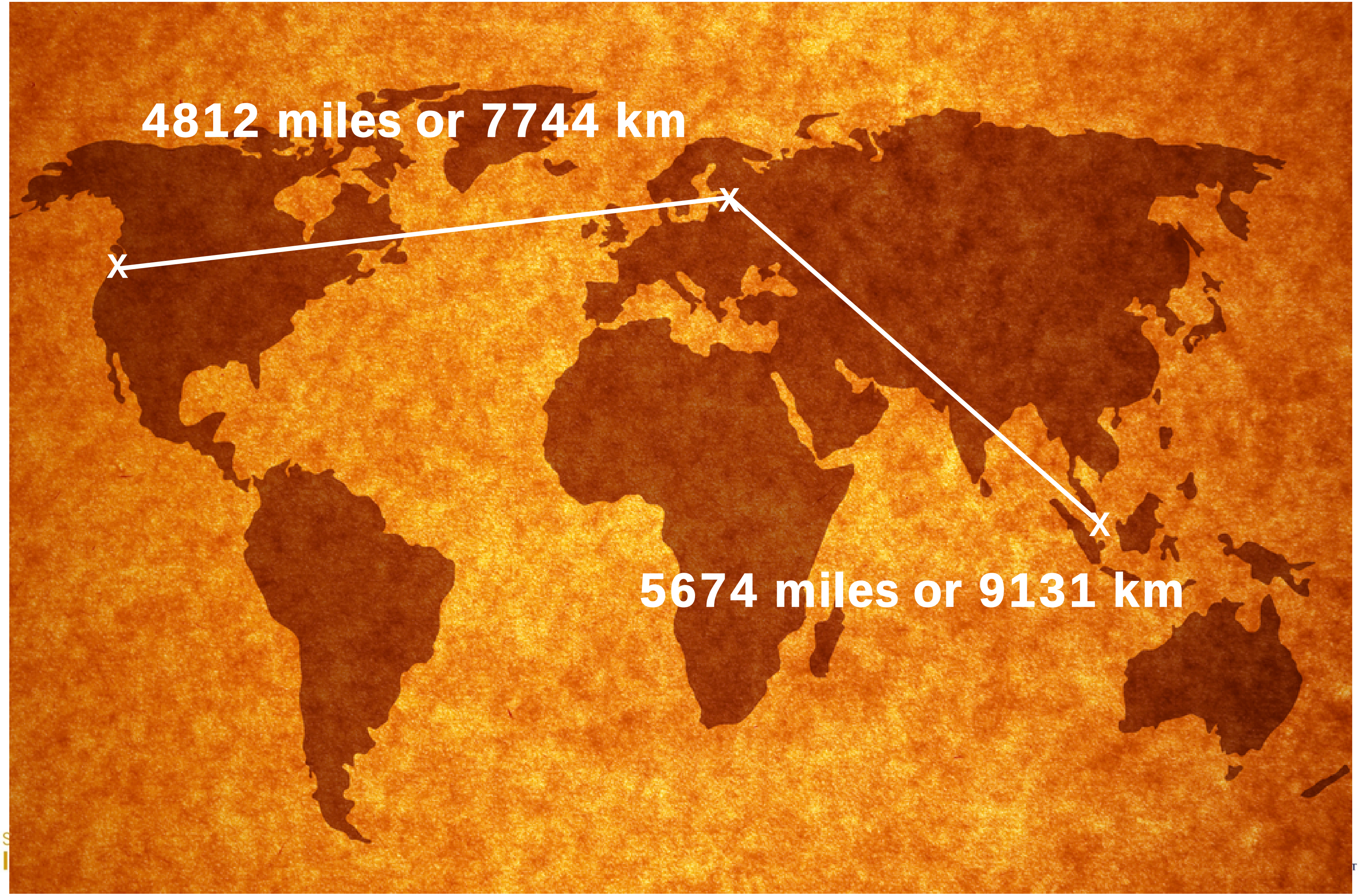
Data Analytics for Automated Software Engineering

David Lo
School of Information Systems
Singapore Management University
davidlo@smu.edu.sg

Short Course, ESSCaSS 2014, Estonia

A Brief Self-Introduction



A Brief Self-Introduction

From
Wikipedia

A Brief Self-Introduction

A Brief Self-Introduction

- Graduated from National Uni. Of Singapore, 2008
- Work on the intersection of software engineering and data mining

Mining Software Traces
Specification Mining
Fault Localization
Malware Detection

Mining Software Text
Bug Report Analysis
Concern Localization
Software Forum Mining

Mining Code
Code Search
Anomaly Detection
Privacy Preserving Testing

Mining Socio-Technical Network
Mining Developer Network
Mining Developer Microblogs
Community Detection

Empirical Studies
Widespread Changes
Feature Diffusion
Effectiveness of Exist. Tools

Data Mining Algorithms
Sequential/Graph Pattern Mining
Discriminative Pattern Mining
Game Mining

Focus of This Short Course

- Highlight research problems in software engineering
- Describe the wealth of software data available for analysis
- Present some data mining concepts and how it can be used to automate software engineering tasks
- Present some information retrieval concepts and how it can be used to automate software engineering tasks

Three Lectures

I. Software Engineering (SE): A Primer

- Challenges & Problems
- Research Topics
- Data Sources
- Basic Tools

II. Data Mining for Automated Software Engineering

- Pattern Mining
- Clustering
- Classification

III. Information Retrieval for Automated SE

- Vector Space Model
- Language Model
- Topic Model
- Text Classification

Software Engineering: A Primer

```
tring
if(parameters.contains(
 hql += " and p.name = :name";
}
if(parameters.contains("age")){
 hql += " and p.age = :age";
}
TypedQuery<Person> query = em.createQ
if(parameters.contains("name")){
query.setParameter("name", values[
("age")){
Intege
```

The most certain way to succeed is always to try just one more time.

- Thomas A. Edison

Slide Outline

- Part I: SE Challenges & Problems
- Part II: Research Topics
  - Software Testing and Reliability
  - Software Maintenance and Evolution
  - Software Analytics: A Recent Trend
- Part III: Data Sources
- Part IV: Basic Program Analysis Tools

Part I: Software Engineering (SE)

"Process and techniques that are followed to design, develop, verify, validate, and maintain a software system that satisfies a set of requirements and properties with reasonable or low cost."

SE: Challenges & Problems

- Building and maintaining complex software system is challenging and costs much resources
  - High cost and scarcity of qualified manpower
  - Software changes over time

- Software systems are plagued with bugs and removing them costs much resources
  - Hard to ensure high reliability of complex systems

SE: Challenges & Problems

- Many other challenges:
    - Hard to capture needs of end users
    - Hard to manage developers working at geographically disparate locations
    - Etc.

Part II: Research Topics in SE

- Software Testing and Reliability
- Software Maintenance and Evolution
- Software Analytics, A Recent Trend
- Empirical Software Engineering
- Requirement Engineering
- Many more

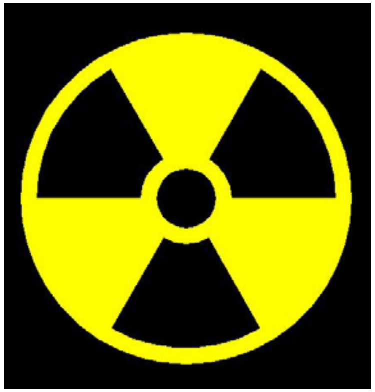
Topics: Software Testing and Reliability

- Software and bugs are often inseparable
- Many systems receive hundreds of bug reports daily (Anvik et al., 2005)
- Software gets more complex
  - Written in multiple languages
  - Written by many people
  - Over a long period of time
  - Increases likelihood of bugs

**Anvik et al.: Coping with an open bug repository. ETX 2005: 35-39*

Software Testing and Reliability: Why Bother?

- Software bugs cost US economy 22.2 – 59.5 billions annually (NIST, 2002)
- Many software bugs have disastrous effects



Therac-25

Ariane-5

Mars Climate Orbiter

**National Institute of Standards and Technology (NIST): The Economic Impacts of Inadequate Infrastructure for Software Testing. Report 2002.*

Software Reliability: Goals

- Ensure the absence of (a family of) bugs
  - Formal verification of a set of properties
  - Heavyweight

Verification

- Prevention and early detection of bugs
  - Does not guarantee the absence of bugs
  - Identify as many bugs as possible as early as possible
  - Lightweight

Testing and Bug Finding

Software Testing and Bug Finding: Goals

- Test adequacy measurement
  - How thorough is a test suite?
  - Does it cover all parts of a code?
- Test adequacy improvement
  - How to create additional test cases?
  - How to make a test suite more thorough?
- Test selection
  - How to reduce the number of test cases to run when a change is made?
- Identification of software bugs

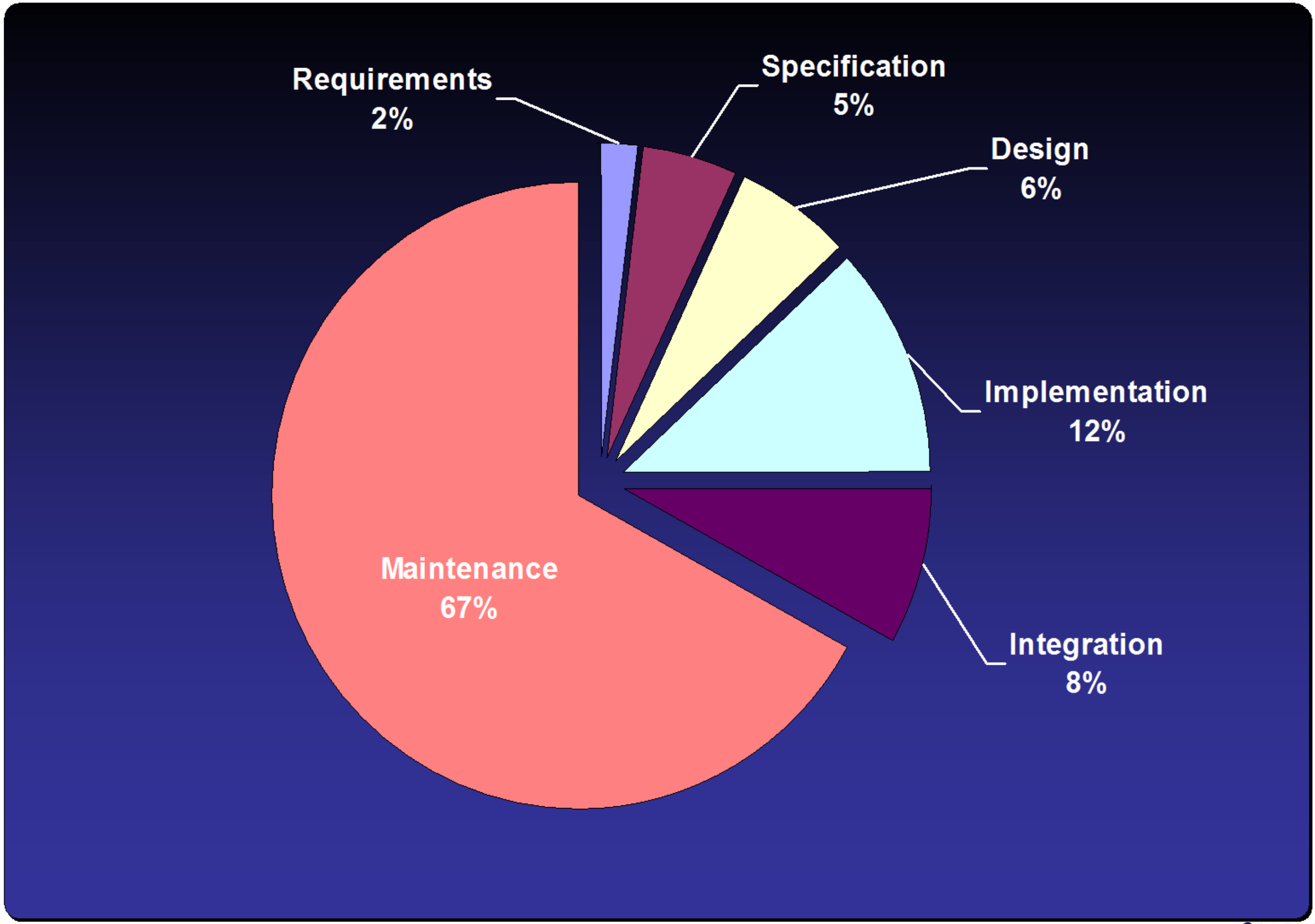
SE Topics: What is Software Evolution?

- A piece of software system becomes gradually more and more different than the original code
- Reasons:
  - Bug fixes
  - New requirements or features
  - Changing environment (e.g., GUI, database, etc.)
  - Code quality improvement

What is Software Maintenance?

- Changes made to existing software system
- Resulting in software evolution
- Types of maintenance tasks:
  - Corrective maintenance: Bug fixes
  - Perfective maintenance: New features
  - Adaptive maintenance: Changing environments
  - Preventive maintenance: Code quality improvement

Software Maintenance – Relative Costs



Why is Software Maintenance Expensive?

- Costs can be high because:
    - Inexperienced maintenance staffs
    - Poor code
    - Poor documentation
    - Changes may introduce new faults, which trigger further changes
    - As a system is changed, its structure tends to degrade, which makes it harder to change

Lehman’s Laws of Evolution

- A classic study by Lehman and Belady (1985) identified several “laws” of system change.
- Continuing change
  - A program that is used in a real-world environment must change, or become progressively less useful in that environment
- Increasing complexity
  - As a program evolves, it becomes more complex, and extra resources are needed to preserve and simplify its structure

WORDSTAR

INTERNATIONAL INCORPORATED

What is a Legacy System?

