	15.71
0.18.1	10.00	2.1.3	1.15	2.0	14.17	1.5R4	16.19
0.20	26.54	3.0	0.88	2.1	10.89	1.5R41	30.71
0.22	8.83	3.0.1	0.86	2.2.0	11.10	1.5R5	15.51
0.24	15.40	3.0.2	0.89	2.3.0	9.83	1.6R1	24.73
0.26	3.98	3.2	2.19	2.3.1	7.66	1.6R2	12.69
0.26.2	6.75	3.2.1	1.94	2.3.2	24.38	1.6R3	19.95
		3.2.2	1.47	3.0.0	9.45	1.6R4	33.05
		3.3	2.43	3.0.1	9.85	1.6R5	19.97
		3.3.1	1.42	3.0.2	5.31	1.6R6	20.56
				3.0.3	8.18		
				3.0.4	3.77		
				3.0.5	4.96		
				3.1.0	10.53		
				3.1.1	5.59		

Fault proneness

ArgoUML		Eclipse		Mylyn		Rhino	
Releases	Odds ratios	Releases	Odds ratios	Releases	Odds ratios	Releases	Odds ratios
0.10.1	4.43	1.0	1.14	1.0.1	10.45	1.4R3	6.44
0.12	4.87	2.0	2.06	2.0M1	17.70	1.5R1	31.29
0.14	17.53	2.1.1	2.19	2.0M2	≫300	1.5R2	–
0.16	6.58	2.1.2	2.27	2.0M3	–	1.5R3	13.93
0.18.1	5.33	2.1.3	2.75	2.0	–	1.5R4	9.06
0.20	4.95	3.0	3.30	2.1	–	1.5R41	30.05
0.22	9.42	3.0.1	2.12	2.2.0	–	1.5R5	10.57
0.24	2.25	3.0.2	1.75	2.3.0	–	1.6R1	29.26
0.26	8.08	3.2	3.55	2.3.1	–	1.6R2	–
0.26.2	9.73	3.2.1	2.54	2.3.2	–	1.6R3	–
		3.2.2	2.41	3.0.0	–	1.6R4	23.00
		3.3	2.90	3.0.1	–	1.6R5	13.29
		3.3.1	1.17	3.0.2	–	1.6R6	–
				3.0.3	–		
				3.0.4	–		
				3.0.5	–		
				3.1.0	–		
				3.1.1	–		

Antipatterns	Change proneness			
	ArgoUML	Eclipse	Mylyn	Rhino
AntiSingleton	8 (80%)	5 (38%)	7 (39%)	–
Blob	2 (20%)	8 (62%)	9 (50%)	–
CDSBP	3 (30%)	7 (54%)	9 (50%)	6 (46%)
ComplexClass	2 (20%)	12 (92%)	2 (11%)	–
LargeClass	2 (20%)	–	4 (22%)	4 (31%)
LazyClass	5 (50%)	12 (92%)	3 (17%)	1 (8%)
LongMethod	10 (100%)	12 (92%)	17 (94%)	5 (38%)
LPL	9 (90%)	10 (77%)	7 (39%)	3 (23%)
MessageChain	10 (100%)	12 (92%)	18 (100%)	13 (100%)
RPB	9 (90%)	6 (46%)	10 (56%)	5 (38%)
SpaghettiCode	–	–	–	–
SG	–	3 (23%)	6 (33%)	1 (8%)
SwissArmyKnife	–	6 (46%)	–	–

Antipatterns	Fault proneness			
	ArgoUML	Eclipse	Mylyn	Rhino
AntiSingleton	5 (50%)	11 (84%)	–	–
Blob	1 (10%)	6 (46%)	–	–
CDSBP	2 (20%)	7 (54%)	2 (66%)	3 (33%)
ComplexClass	–	13 (100%)	1 (33%)	–
LargeClass	3 (30%)	–	–	3 (33%)
LazyClass	–	12 (92%)	–	2 (22%)
LongMethod	1 (10%)	13 (100%)	–	3 (33%)
LPL	5 (50%)	7 (54%)	2 (66%)	3 (33%)
MessageChain	7 (70%)	10 (77%)	1 (33%)	7 (78%)
RPB	4 (40%)	3 (23%)	1 (33%)	–
SpaghettiCode	–	–	–	–
SG	–	3 (23%)	–	1 (11%)
SwissArmyKnife	–	3 (23%)	–	–

Are the clients of flawed classes (also) defect prone?

Authors: Radu & Cristina Marinescu

IEEE International Working Conference on Source Code Analysis and Manipulation, 2011

Clients of Flawed Classes
vs.
Defects

Does any correlation exist in Eclipse ?