- Legacy IS are large software systems
- They are old, often more than 10 years old
- They are written in a legacy language (e.g., COBOL), and built around legacy databases
- Legacy ISs are autonomous, and mission critical
- They are inflexible and brittle
- They are responsible for the consumption of at least 80% of the IS budget

Problems of Legacy Systems

- Availability of original developers
- Lack of documentation
- Size and complexity of the software system
- Accumulated past maintenance activities

History of Eclipse

- 1997 – IBM VisualAge for Java (implemented in small talk)
- 1999 – IBM VisualAge for Java micro-edition (Eclipse code based from here)
- 2001 – Eclipse (change name for marketing issue)
- 2003 – Eclipse.org foundation
- 2005 – Eclipse V3.1
- 2006 – Eclipse V3.2
- ....
- 2014 – Eclipse V4.4

History of Microsoft Word

- 1983 – MS Word for DOS
- 1985 – MS Word for Mac
- 1980 – MS Word for Windows
- 1991 – MS Word 2
- 1993 – MS Word 6
- 1995 – MS Word 95
- 1997 – MS Word 97
- 1998 – MS Word 98
- 2000 – MS Word 2000
- 2002 – MS Word XP
- 2003 – MS Word 2003
- …
- 2014 – MS Word 2013

Topics: Software Analytics

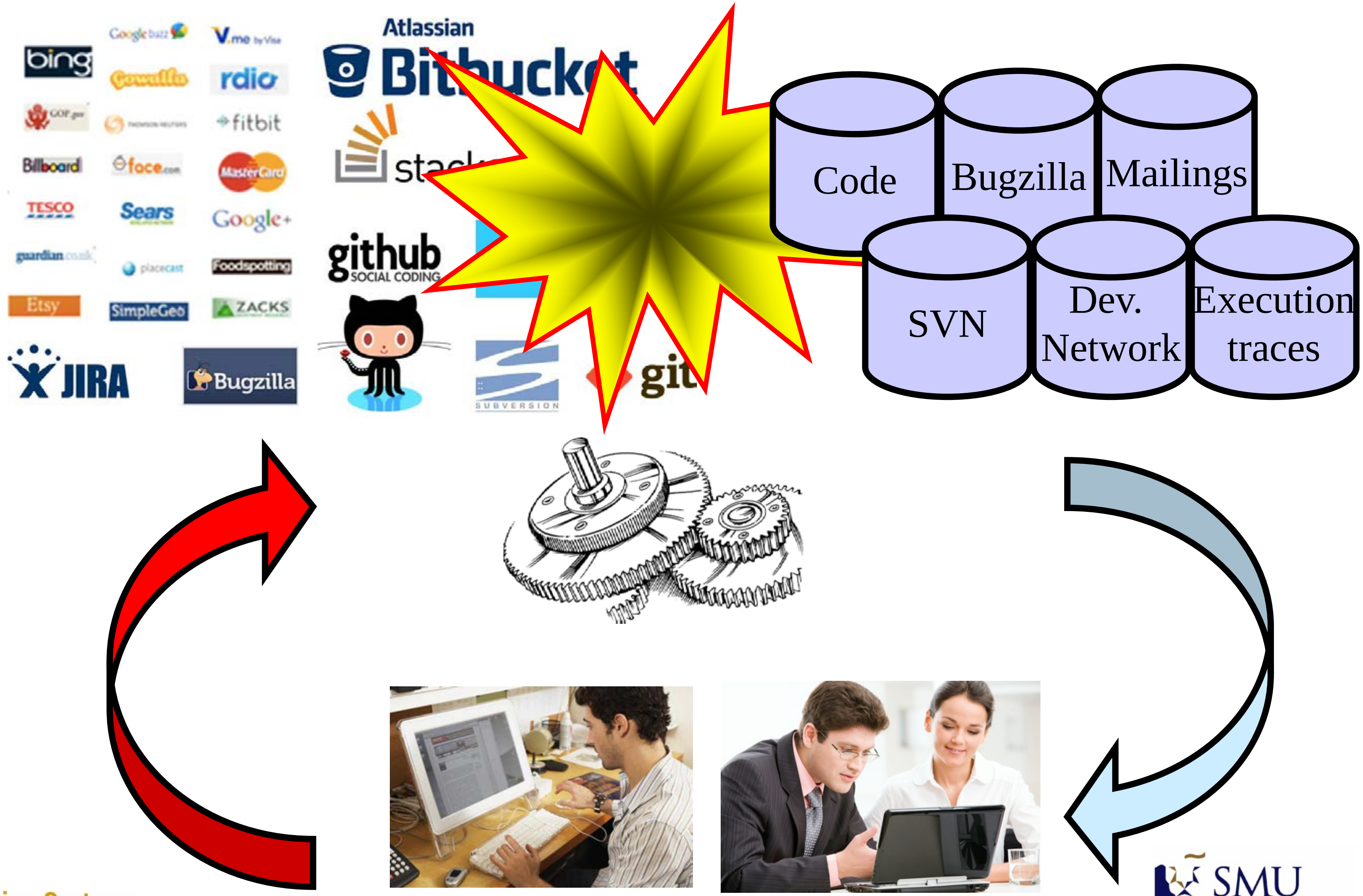
”Data exploration and analysis in order to obtain insightful and actionable information for data-driven tasks around software and services”

(Zhang and Xie, 2012)

Software Analytics: Definition

- Analysis of a large amount of software data stored in various repositories in order to:
    - Understand software development process
    - Help improve software maintenance
    - Help improve software reliability
    - And more

Topics: Software Analytics



Code
Bugzilla
Mailings
SVN
Dev. Network
Execution traces

Big Data for Software Engineering

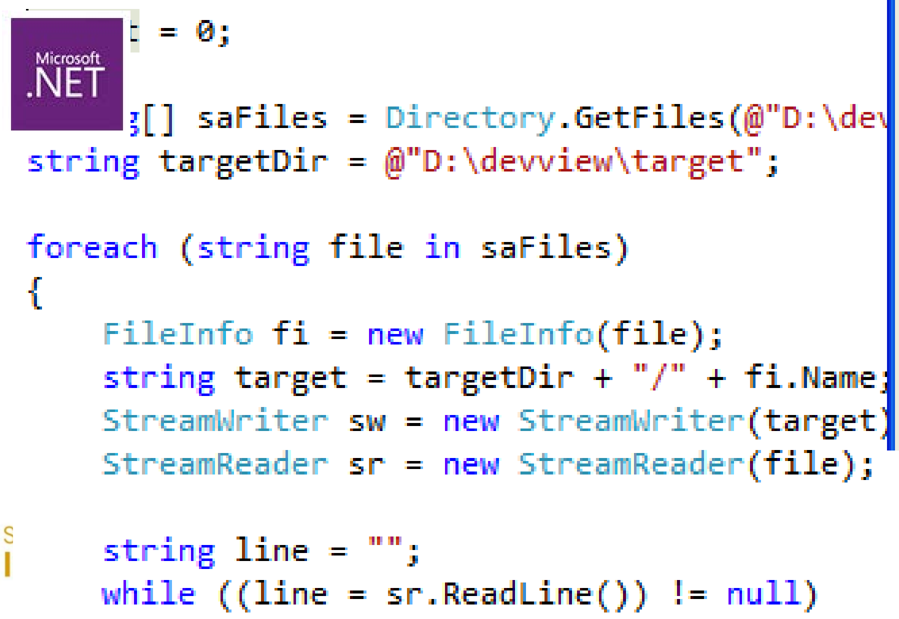
Millions of Projects
Millions of Bug Reports
Millions of Repositories
Millions of Blogs and Posts
Thousands of APIs
….

Part III: Software Data Sources

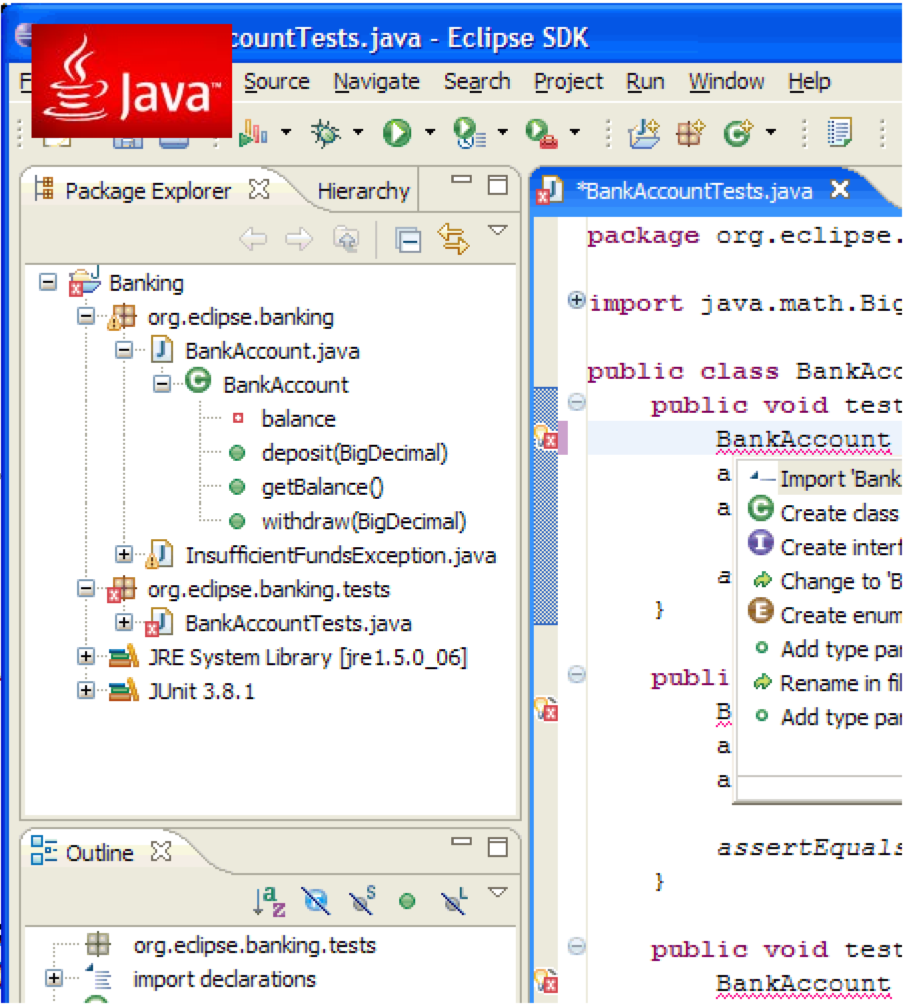
- Source Code
- Execution Trace
- Development History
- Bug Reports
- Developer Activities
- Software Forums
- Software Microblogs
- Other Artifacts

Artifact: Source Code

```
#!/usr/bin/perl -w
use strict;
$| = 1;
my $filename = $ARGV[0] || undef;
my $output = $ARGV[1] || undef;
if ( !defined($filename) or !defined($output) ) {
    print "Usage: \n";
    print "\t$0 inputfile outputfile\n";
}
else {
    open (FILE, $filename);
    my $data = join('', <FILE>);
    close FILE;
    $data =~ s/<\s*([A-Za-z0-9]+)(\s*)(.*?)>/<$1 value="$3"\/>
    $data =~ s/\&/\&amp\;/sg;
    open (FILE, ">$output");
```



```
: = 0;

;[] saFiles = Directory.GetFiles(@"D:\dev
string targetDir = @"D:\devview\target";

foreach (string file in saFiles)
{
    FileInfo fi = new FileInfo(file);
    string target = targetDir + "/" + fi.Name;
    StreamWriter sw = new StreamWriter(target)
    StreamReader sr = new StreamReader(file);

    string line = "";
    while ((line = sr.ReadLine()) != null)
```



```
package org.eclipse.

import java.math.Big

public class BankAcc
    public void test
        BankAccount
        a
        a

        a
    }

    publi
        B
        a
        a

        assertEquals
    }

    public void test
        BankAccount
```

Artifact: Source Code

- Where to find code?
  - Google code: http://code.google.com/
  - Many other places online

- How to analyze source code?
  - Analyze -> automatically parse and understand
  - Program analysis tools

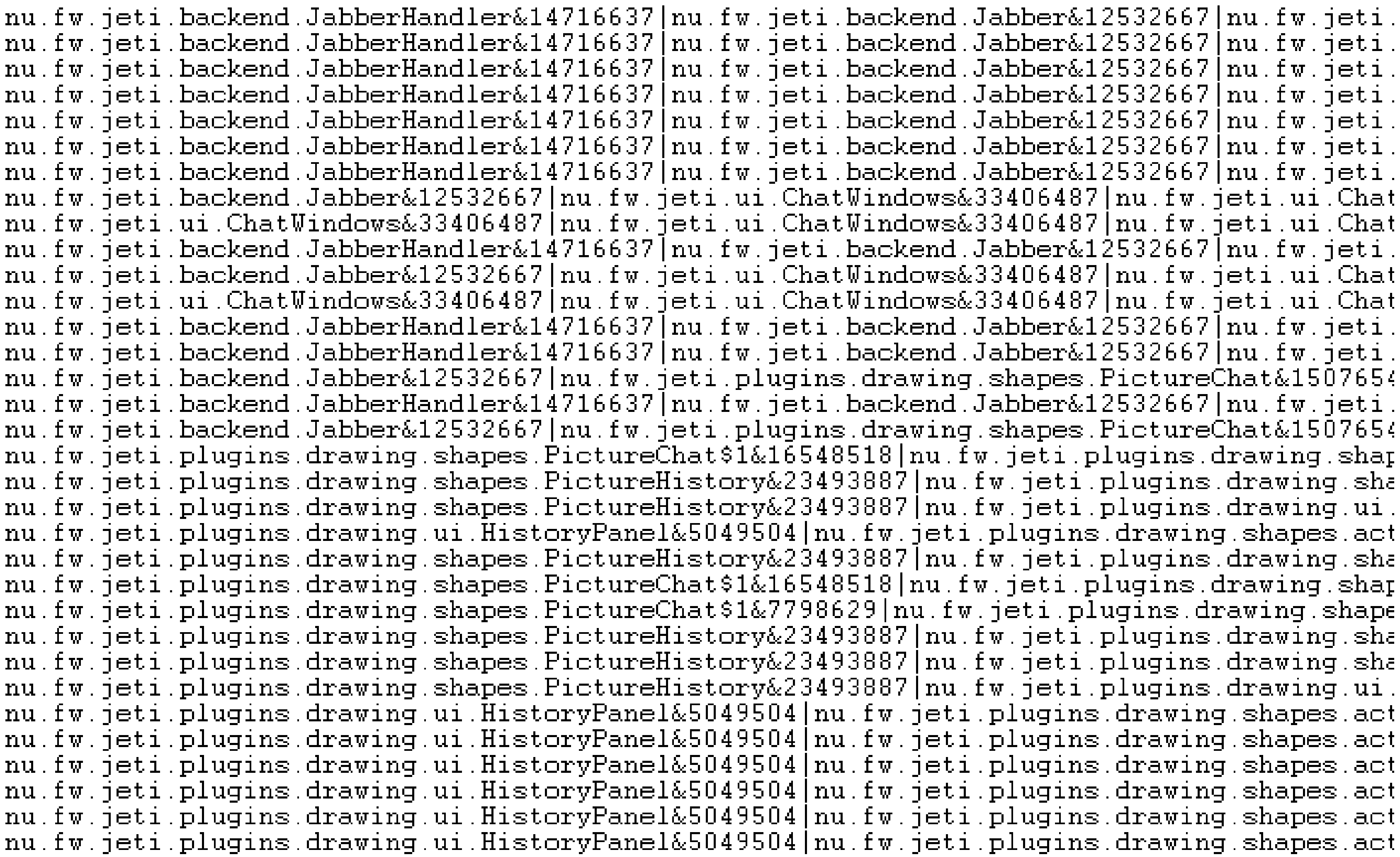
Artifact: Source Code

- Various languages
- Various kinds of systems
- Various scale: small, medium, large
- Various complexities
  - Cyclomatic Complexity
- Various programming styles

Artifact: Execution Trace

- Information collected when a program is run
- What kind of information is collected?
    - Sequences of methods that are executed
    - State of various variables at various times
    - State of various invariants at various times
    - Which components are loaded at various times

Artifact: Execution Traces

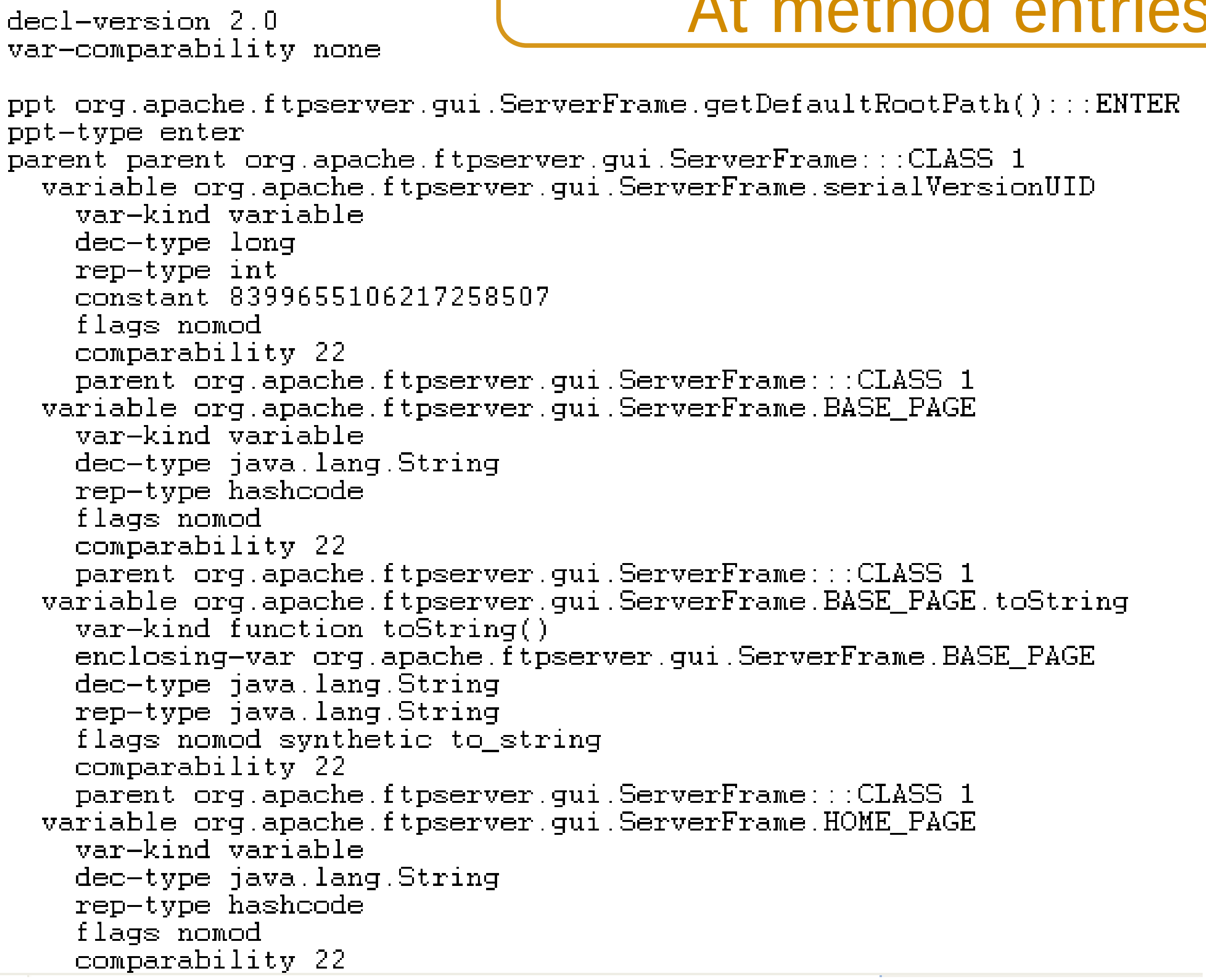


```
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.ui.ChatWindows&33406487|nu.fw.jeti.ui.Chat
nu.fw.jeti.ui.ChatWindows&33406487|nu.fw.jeti.ui.ChatWindows&33406487|nu.fw.jeti.ui.Chat
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.ui.ChatWindows&33406487|nu.fw.jeti.ui.Chat
nu.fw.jeti.ui.ChatWindows&33406487|nu.fw.jeti.ui.ChatWindows&33406487|nu.fw.jeti.ui.Chat
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.plugins.drawing.shapes.PictureChat&150765
nu.fw.jeti.backend.JabberHandler&14716637|nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.
nu.fw.jeti.backend.Jabber&12532667|nu.fw.jeti.plugins.drawing.shapes.PictureChat&150765
nu.fw.jeti.plugins.drawing.shapes.PictureChat$1&16548518|nu.fw.jeti.plugins.drawing.shap
nu.fw.jeti.plugins.drawing.shapes.PictureHistory&23493887|nu.fw.jeti.plugins.drawing.sha
nu.fw.jeti.plugins.drawing.shapes.PictureHistory&23493887|nu.fw.jeti.plugins.drawing.ui.
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
nu.fw.jeti.plugins.drawing.shapes.PictureHistory&23493887|nu.fw.jeti.plugins.drawing.sha
nu.fw.jeti.plugins.drawing.shapes.PictureChat$1&16548518|nu.fw.jeti.plugins.drawing.shap
nu.fw.jeti.plugins.drawing.shapes.PictureChat$1&7798629|nu.fw.jeti.plugins.drawing.shape
nu.fw.jeti.plugins.drawing.shapes.PictureHistory&23493887|nu.fw.jeti.plugins.drawing.sha
nu.fw.jeti.plugins.drawing.shapes.PictureHistory&23493887|nu.fw.jeti.plugins.drawing.sha
nu.fw.jeti.plugins.drawing.shapes.PictureHistory&23493887|nu.fw.jeti.plugins.drawing.ui.
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
nu.fw.jeti.plugins.drawing.ui.HistoryPanel&5049504|nu.fw.jeti.plugins.drawing.shapes.act
```

Caller | Callee | Method Signature

Artifact: Execution Trace

Chicory Trace: Variable values
At method entries and exits



```
decl-version 2.0
var-comparability none

ppt org.apache.ftpserver.gui.ServerFrame.getDefaultRootPath():::ENTER
ppt-type enter
parent parent org.apache.ftpserver.gui.ServerFrame:::CLASS 1
  variable org.apache.ftpserver.gui.ServerFrame.serialVersionUID
    var-kind variable
    dec-type long
    rep-type int
    constant 8399655106217258507
    flags nomod
    comparability 22
    parent org.apache.ftpserver.gui.ServerFrame:::CLASS 1
  variable org.apache.ftpserver.gui.ServerFrame.BASE_PAGE
    var-kind variable
    dec-type java.lang.String
    rep-type hashcode
    flags nomod
    comparability 22
    parent org.apache.ftpserver.gui.ServerFrame:::CLASS 1
  variable org.apache.ftpserver.gui.ServerFrame.BASE_PAGE.toString
    var-kind function toString()
    enclosing-var org.apache.ftpserver.gui.ServerFrame.BASE_PAGE
    dec-type java.lang.String
    rep-type java.lang.String
    flags nomod synthetic to_string
    comparability 22
    parent org.apache.ftpserver.gui.ServerFrame:::CLASS 1
  variable org.apache.ftpserver.gui.ServerFrame.HOME_PAGE
    var-kind variable
    dec-type java.lang.String
    rep-type hashcode
    flags nomod
    comparability 22
```

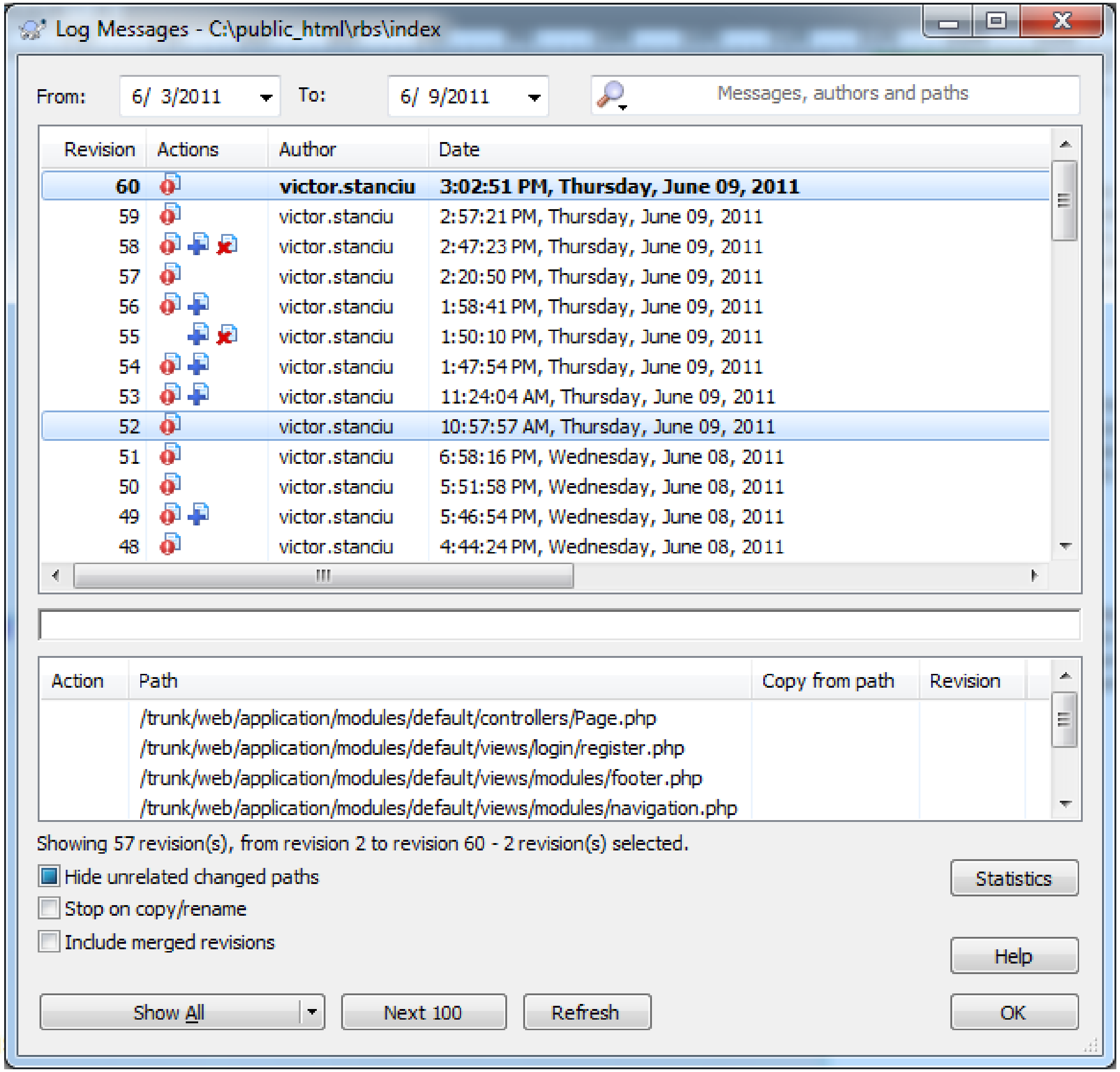
Artifact: Execution Trace

- How to collect?
  - Insert instrumentation code
  - Execute program
  - Instrumentation code writes a log file
- What tools are available to collect traces?
  - Daikon Chicory: http://groups.csail.mit.edu/pag/daikon/dist/doc/daikon.html
  - PIN: http://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool
  - Valgrind: http://valgrind.org/

Artifact: Development History

- What code is
  - Added
  - Deleted
  - Edited
- When
- By Whom
- For What Reason

Artifact: Development History



Log Messages - C:\public_html\rbs\index

From: 6/ 3/2011 To: 6/ 9/2011

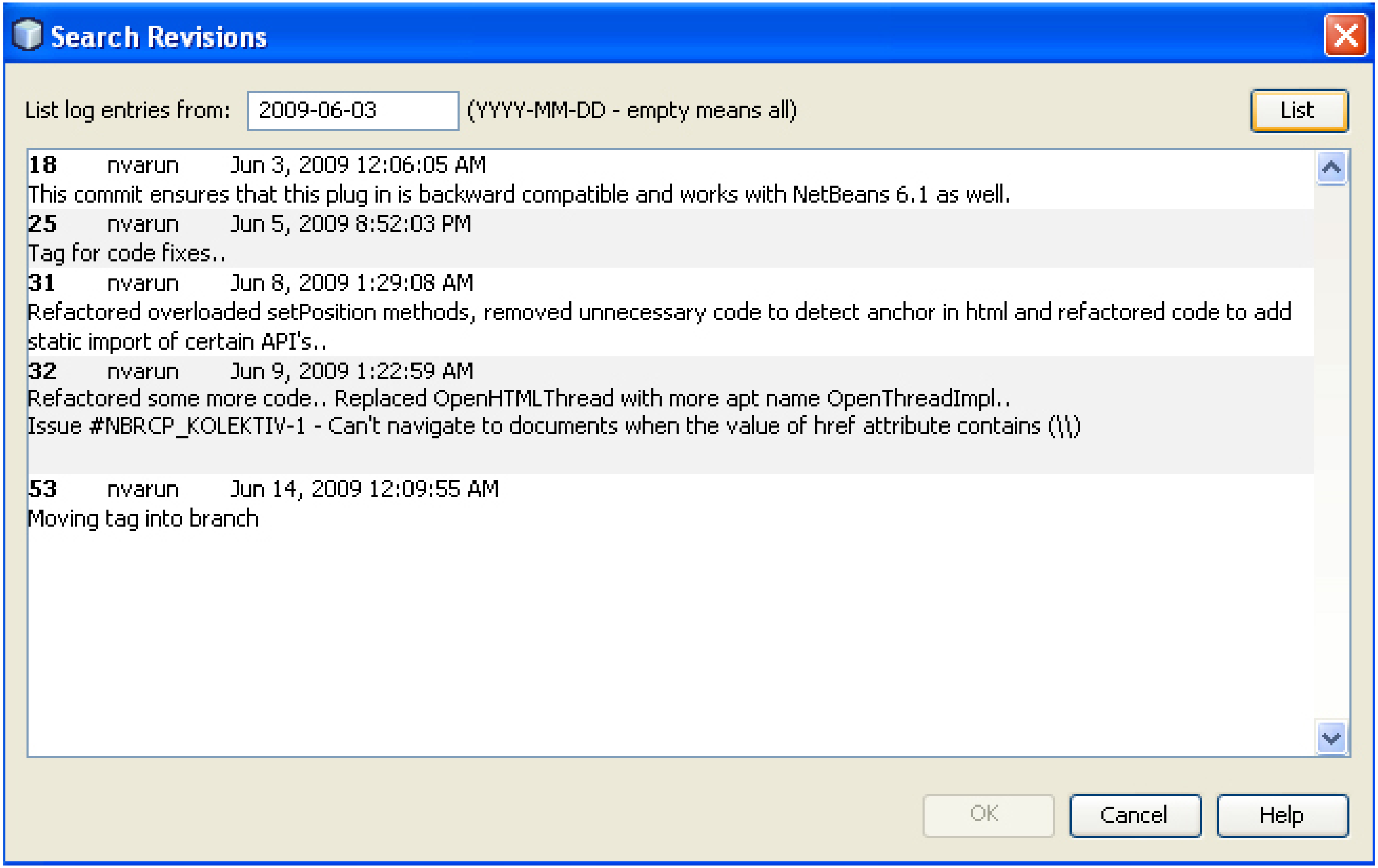
Revision	Actions	Author	Date
60	!	**victor.stanciu**	**3:02:51 PM, Thursday, June 09, 2011**
59	!	victor.stanciu	2:57:21 PM, Thursday, June 09, 2011
58	! + ✗	victor.stanciu	2:47:23 PM, Thursday, June 09, 2011
57	!	victor.stanciu	2:20:50 PM, Thursday, June 09, 2011
56	! +	victor.stanciu	1:58:41 PM, Thursday, June 09, 2011
55	+ ✗	victor.stanciu	1:50:10 PM, Thursday, June 09, 2011
54	! +	victor.stanciu	1:47:54 PM, Thursday, June 09, 2011
53	! +	victor.stanciu	11:24:04 AM, Thursday, June 09, 2011
52	!	victor.stanciu	10:57:57 AM, Thursday, June 09, 2011
51	!	victor.stanciu	6:58:16 PM, Wednesday, June 08, 2011
50	!	victor.stanciu	5:51:58 PM, Wednesday, June 08, 2011
49	! +	victor.stanciu	5:46:54 PM, Wednesday, June 08, 2011
48	!	victor.stanciu	4:44:24 PM, Wednesday, June 08, 2011