Class

Data Class

Feature Envy

God Class

Brain Class

has
design flaws

has provider classes
with design flaws

?

Class

has defects
(pre and post release)

Chi-Square

Odds Ratio

Mann-Whitney

Non-parametric statistical tests

flaws / flawed providers

vs.

defects

flawed **providers** correlates with defects!

Odds Ratio on Defects	HAS Design Flaws	NO Design Flaws
HAS Flawed Providers		
NO Flawed Providers		-

Odds Ratio on Defects	HAS Design Flaws	NO Design Flaws
HAS Flawed Providers		
NO Flawed Providers	0.5 - 0.7	-

Odds Ratio on Defects	HAS Design Flaws	NO Design Flaws
HAS Flawed Providers		2 - 3
NO Flawed Providers	0.5 - 0.7	-

Odds Ratio on Defects	HAS Design Flaws	NO Design Flaws
HAS Flawed Providers	4 - 8	2 - 3
NO Flawed Providers	0.5 - 0.7	-

Odds Ratio on Defects	HAS Design Flaws	NO Design Flaws
HAS Flawed Providers	4 - 8	2 - 3
NO Flawed Providers	0.5 - 0.7	-

The biggest danger is **using** flawed classes!

Percent of classes with...	Design Flaws	NO Design Flaws
Flawed Providers		
NO Flawed Providers		

Percent of classes with...	Design Flaws	NO Design Flaws
Flawed Providers	12	
NO Flawed Providers	7	

Percent of classes with...	Design Flaws	NO Design Flaws
Flawed Providers	12	42
NO Flawed Providers	7	40

Are the Classes that use Exceptions Defect Prone?

Author: Cristina Marinescu

12th International Workshop on Principles on Software Evolution
7th ERCIM Workshop on Software Evolution
2011

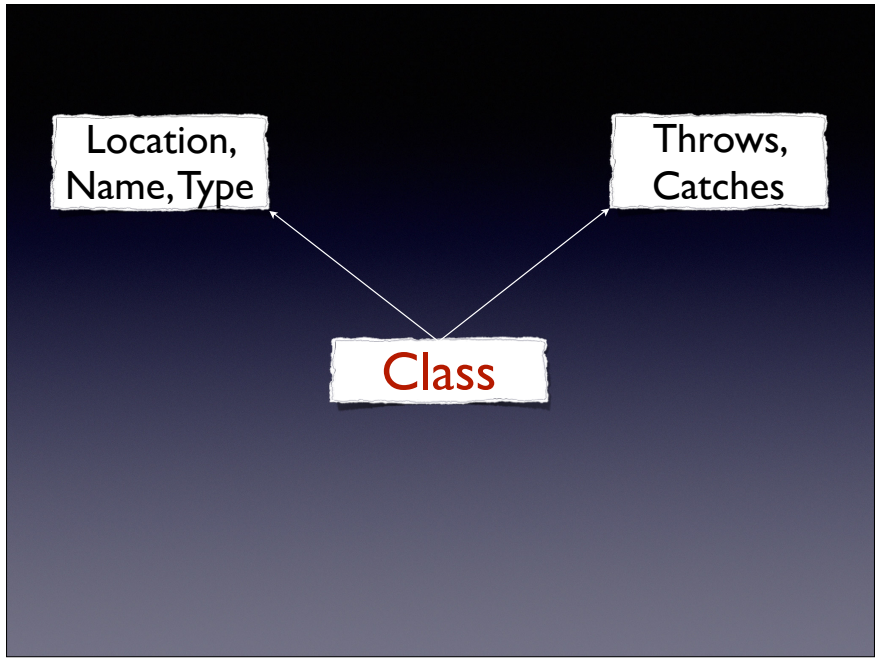
Exceptions
&
Defects

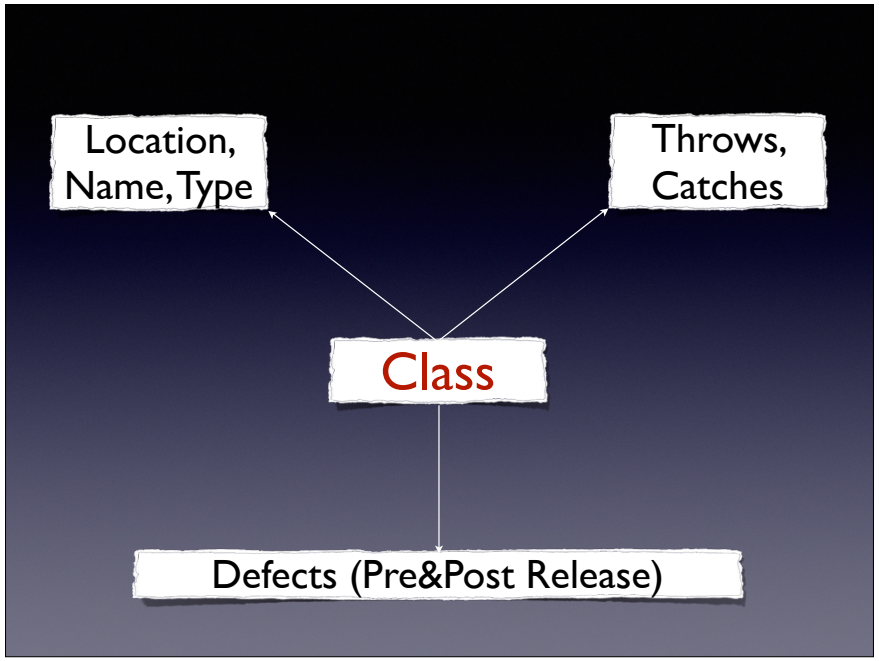
Does any correlation exist in Eclipse ?