Action	Path	Copy from path	Revision
	/trunk/web/application/modules/default/controllers/Page.php		
	/trunk/web/application/modules/default/views/login/register.php		
	/trunk/web/application/modules/default/views/modules/footer.php		
	/trunk/web/application/modules/default/views/modules/navigation.php		

Showing 57 revision(s), from revision 2 to revision 60 - 2 revision(s) selected.

☑ Hide unrelated changed paths
☐ Stop on copy/rename
☐ Include merged revisions

Artifact: Development History



List log entries from: 2009-06-03 (YYYY-MM-DD - empty means all)

```
18      nvarun      Jun 3, 2009 12:06:05 AM
This commit ensures that this plug in is backward compatible and works with NetBeans 6.1 as well.
25      nvarun      Jun 5, 2009 8:52:03 PM
Tag for code fixes..
31      nvarun      Jun 8, 2009 1:29:08 AM
Refactored overloaded setPosition methods, removed unnecessary code to detect anchor in html and refactored code to add
static import of certain API's..
32      nvarun      Jun 9, 2009 1:22:59 AM
Refactored some more code.. Replaced OpenHTMLThread with more apt name OpenThreadImpl..
Issue #NBRCP_KOLEKTIV-1 - Can't navigate to documents when the value of href attribute contains (\\)

53      nvarun      Jun 14, 2009 12:09:55 AM
Moving tag into branch
```

Artifact: Development History

- Useful for distributed software development
  - Various people updating different parts of code
- Easy to backtrack changes
- Easier to find out answer to the question:
  - My code works yesterday but not today. Why?
- Easier to quantify contributions of team members

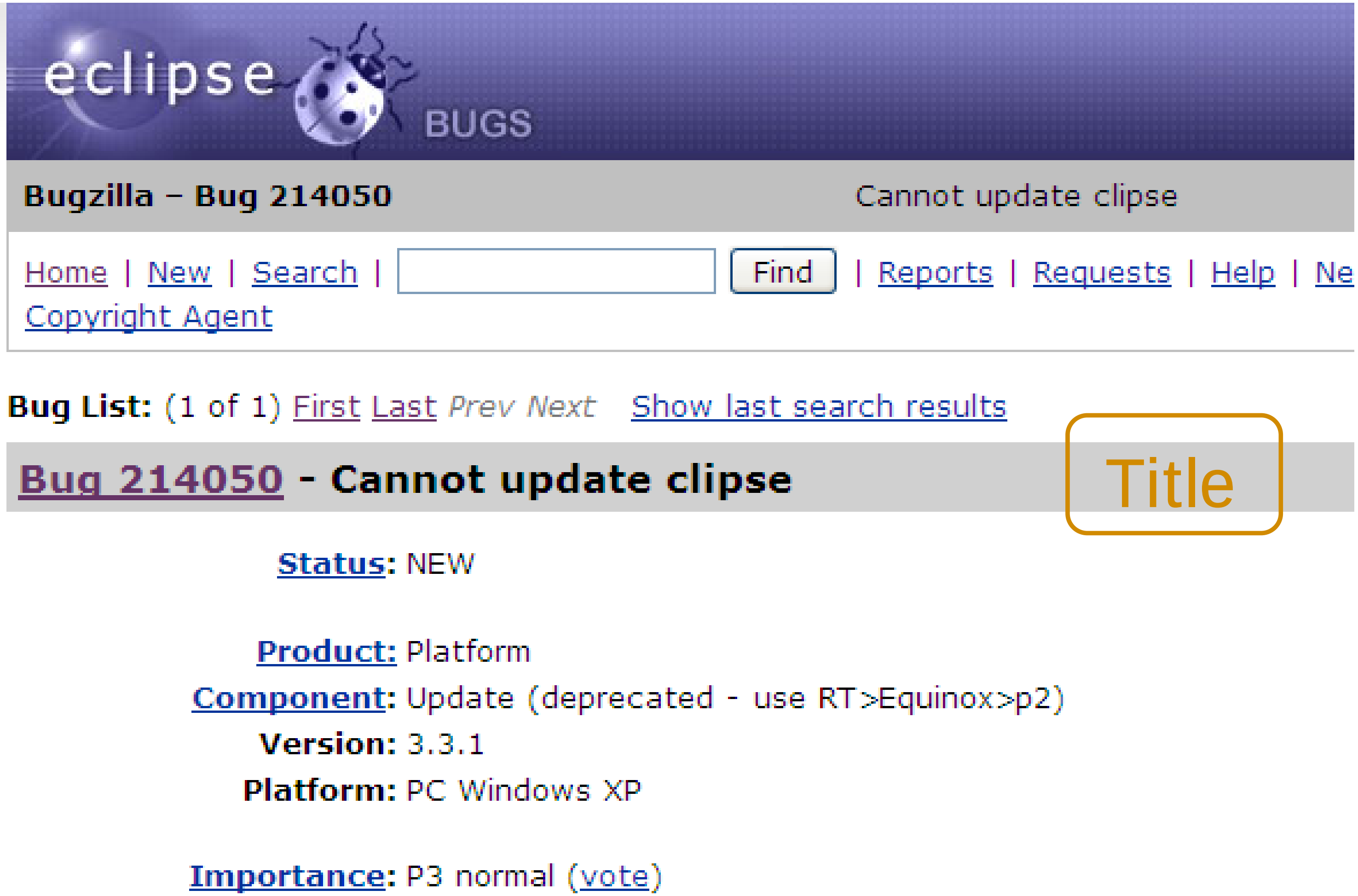
Artifact: Development History

- Various tools
  - CVS – Version per file
  - SVN – Version per snapshot
  - Git - Distributed
- Slightly different ways to manage content

Artifact: Bug Reports

- People report errors and issues that they encounter in the field
- These errors include:
    - Description of the bugs
    - Steps to reproduce the bugs
    - Severity level
    - Parts of the system affected by the bug
    - Failure traces

Artifact: Bug Reports



Bugzilla – Bug 214050 Cannot update clipse

Bug List: (1 of 1) First Last *Prev Next* Show last search results

Bug 214050 - Cannot update clipse

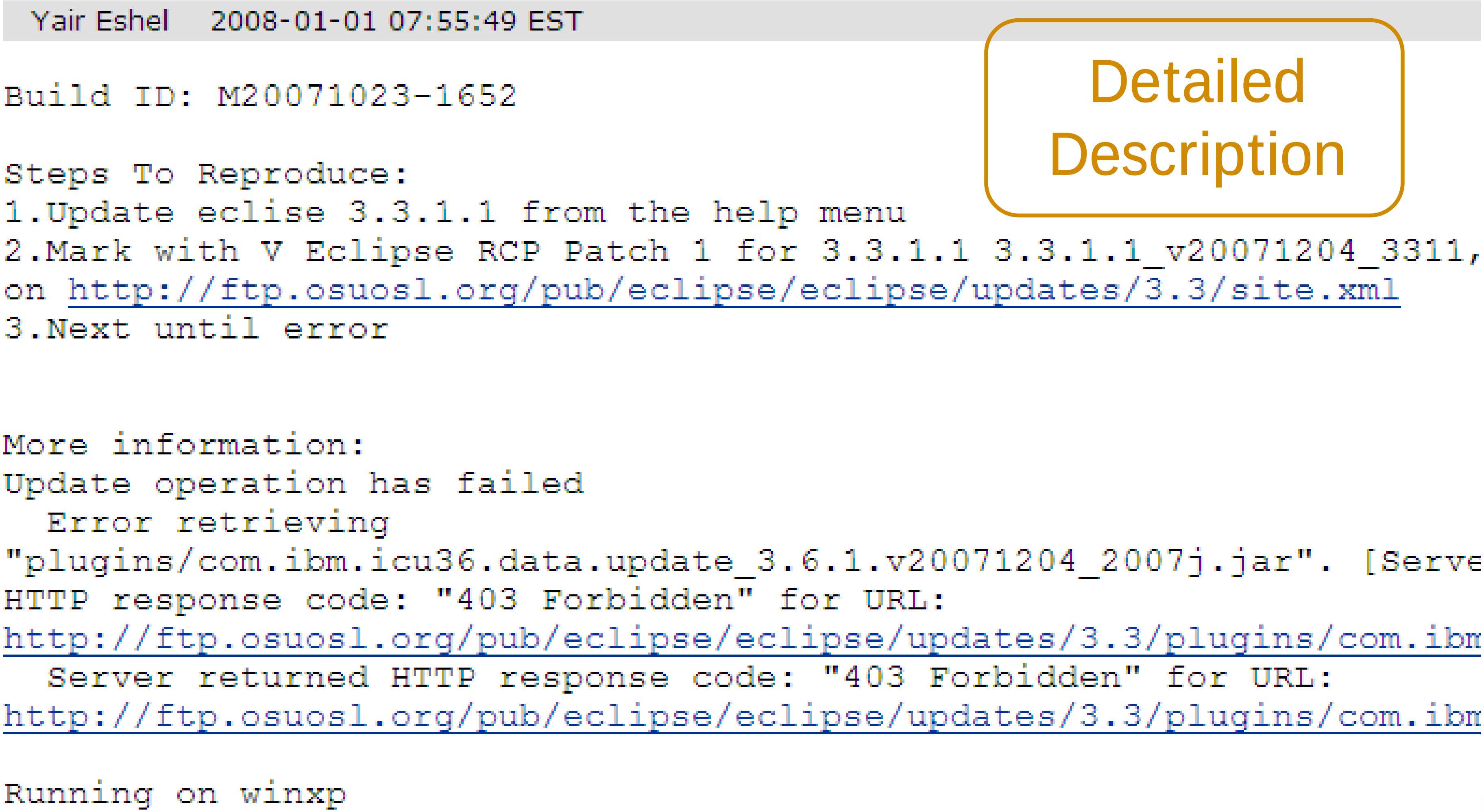
Title

Status: NEW

Product: Platform
Component: Update (deprecated - use RT>Equinox>p2)
Version: 3.3.1
Platform: PC Windows XP

Importance: P3 normal (vote)

Artifact: Bug Reports



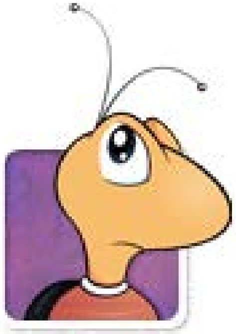
Yair Eshel 2008-01-01 07:55:49 EST

```
Build ID: M20071023-1652

Steps To Reproduce:
1.Update eclise 3.3.1.1 from the help menu
2.Mark with V Eclipse RCP Patch 1 for 3.3.1.1 3.3.1.1_v20071204_3311,
on http://ftp.osuosl.org/pub/eclipse/eclipse/updates/3.3/site.xml
3.Next until error


More information:
Update operation has failed
  Error retrieving
"plugins/com.ibm.icu36.data.update_3.6.1.v20071204_2007j.jar". [Serve
HTTP response code: "403 Forbidden" for URL:
http://ftp.osuosl.org/pub/eclipse/eclipse/updates/3.3/plugins/com.ibm
  Server returned HTTP response code: "403 Forbidden" for URL:
http://ftp.osuosl.org/pub/eclipse/eclipse/updates/3.3/plugins/com.ibm

Running on winxp
```

Detailed Description

Artifact: Bug Reports

- Various kinds of bug repositories
  - BugZilla: http://www.bugzilla.org/
    - Example site: https://bugzilla.mozilla.org/
  - JIRA: http://www.atlassian.com/software/jira/
    - Example site: https://issues.apache.org/jira/browse/WW



Artifact: Developer Activities

- Developers form a social network
  - Developers work on various projects
  - Projects have various types, programming languages and developers
  - Developers follow updates from various other developers and projects
    - Social coding sites
- *A* heterogeneous social network is formed