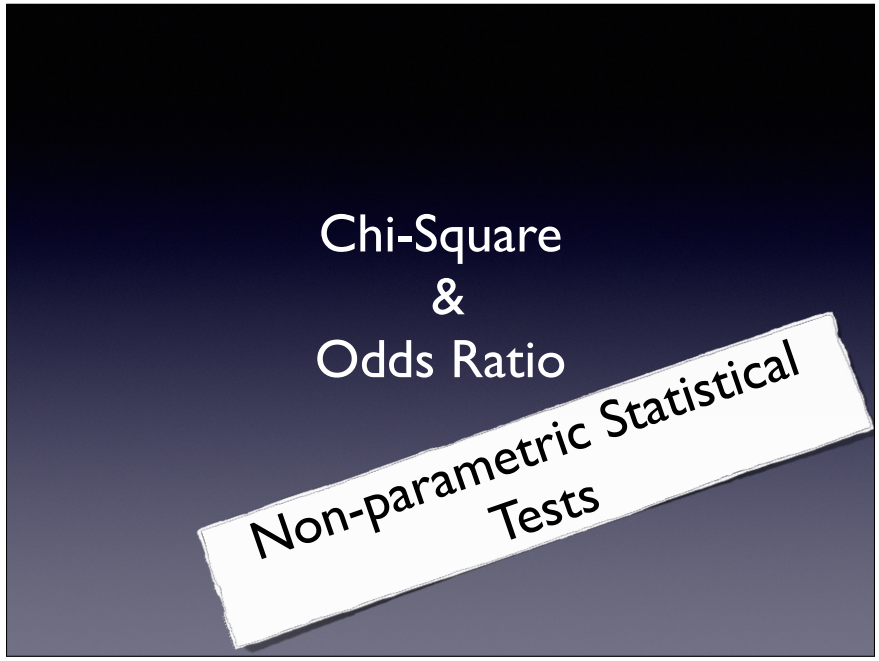
Class

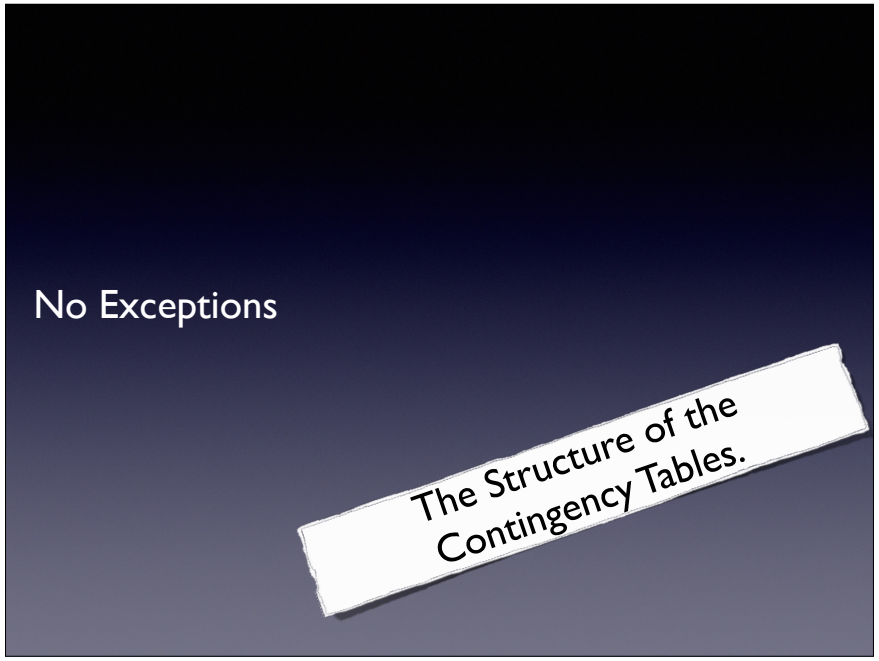
Location,
Name, Type

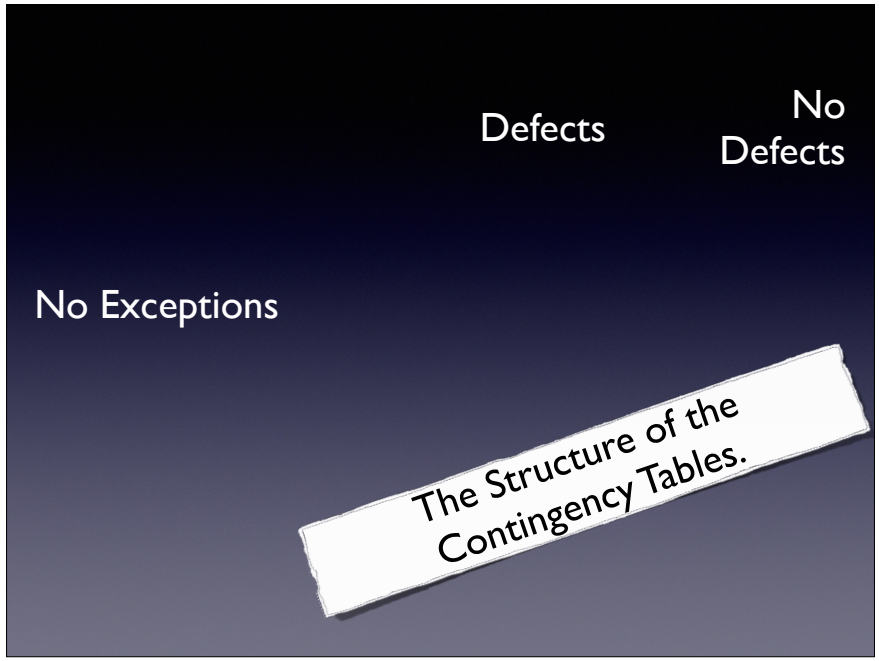
Class

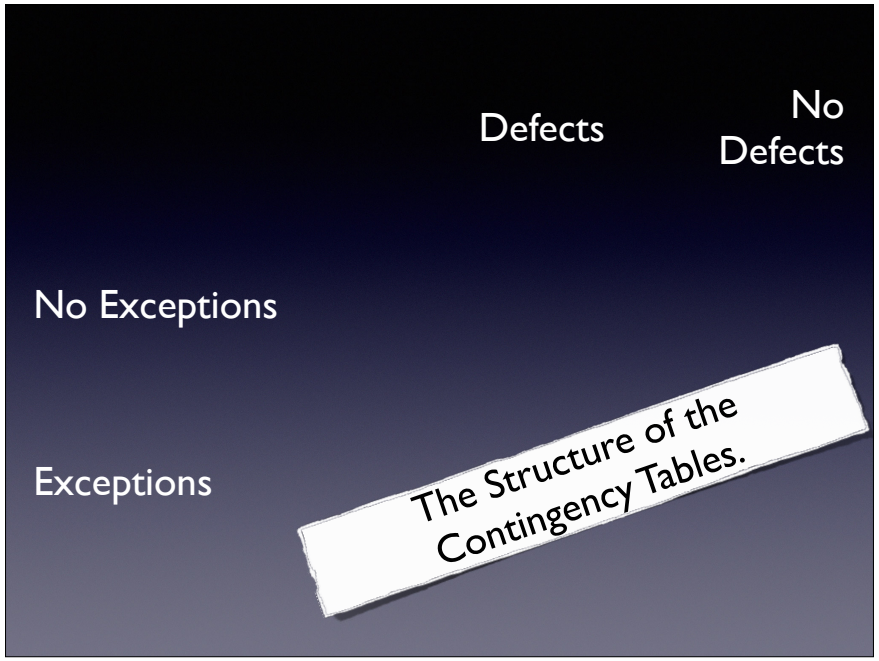

Location,
Name, Type
Throws,
Catches
Class

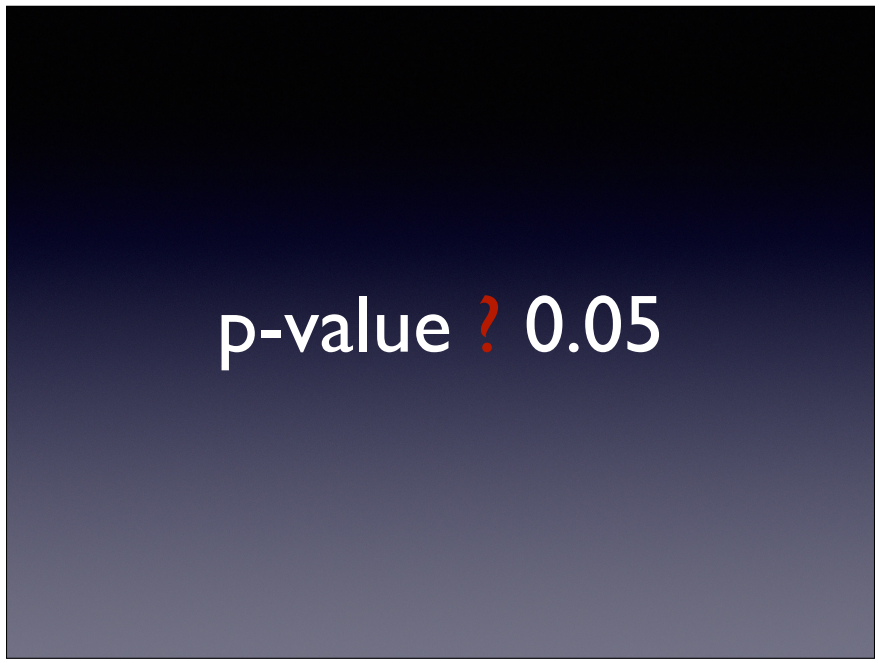

Location,
Name, Type
Throws,
Catches
Class
Defects (Pre&Post Release)

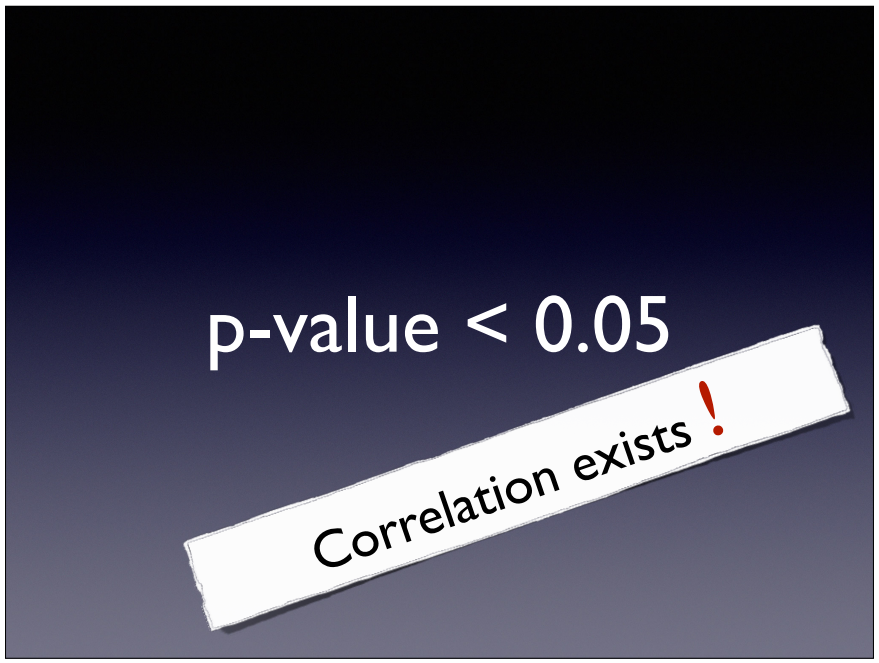

Chi-Square
&
Odds Ratio
Non-parametric Statistical Tests


No Exceptions
The Structure of the Contingency Tables.


Defects
No Defects
No Exceptions
The Structure of the Contingency Tables.


Defects
No Defects
No Exceptions
Exceptions
The Structure of the Contingency Tables.


p-value ? 0.05


p-value < 0.05
Correlation exists !

	Defects	No Defects
No Exceptions	<	>
Exceptions	>	<

How Much ?

	Defects	No Defects
No Exceptions	<	>
Exceptions	>	<

	Eclipse		
	2	2,1	3
Pre-Release Defects	2,4	2,03	1,98
Post-Release Defects	2,98	2,45	1,69

?

Correlation exists !