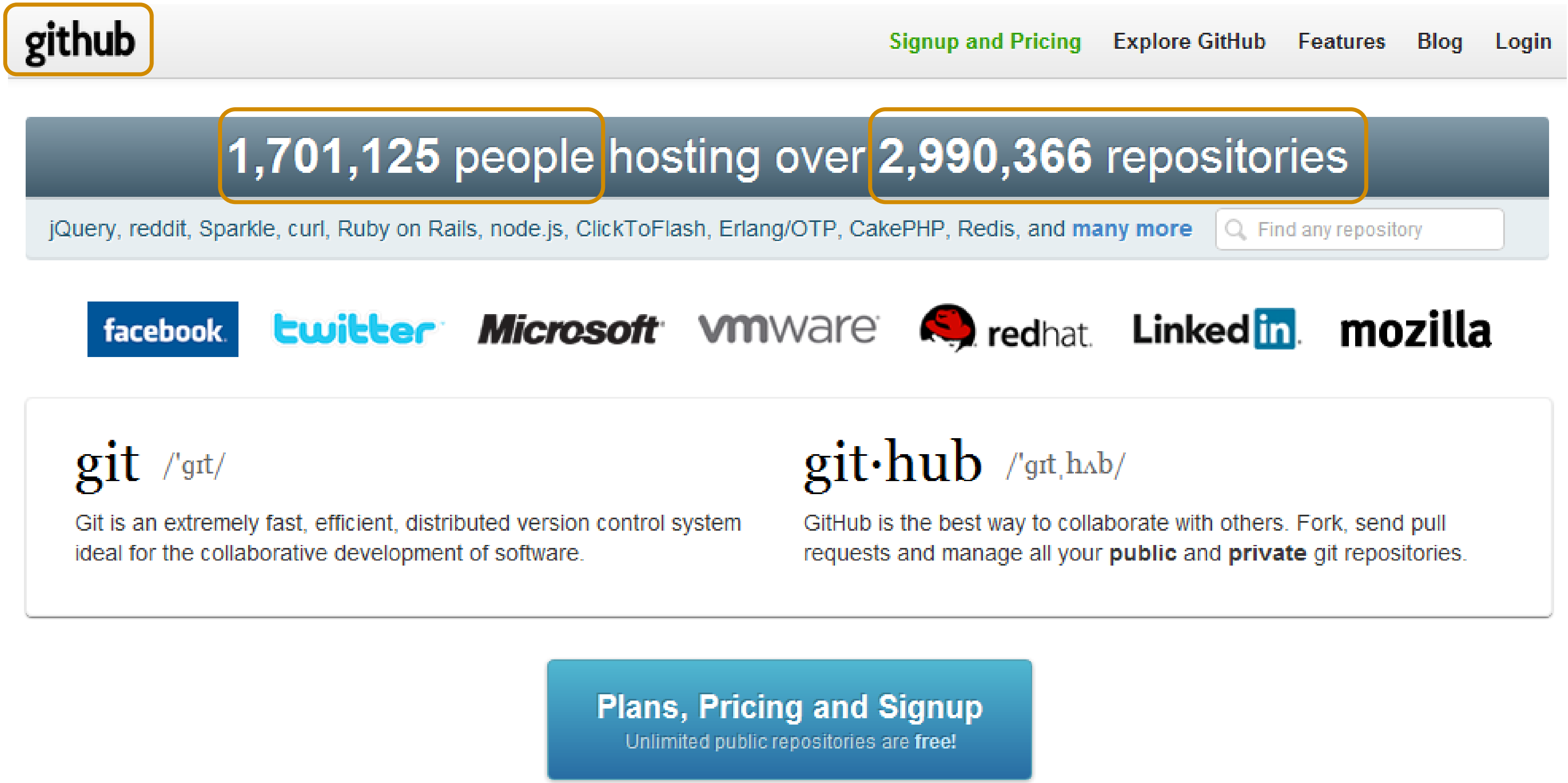
Artifact: Developer Activities

github
SOCIAL CODING

Atlassian
bitbucket

Social Coding Sites

Artifact: Developer Activities



github

Signup and Pricing Explore GitHub Features Blog Login

1,701,125 people hosting over 2,990,366 repositories

jQuery, reddit, Sparkle, curl, Ruby on Rails, node.js, ClickToFlash, Erlang/OTP, CakePHP, Redis, and **many more**

Find any repository

git /ˈɡɪt/

Git is an extremely fast, efficient, distributed version control system ideal for the collaborative development of software.

git·hub /ˈɡɪtˌhʌb/

GitHub is the best way to collaborate with others. Fork, send pull requests and manage all your **public** and **private** git repositories.

Plans, Pricing and Signup

Unlimited public repositories are **free!**

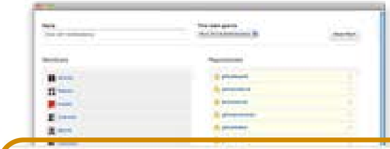
More than a year ago

Features

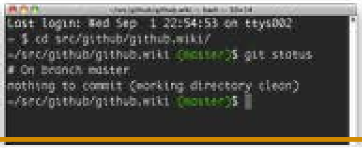
Project Management · Code Hosting · Community

MooTools Core Repository — Read more
http://mootools.net

Every repository on GitHub comes with the tools needed to manage your project. Open to the community for public projects – secured for private projects.

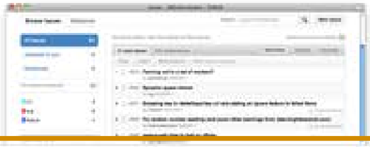


Collaboration
manage large teams with ease

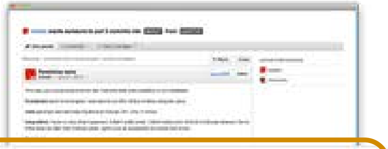


```
Last login: Wed Sep  1 22:54:53 on ttys002
- $ cd src/github/github.wiki/
~/src/github/github.wiki (master)$ git status
# On branch master
nothing to commit (working directory clean)
~/src/github/github.wiki (master)$ 
```

Git Powered Wikis
clone, push, branch and tag it



Integrated Issue Tracking
bug tracking tied to commits



Code Review
living discussions about code

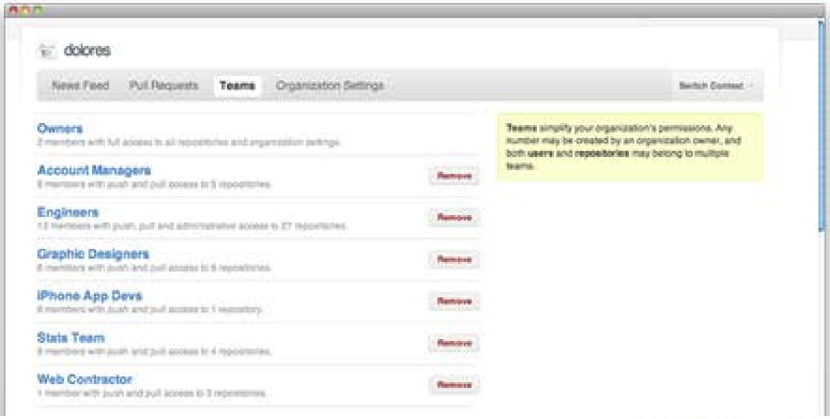
Manage Teams with Organizations

Whether you're running an open source project or a Fortune 500 company, Organizations simplify team management.

With **teams** you can give your developers as much or as little power as they need, from the ability to create projects on behalf of your organization to read-only access on existing projects.

Team permissions: Read-only, read-write, and admin-level access.

Best of all: create **as many** teams with as many members as you need.



Owners
2 members with full access to all repositories and organization settings.

Account Managers
5 members with push and pull access to 5 repositories.

Engineers
12 members with push, pull and administrative access to 27 repositories.

Graphic Designers
8 members with push and pull access to 6 repositories.

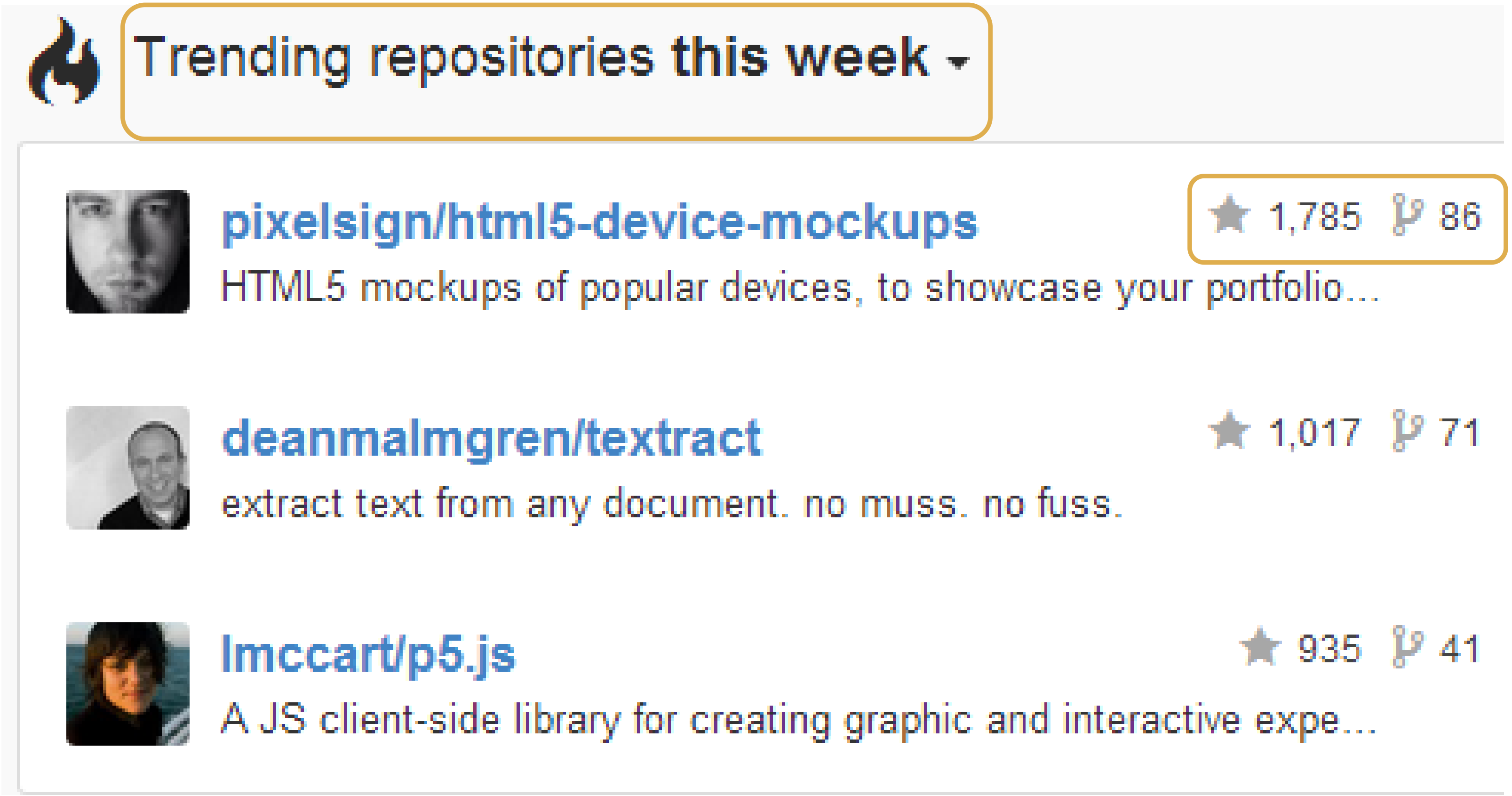
iPhone App Devs
8 members with push and pull access to 1 repository.

Stats Team
3 members with push and pull access to 4 repositories.

Web Contractor
1 member with push and pull access to 3 repositories.

Teams simplify your organization's permissions. Any number may be created by an organization owner, and both **users** and **repositories** may belong to multiple teams.

Artifact: Developer Activities



Trending repositories this week

pixelsign/html5-device-mockups ★ 1,785 ⑂ 86

HTML5 mockups of popular devices, to showcase your portfolio…

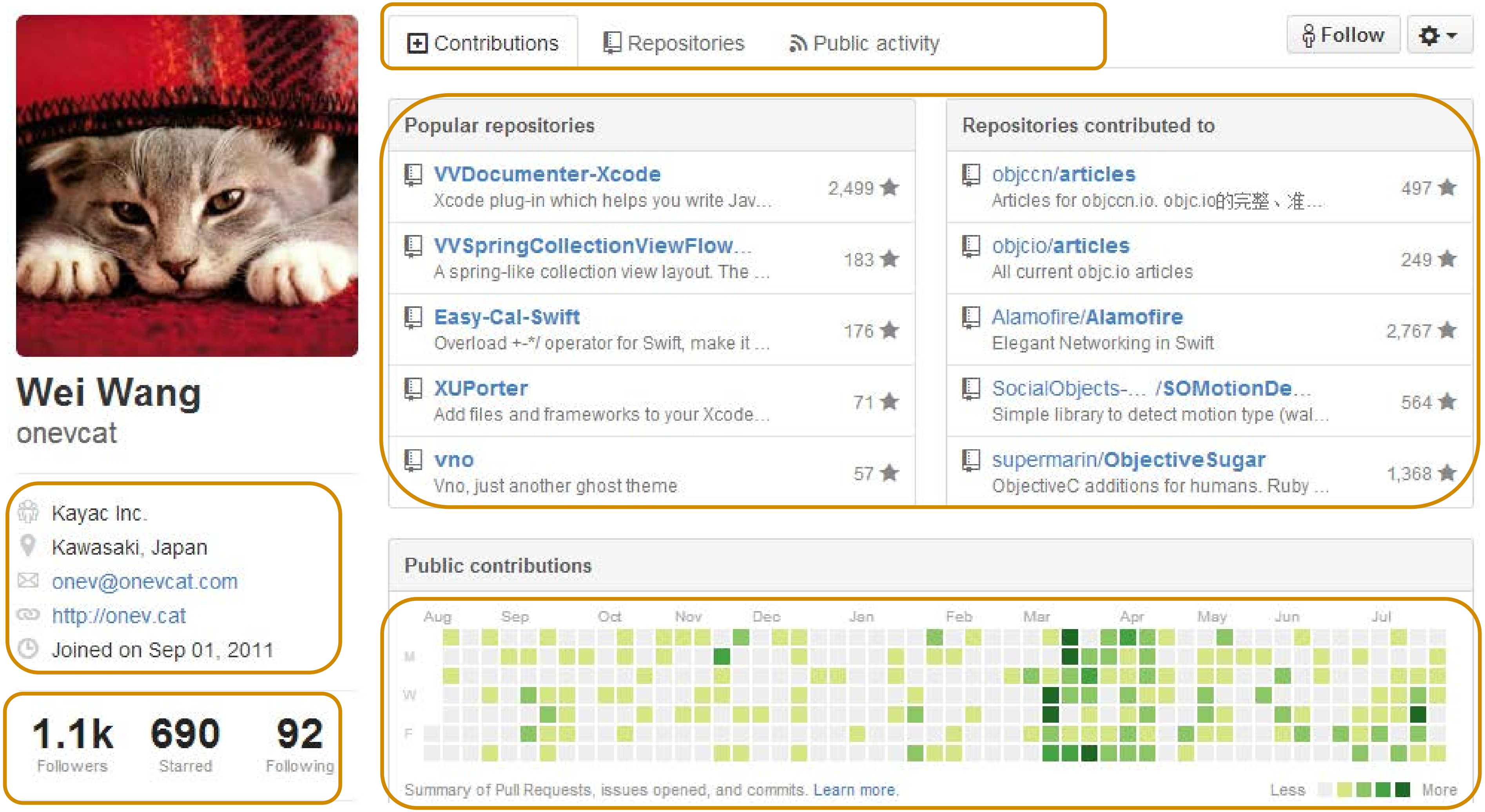
deanmalmgren/textract ★ 1,017 ⑂ 71

extract text from any document. no muss. no fuss.

lmccart/p5.js ★ 935 ⑂ 41

A JS client-side library for creating graphic and interactive expe…

Artifact: Developer Activities



Contributions Repositories Public activity

Follow

Wei Wang
onevcat

Kayac Inc.
Kawasaki, Japan
onev@onevcat.com
http://onev.cat
Joined on Sep 01, 2011

1.1k	690	92
Followers	Starred	Following

Popular repositories

VVDocumenter-Xcode	Xcode plug-in which helps you write Jav…	2,499 ★
VVSpringCollectionViewFlow…	A spring-like collection view layout. The …	183 ★
Easy-Cal-Swift	Overload +-*/ operator for Swift, make it …	176 ★
XUPorter	Add files and frameworks to your Xcode…	71 ★
vno	Vno, just another ghost theme	57 ★

Repositories contributed to

objccn/**articles**	Articles for objccn.io. objc.io的完整、准…	497 ★
objcio/**articles**	All current objc.io articles	249 ★
Alamofire/**Alamofire**	Elegant Networking in Swift	2,767 ★
SocialObjects-… /**SOMotionDe…**	Simple library to detect motion type (wal…	564 ★
supermarin/**ObjectiveSugar**	ObjectiveC additions for humans. Ruby …	1,368 ★

Public contributions

Summary of Pull Requests, issues opened, and commits. Learn more.