Improper Usages of Exceptions
&
Defects

Does any correlation exist ?

Improper Usages
=
Exceptions handled by
Superclasses

	Defects	No Defects
No Improper Usages		
Improper Usages		

The Structure of the NEW Contingency Tables.

	Eclipse		
	2	2,1	3
Pre-Release Defects	2,18	2,55	2,15
Post-Release Defects	2,94	2,33	1,47

Improper Usages
&
Defects

Correlation exists !

Exceptions
&
Defects

Improper Usages
&
Defects

Correlation exists !

Nachiappan Nagappan, Brendan Murphy, Victor R. Basili

The influence of organizational structure on software quality: an empirical case study.

International Conference of Software Engineering, 2008

Conway's Law

Conway's Law

organizations that design systems are constrained to produce systems which are copies of the communication structure of these organizations.

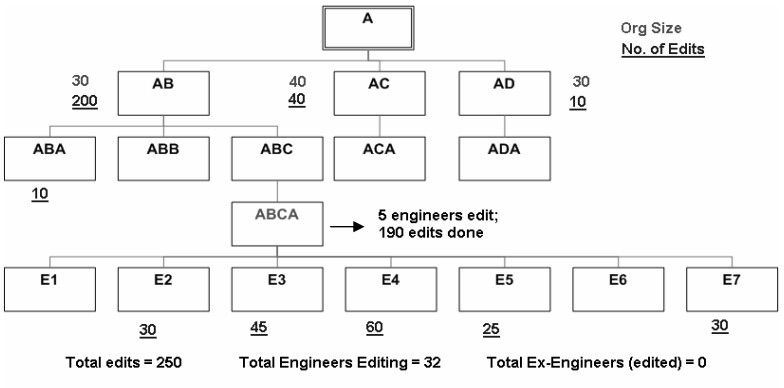


Figure 1: Example Organization Structure of Company "XYZ"

Number of Engineers (NOE)

N engineers

(N*(N-1))/2 communication paths

Number of Ex-Engineers (NOEE)

knowledge transfer

Edit Frequency (EF)

total number times the source code was edited

Depth of Master Ownership (DMO)

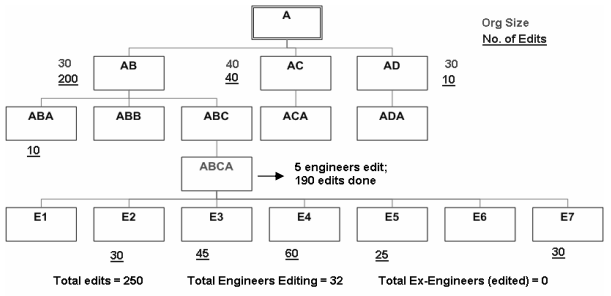


Figure 1: Example Organization Structure of Company "XYZ"

level of ownership (> 75%)

ABCA (190/250)

Percentage of Org Contributing to development (PO)

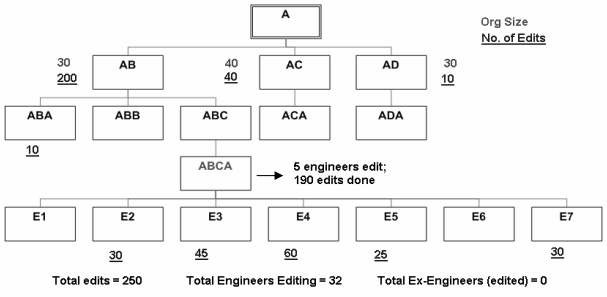


Figure 1: Example Organization Structure of Company "XYZ"

ratio of the number of people reporting at the DMO level owner relative to the Master org size

7/30

Level of Organizational Code Ownership (OCO)

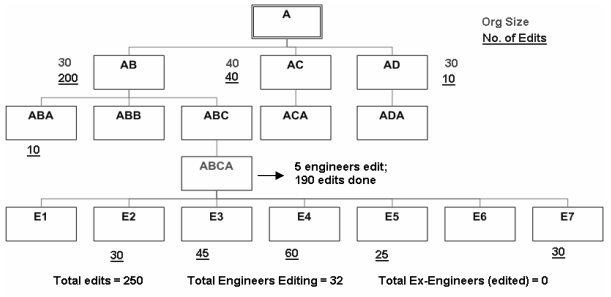


Figure 1: Example Organization Structure of Company "XYZ"

percents of edits from the org that contains the owner

200/250

Overall Organizational Ownership(OOW)