Less More

Artifact: Developer Activities

reddit

San Francisco, CA http://www.reddit.com/r/r...

reddit Python ★ 5,996 1,343

the code that powers reddit.com

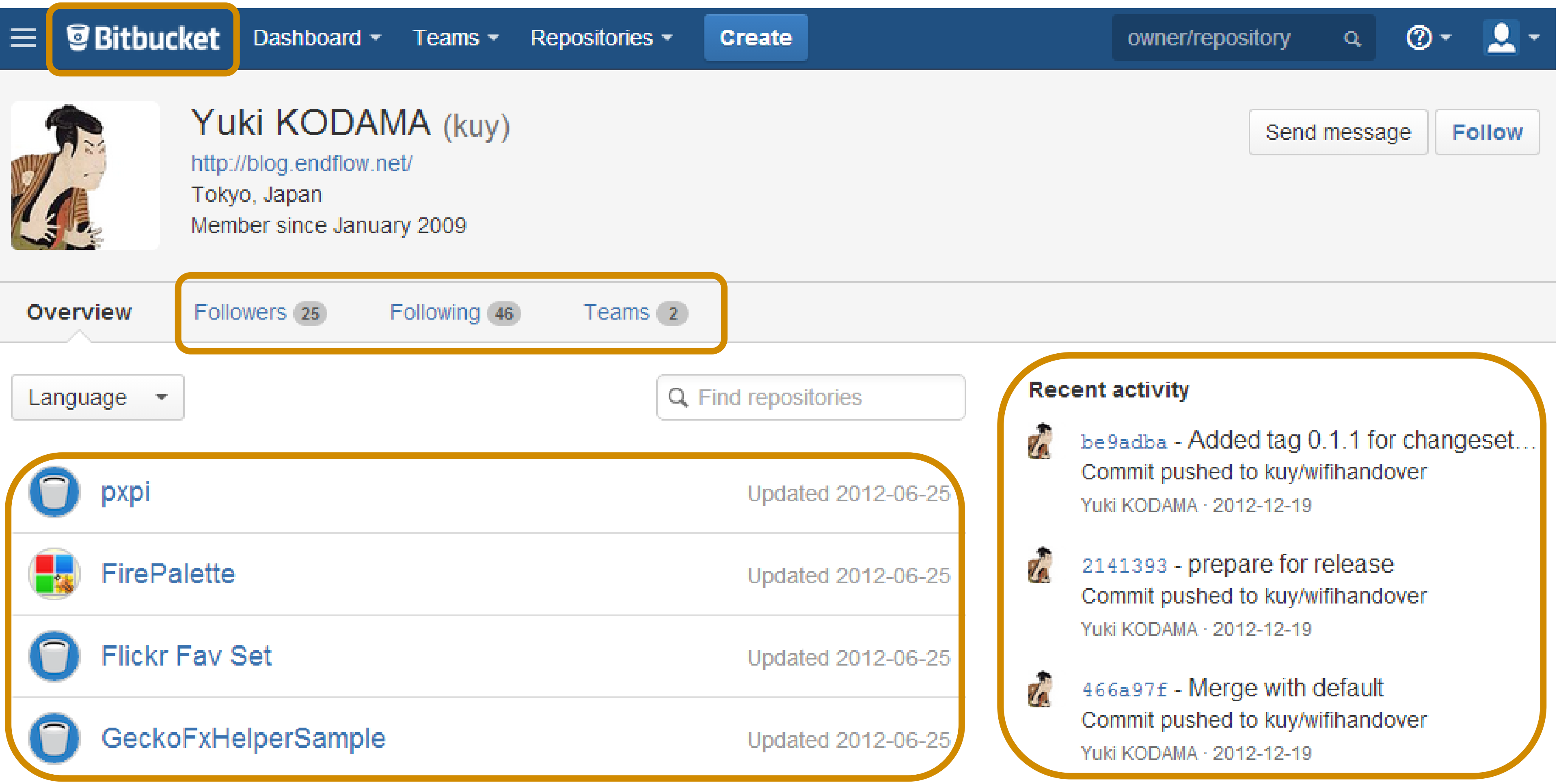
Updated a day ago

reddit-plugin-liveupdate JavaScript ★ 13 14

the code behind reddit live

Updated 2 days ago

Artifact: Developer Activities



Yuki KODAMA (kuy)

http://blog.endflow.net/

Tokyo, Japan

Member since January 2009

pxpi Updated 2012-06-25

FirePalette Updated 2012-06-25

Flickr Fav Set Updated 2012-06-25

GeckoFxHelperSample Updated 2012-06-25

Recent activity

be9adba - Added tag 0.1.1 for changeset…
Commit pushed to kuy/wifihandover
Yuki KODAMA · 2012-12-19

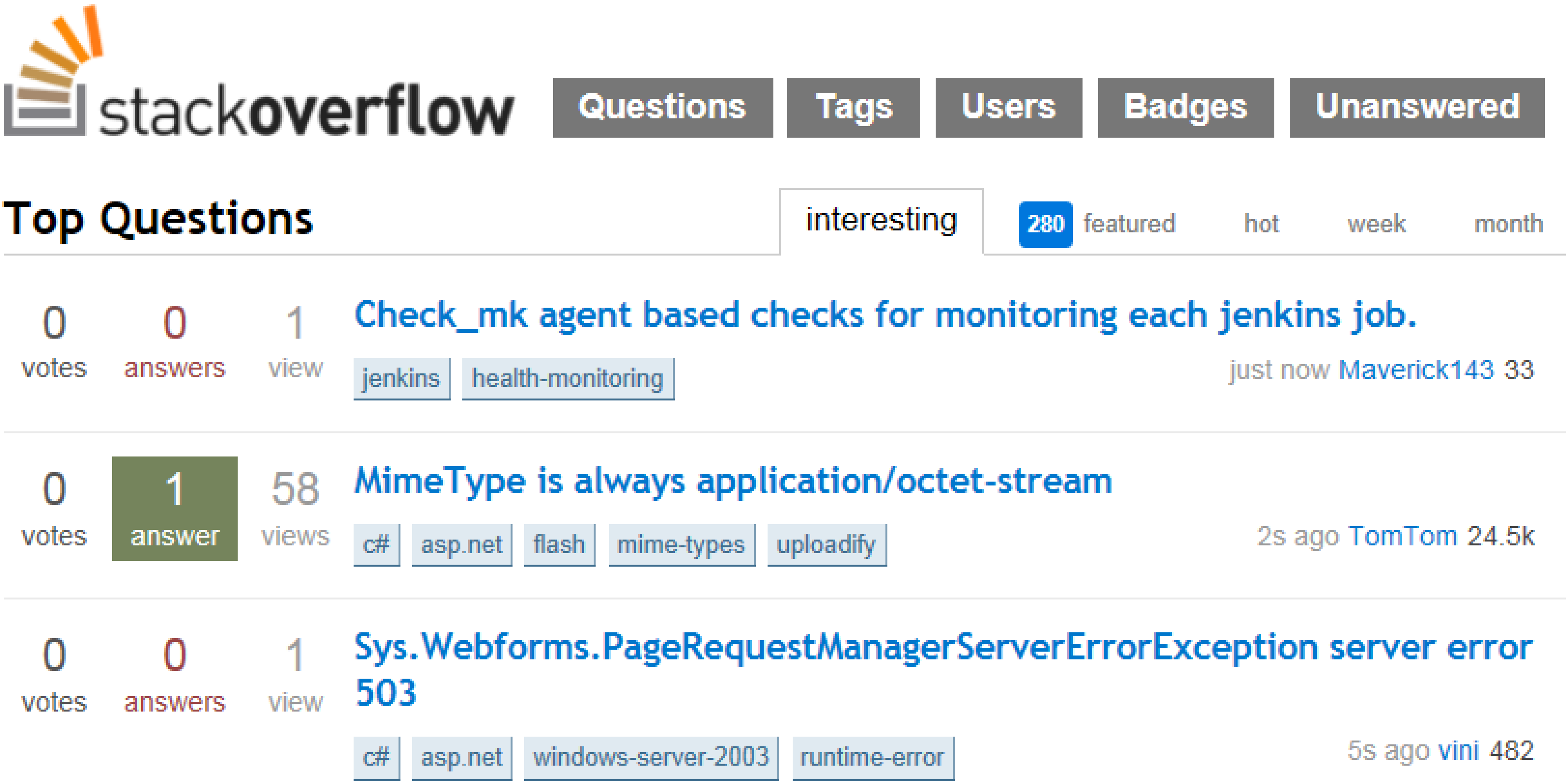
2141393 - prepare for release
Commit pushed to kuy/wifihandover
Yuki KODAMA · 2012-12-19

466a97f - Merge with default
Commit pushed to kuy/wifihandover
Yuki KODAMA · 2012-12-19

Artifact: Software Forums

- Developers ask and answer questions
- About various topics
- In various threads, some of which are very long
- Stored in various sites
  - StackOverflow: http://stackoverflow.com/
  - SoftwareTripsAndTricks: http://www.softwaretipsandtricks.com/forum/

Artifact: Software Forums



Top Questions

0 votes 0 answers 1 view

Check_mk agent based checks for monitoring each jenkins job.

jenkins health-monitoring

just now Maverick143 33

0 votes 1 answer 58 views

MimeType is always application/octet-stream

c# asp.net flash mime-types uploadify

2s ago TomTom 24.5k

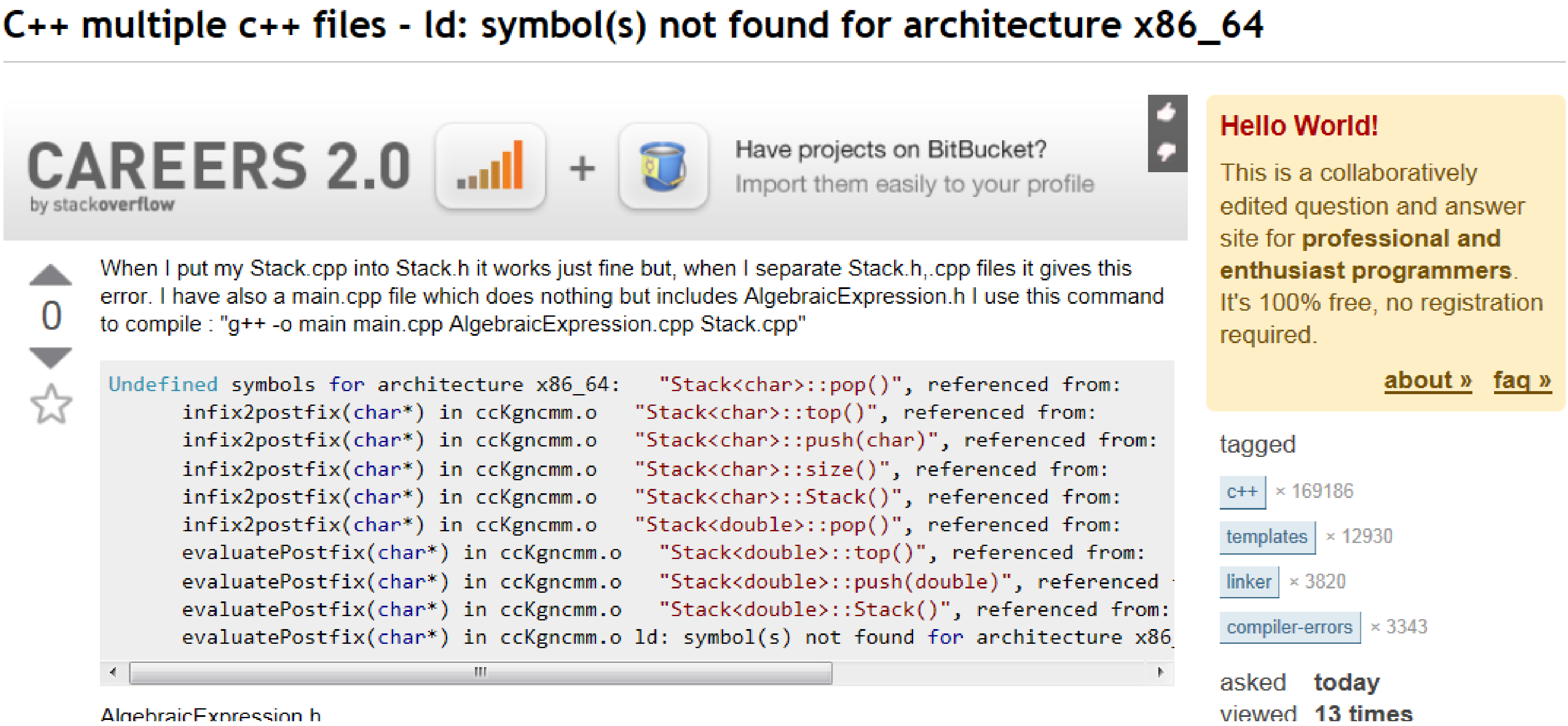
0 votes 0 answers 1 view

Sys.Webforms.PageRequestManagerServerErrorException server error 503

c# asp.net windows-server-2003 runtime-error

5s ago vini 482

Artifact: Software Forums



C++ multiple c++ files - ld: symbol(s) not found for architecture x86_64

When I put my Stack.cpp into Stack.h it works just fine but, when I separate Stack.h,.cpp files it gives this error. I have also a main.cpp file which does nothing but includes AlgebraicExpression.h I use this command to compile : "g++ -o main main.cpp AlgebraicExpression.cpp Stack.cpp"

```
Undefined symbols for architecture x86_64:   "Stack<char>::pop()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::top()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::push(char)", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::size()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::Stack()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<double>::pop()", referenced from:
      evaluatePostfix(char*) in ccKgncmm.o   "Stack<double>::top()", referenced from:
      evaluatePostfix(char*) in ccKgncmm.o   "Stack<double>::push(double)", referenced 
      evaluatePostfix(char*) in ccKgncmm.o   "Stack<double>::Stack()", referenced from:
      evaluatePostfix(char*) in ccKgncmm.o ld: symbol(s) not found for architecture x86
```

AlgebraicExpression.h

Hello World!