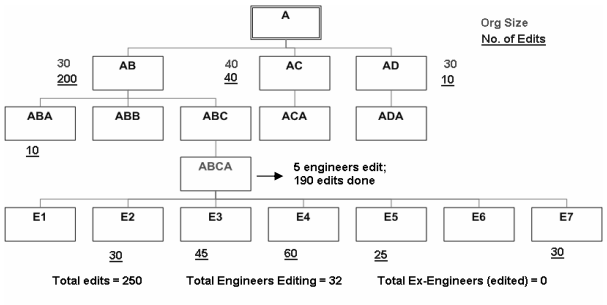


Figure 1: Example Organization Structure of Company "XYZ"

ratio of the percentage of people at the DMO level making edits relative to the total engineers editing

5/32

Organizational Intersection Factor(OIF)

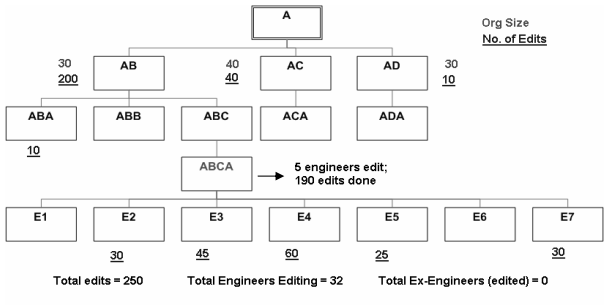


Figure 1: Example Organization Structure of Company "XYZ"

number of different org that that contribute greater than 10% of edits

lower value = better

Prediction based on Organizational Structure

Precision = 86.2%

Recall = 84.0%

Ralph Peters, Andy Zaidman

Evaluating the Lifespan of Code Smells using Software Repository Mining.

European Conference on Software Maintenance and Reengineering, 2012

Engineers are aware of code smells, but are not very concerned with their impact, given the low refactoring activity.

Research Questions (RQ)

RQ1

Are some types of code smells refactored more and quicker than other smell types?

Research Questions (RQ1)

	God Class	Feature Envy	Data Class	Message Chain	Long Parameter List
Avg	49,2%	40,39%	55,69%	47,04%	49,41%
Sd	12,1%	18,08%	20,79%	18,76%	12,81%

Research Questions (RQ)

RQ2

Are relatively more code smells being refactored at an early or later stage of a system's life cycle?

Research Questions (RQ)

RQ3

Do some developers refactor more code smells than others and to what extend?

Research Questions (RQ)

RQ4

What refactoring rationales for code smells can be identified?

Research Questions (RQ4)

Cleaning up dead or obsolete code.

Dedicated refactoring.

Maintenance activities.

Venera Arnaoudova, Massimiliano Di Penta, Giuliano Antoniol, Yann-Gaël Guéhéneuc

A New Family of Software Anti-Patterns: Linguistic Anti-Patterns.

European Conference on Software Maintenance and Reengineering, 2013

Linguistic Antipatterns

LAs in software systems are recurring poor practices in the naming, documentation, and choice of identifiers in the implementation of an entity, thus possibly impairing program understanding.

A. Does more than it says

B. Says more than it does

C. Does the opposite

D. Contains more than it says

E. Says more than it contains

F. Contains the opposite

A. "Get" - more than an accessor

```
public ImageData getImageData() {
  Point size = getSize();
  RGB black = new RGB(0, 0, 0);
  RGB[] rgbs = new RGB[256];
  rgbs[0] = black; // transparency
  rgbs[1] = black; // black
  PaletteData dataPalette=new PaletteData(rgbs);
  imageData = new
    ImageData(size.x,size.y,8,dataPalette);
  imageData.transparentPixel = 0;
  drawCompositeImage(size.x, size.y);
  for (int i = 0; i < rgbs.length; i++)
    if (rgbs[i] == null) rgbs[i] = black;
  return imageData;
}
```


A. "Is" - returns more than a Boolean

```
public int isValid() {
 final long currentTime =
        System.currentTimeMillis();
 if (currentTime <= this.expires) {
  // The delay has not passed yet –
  // assuming source is valid.
  return SourceValidity.VALID;
 }
 // The delay has passed,
 // prepare for the next interval.
 this.expires = currentTime + this.delay;
 return this.delegate.isValid();
}
```

A. "Set" - method returns

```
public Dimension setBreadth
        (Dimension target, int source) {
  if (orientation == VERTICAL)
    return new Dimension(source,
        (int)target.getHeight());
  else
    return new Dimension(
                (int)target.getWidth(), source);
}
```


A. Expecting but not getting a single instance

```
/* Returns the expansion state for a tree.
 * @return the expansion state for a tree */
public List getExpansion() { return fExpansion;}
```