This is a collaboratively edited question and answer site for **professional and enthusiast programmers**. It's 100% free, no registration required.

about » faq »

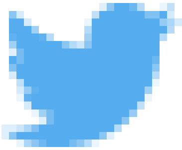
tagged

c++	× 169186
templates	× 12930
linker	× 3820
compiler-errors	× 3343

asked **today**
viewed **13 times**

Artifact: Software Microblogs

- Developers microblog too
- Developers microblog about various activities (Tian et al. 2012) :
    - Advertisements
    - Code and tools
    - News
    - Q&A
    - Events
    - Opinions
    - Tips
    - Etc.



**Tian et al.: What does software engineering community microblog about? MSR 2012: 247-250*

Artifact: Software Microblogs

Results for **#csharp**

Tweets Top / All

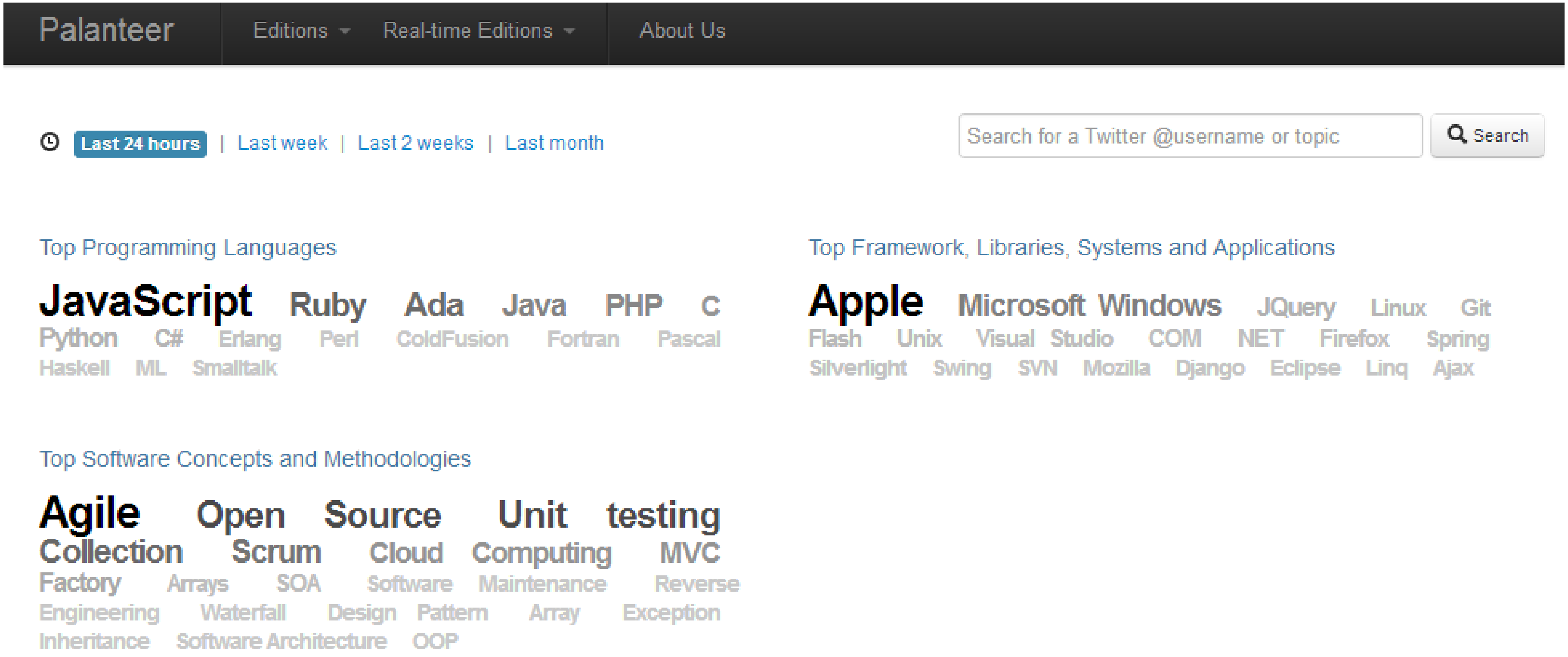
Kunal Chowdhury @kunal2383 1h
What's the difference between the "ref" and "out" keywords in C#? - shar.es/4aRyl #dotNet **#CSharp** #InterviewQuestion
Expand

Rozalia Mihailescu @rozaliamih 1h
Wow! 70% off Programming Courses w/ Paul Deitel: #iOS6, #Android, **#Csharp**, #Java, #Javascript, C++ ow.ly/gqrvU #udemy
Expand

Robert W. McBean @robertmcbean 1h
please join my linkedin network @ linkedin.com/in/robertwmcbe…
#csharp #it #software #yyc #web
Expand

Lucian Mihailescu @rointercer 1h
Wow! STILL ON! 70% off Programming Courses w/ Paul Deitel: #iOS6, #Android, **#Csharp**, #Java, #Javascript, C++ ow.ly/gv1pn #udemy
Expand

Artifact: Software Microblogs



Top Programming Languages

JavaScript Ruby Ada Java PHP C Python C# Erlang Perl ColdFusion Fortran Pascal Haskell ML Smalltalk

Top Framework, Libraries, Systems and Applications

Apple Microsoft Windows JQuery Linux Git Flash Unix Visual Studio COM NET Firefox Spring Silverlight Swing SVN Mozilla Django Eclipse Linq Ajax

Top Software Concepts and Methodologies

Agile Open Source Unit testing Collection Scrum Cloud Computing MVC Factory Arrays SOA Software Maintenance Reverse Engineering Waterfall Design Pattern Array Exception Inheritance Software Architecture OOP

http://research.larc.smu.edu.sg/palanteer/swdev

Part IV: Basic Program Analysis Tools

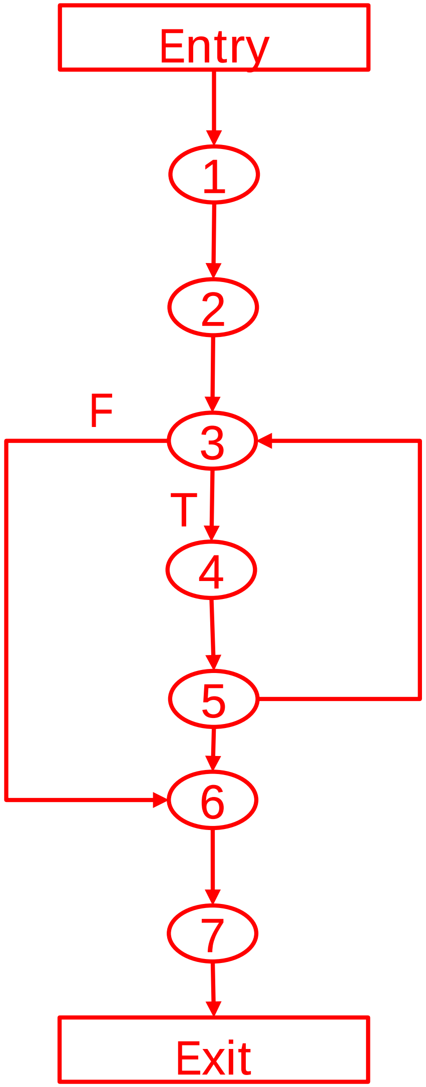
- Static Analysis
- Dynamic Analysis

Static Analysis

- Control Flow Graph Construction
- Program Dependence Graph Construction

Control Flows: Control-Flow Graphs

```
int main() {
 ①int sum = 0;
 ②int i = 1;
 ③while ( i < 11 ) {
 ④    sum = sum + i;
 ⑤    i = i + 1;
   }
 ⑥printf(“%d\n”, sum);
 ⑦printf(“%d\n”, i);
}
```



- Control flow is a relation that represents the *possible* flow of execution in a program.
    - (*a*, *b*) in the relation means that control can directly flow from element *a* to element *b* during execution.

Control Flows: Control Dependence

```
int main() {
 ①int sum = 0;
 ②int i = 1;
 ③while ( i < 11 ) {
 ④   sum = sum + i;
 ⑤   i = i + 1;
   }
 ⑥printf(“%d\n”, sum);
 ⑦printf(“%d\n”, i);
}
```

Entry node controls nodes 1,2,3,6,7
Node 3 controls nodes 4 and 5

- Given nodes C and N in a CFG, N is control-dependent on C if the outcome of C determines if N is reached in the CFG.
- We call C as a controller of N.

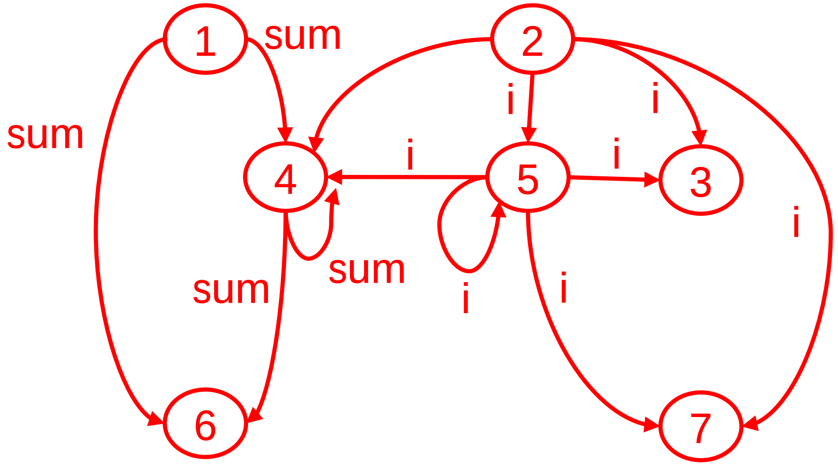
Data Flows: Definitions / Uses

```
int main() {
 ①int sum = 0;
 ②int i = 1;
 ③while ( i < 11 ) {
 ④   sum = sum + i;
 ⑤   i = i + 1;
   }
 ⑥printf(“%d\n”, sum);
 ⑦printf(“%d\n”, i);
}
```

- A definition-use chain or DU-chain, for a definition D of variable v, is:
  - the set of pair-wise connections
  - between D and all uses of v that D can reach.

Data Dependence Graphs

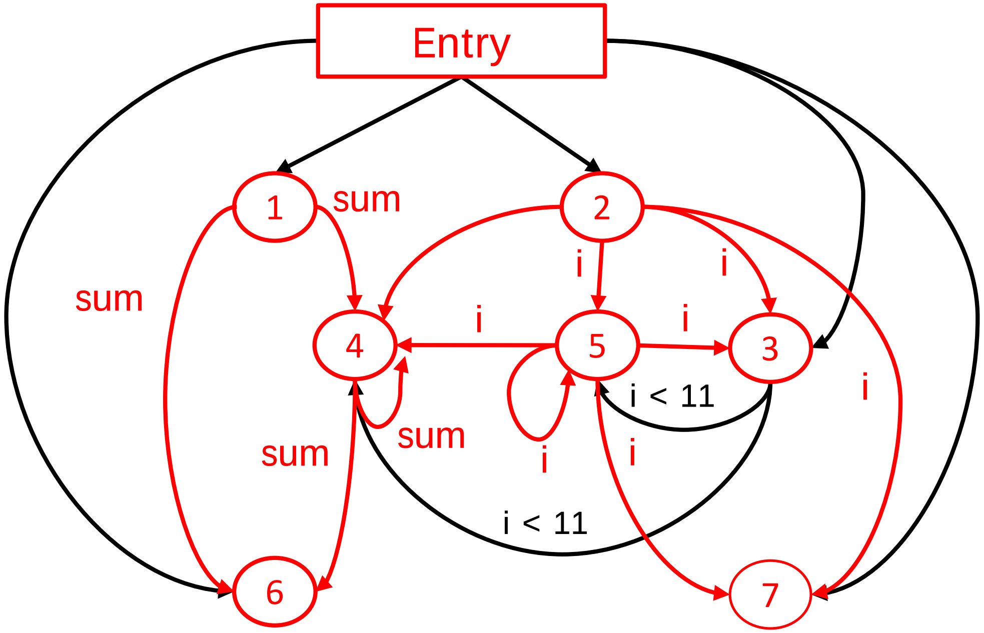
```
int main() {
 ①int sum = 0;
 ②int i = 1;
 ③while ( i < 11 ) {
 ④    sum = sum + i;
 ⑤    i = i + 1;
   }
 ⑥printf(“%d\n”, sum);
 ⑦printf(“%d\n”, i);
}
```



- A data-dependence graph contains:
    - one node for every program line (or an instruction, or a basic block, or a desired granularity) and
    - labelled edges that correspond to DU-chains.

Program/Procedural Dependence Graphs

```
int main() {
 ①int sum = 0;
 ②int i = 1;
 ③while ( i < 11 ) {
 ④    sum = sum + i;
 ⑤    i = i + 1;
   }
 ⑥printf(“%d\n”, sum);
 ⑦printf(“%d\n”, i);
}
```



- PDGs are control- and data-dependence graphs
    - Capture “semantics”
    - Expose parallelism
    - Facilitate debugging

Tools

- Program analysis platforms
  - WALA: http://wala.sourceforge.net/wiki/index.php/Main_Page
  - Chord: http://code.google.com/p/jchord/
  - ROSE: http://www.rosecompiler.org/
- Other tools
  - JPF: http://babelfish.arc.nasa.gov/trac/jpf
  - BLAST: http://mtc.epfl.ch/software-tools/blast/index-epfl.php
  - ESC/Java: http://kindsoftware.com/products/opensource/ESCJava2/
  - SPIN: http://spinroot.com/spin/whatispin.html
  - PAT: http://www.comp.nus.edu.sg/~pat/
  - Choco: http://www.emn.fr/z-info/choco-solver/
  - Yices: http://yices.csl.sri.com/
  - STP: https://sites.google.com/site/stpfastprover/STP-Fast-Prover
  - Z3: http://z3.codeplex.com/

Dynamic Analysis

- Instrumentation
- Test Case Generation

What is Dynamic (Program) Analysis?

- Basically
  - Run a program
  - Monitor program states during/after the executions
  - Extract useful information/properties about the program

How to Run?