B. Not implemented condition

```
/* Returns the children of this object.
 * When this object is displayed in a tree,
 * the returned objects will be this element's
 * children. Returns an empty array if this
 * object has no children.
 * @param object The object to get the
 * children for. */
public Object[] getChildren(Object o)
 { return new Object[0]; }
```


B. Validation method does not confirm

```
public void checkCollision(String before,
        String after){
boolean collision=(before!=null
   && before.equals(_shortName)) ||
    (after!=null&&after.equals(_shortName));
 if(collision) {
  if(_longName==null){_longName=getLongName();}
  _displayName = _longName;
 }
}
```

B. "Get" method does not return

```
protected void getMethodBodies
  (CompilationUnitDeclaration unit,int place){
  //fill the methods bodies in order
  //for the code to be generated
  if (unit.ignoreMethodBodies) {
    unit.ignoreFurtherInvestigation = true;
    return; // if initial diet parse did not
    //work, no need to dig into method bodies.
  }
  if (place < parseThreshold)
    return; //work already done ...
  //real parse of the method....
  parser.scanner.setSourceBuffer(
     unit.compilationResult.
     compilationUnit.getContents());
  if (unit.types != null) {
    for (int i = unit.types.length; --i >= 0;)
      unit.types[i].parseMethod(parser, unit);
        }
}
```


B. Not answered question

```
public void isValid(
        Object[] selection, StatusInfo res) {
  // only single selection
  if (selection.length == 1
          && (selection[0] instanceof IFile))
    res.setOK();
  else res.setError(""); //$NON-NLS-1$
}
```


B. Transform method does not return

```
public void javaToNative(
    Object object, TransferData transferData) {
  byte[] check= TYPE_NAME.getBytes();
  super.javaToNative(check, transferData);
}
```


B. Expecting but not getting a collection

```
public boolean getStats(){ return _stats; }
```

C. Method name and return type are opposite

```
/* Saves the current enable/disable state of
 * the given control and its descendents in the
 * returned object;the controls are all disabled.
 * @param w the control
 * @return an object capturing the enable/disable
 * state */
public static ControlEnableState
    disable(Control w){
        return new ControlEnableState(w);
}
```


C. Method signature and comment are opposite

```
/* Returns true if this listener has a target
 * for a back navigation.  Only one listener
 * needs to return true for the back button
 * to be enabled. */
public boolean isNavigateForwardEnabled() {
  boolean enabled = false;
  if(_isForwardEnabled==1) {enabled = true;}
  else {
    if(_isForwardEnabled != 0) {
      enabled = navigateForward(false) != null;
    }
  }
  return enabled;
}
```


D. Contains more than it says

1. Says one but contains many

   Vector _target;

2. Name suggests Boolean but type does not

   int[] isReached;

E. Says more than it contains

private static boolean _stats = true;

F. Contains the opposite

1. Attribute name and type are opposite

   MAssociationEnd start = null;

2. Attribute signature and comment are opposite

	ArgoUML 0.10.1	ArgoUML 0.34	Cocoon 2.2.0	Eclipse 1.0	Validated	TP	Precision
"Get" - more than an accessor (Section II-A1)	0	2	1	15	18/18	12	67%
"Is" returns more than a Boolean (Section II-A2)	2	0	4	26	24/32	24	100%
"Set" method returns (Section II-A3)	4	30	6	53	47/93	46	98%
Expecting but not getting a single instance (Section II-A4)	7	3	8	33	34/51	26	77%
Not implemented condition (Section II-B1)	20	28	43	232	74/323	58	78 %
Validation method does not confirm (Section II-B2)	1	8	11	235	70/255	52	74%
"Get" method does not return (Section II-B3)	1	3	2	57	38/63	37	97%
Not answered question (Section II-B4)	0	2	0	34	36/36	36	100%
Transform method does not return (Section II-B5)	0	86	15	44	59/145	58	98%
Expecting but not getting a collection (Section II-B6)	8	39	12	135	66/194	49	74%
Method name and return type are opposite (Section II-C1)	0	0	0	6	6/6	3	50%
Method signature and comment are opposite (Section II-C2)	7	20	12	243	72/282	6	8%
Says one but contains many (Section II-D1)	15	92	42	103	70/252	40	57%
Name suggests Boolean but type does not (Section II-D2)	14	13	21	138	64/186	36	56%
Says many but contains one (Section II-E1)	45	117	24	116	73/302	55	75%
Attribute name and type are opposite (Section II-F1)	1	0	0	0	1/1	1	100%
Attribute signature and comment are opposite (Section II-F2)	1	0	3	19	23/23	2	9%