- Testing
- Choose good test cases
  - The test suite determines the expense
    - In time and space
  - The test suite determines the accuracy
    - What executions are seen or not seen

Tracing / Profiling

- Tracing: Record faithfully (lossless) detailed information of program executions
  - Control flow tracing
    - Sequence of executed statements.
  - Dependence tracing
    - Sequence of exercised dependences.
  - Value tracing
    - Sequence of values produced by each instruction.
  - Memory access tracing
    - Sequence of memory references during an execution
- Profiling: Record aggregated (lossy) information about program executions
  - Control flow profiling: execution frequencies of instructions
  - Value profiling: occurrence frequencies of values

Tracing/Profiling by Instrumentation

- Source code instrumentation
- Binary instrumentation

Test Case Generation: Concolic Testing

- Goal: find actual inputs that exhibit an error or execute as many program elements as possible
  - By exploring different execution paths

1) Start with an execution with random inputs
2) Collect the path conditions for the execution
3) Negate some of the path conditions
   - So as to be used as the path conditions for the next execution which should follow a different path
4) Solve the new path conditions to get actual values for the inputs
   - The execution using these new inputs should follow a different path
5) Repeat 2)—4) until no more new paths to explore

**Godefroid et al.: DART: directed automated random testing.*
PLDI 2005: 213-223
**Sen et al.: CUTE: a concolic unit testing engine for C.*
ESEC/SIGSOFT FSE 2005: 263-272

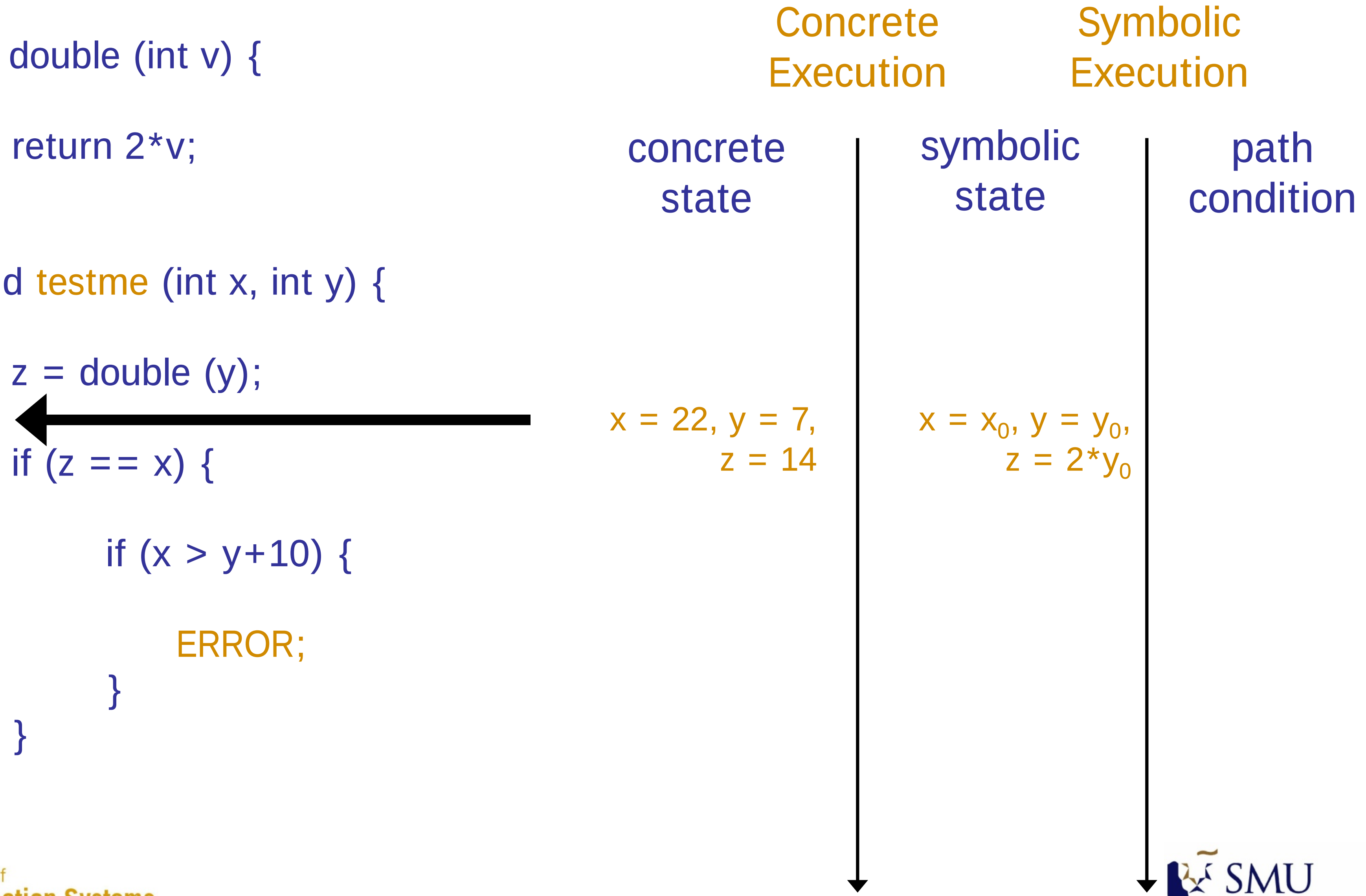
Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 22, y = 7	$x = x_0$, $y = y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```



Concrete Execution

Symbolic Execution

concrete state	symbolic state	path condition
x = 22, y = 7, z = 14	$x = x_0$, $y = y_0$, $z = 2*y_0$	

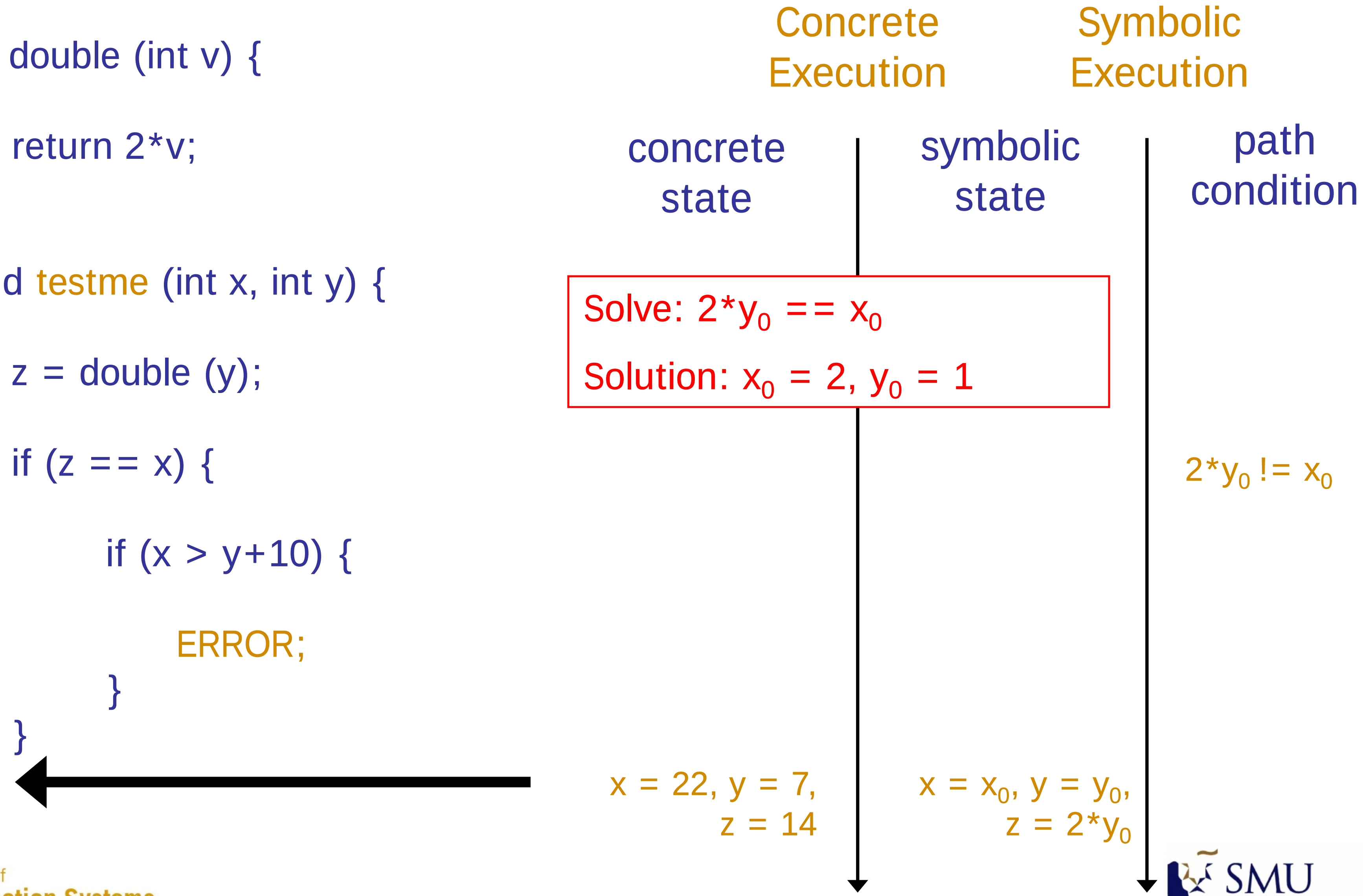
Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
		$2*y_0 != x_0$
x = 22, y = 7, z = 14	$x = x_0$, $y = y_0$, $z = 2*y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```



Concrete Execution

Symbolic Execution

Solve: $2*y_0 == x_0$

Solution: $x_0 = 2$, $y_0 = 1$

$2*y_0 != x_0$

$x = 22, y = 7, z = 14$

$x = x_0, y = y_0, z = 2*y_0$

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2, y = 1	x = x_0, y = y_0	

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 2, y = 1, z = 2	$x = x_0$, $y = y_0$, $z = 2*y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution

Symbolic Execution

concrete state	symbolic state	path condition
		$2*y_0 == x_0$
$x = 2, y = 1, z = 2$	$x = x_0, y = y_0, z = 2*y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {

    return 2*v;
}

void testme (int x, int y) {

    z = double (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

$2*y_0 == x_0$

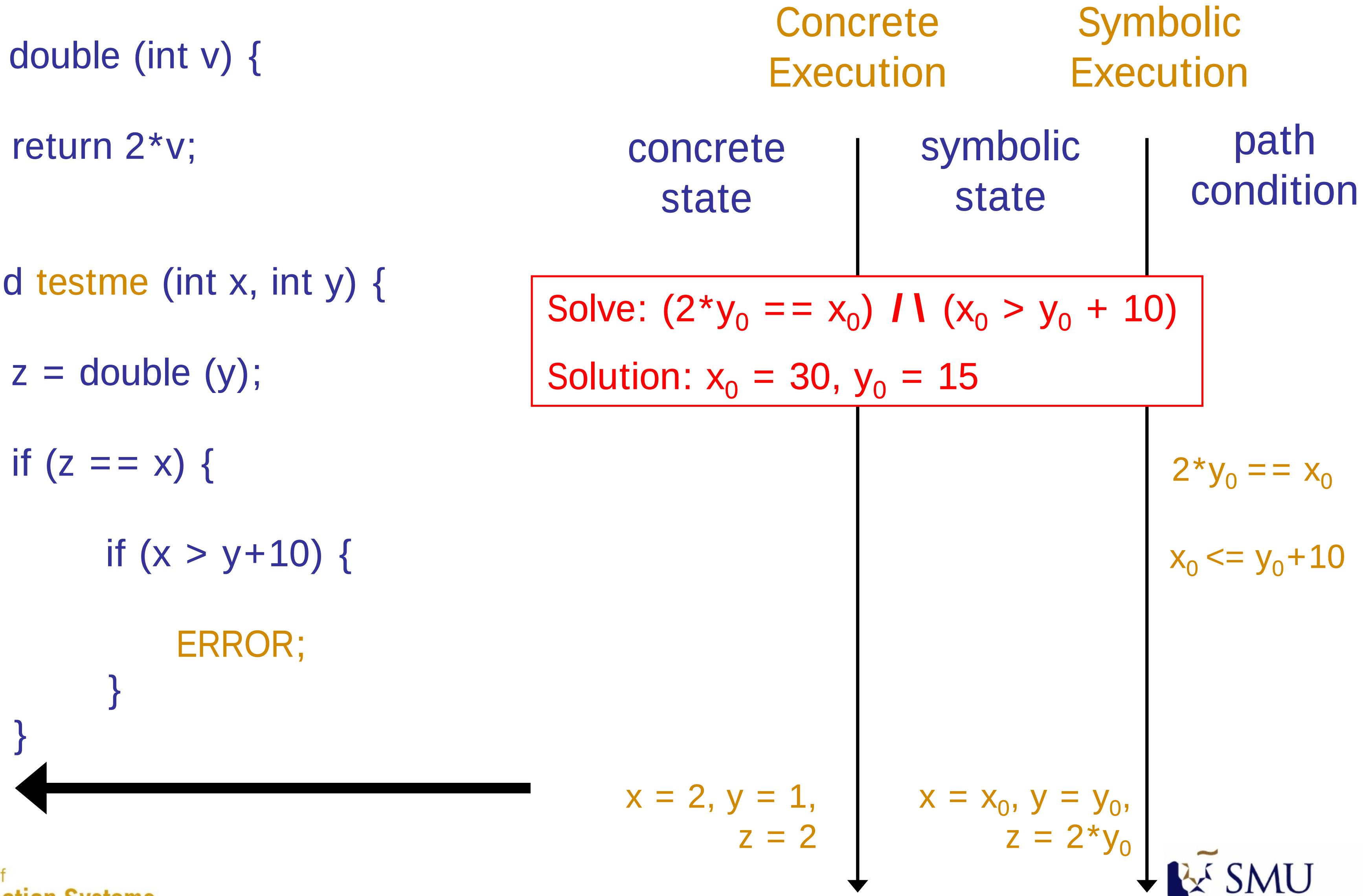
$x_0 <= y_0+10$

x = 2, y = 1, z = 2

$x = x_0$, $y = y_0$, $z = 2*y_0$

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }

}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $(2*y_0 == x_0)$ **/** $(x_0 > y_0 + 10)$

Solution: $x_0 = 30$, $y_0 = 15$

$2*y_0 == x_0$

$x_0 <= y_0+10$

x = 2, y = 1,
z = 2

$x = x_0$, $y = y_0$,
$z = 2*y_0$

School of Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

	Concrete Execution	Symbolic Execution	
	concrete state	symbolic state	path condition
←	x = 30, y = 15	$x = x_0, y = y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int double (int v) {
    return 2*v;
}

void testme (int x, int y) {
    z = double (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Program Error

x = 30, y = 15

$x = x_0, y = y_0$

$2*y_0 == x_0$

$x_0 > y_0+10$

Concolic Testing: Symbolic & Concrete Executions

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

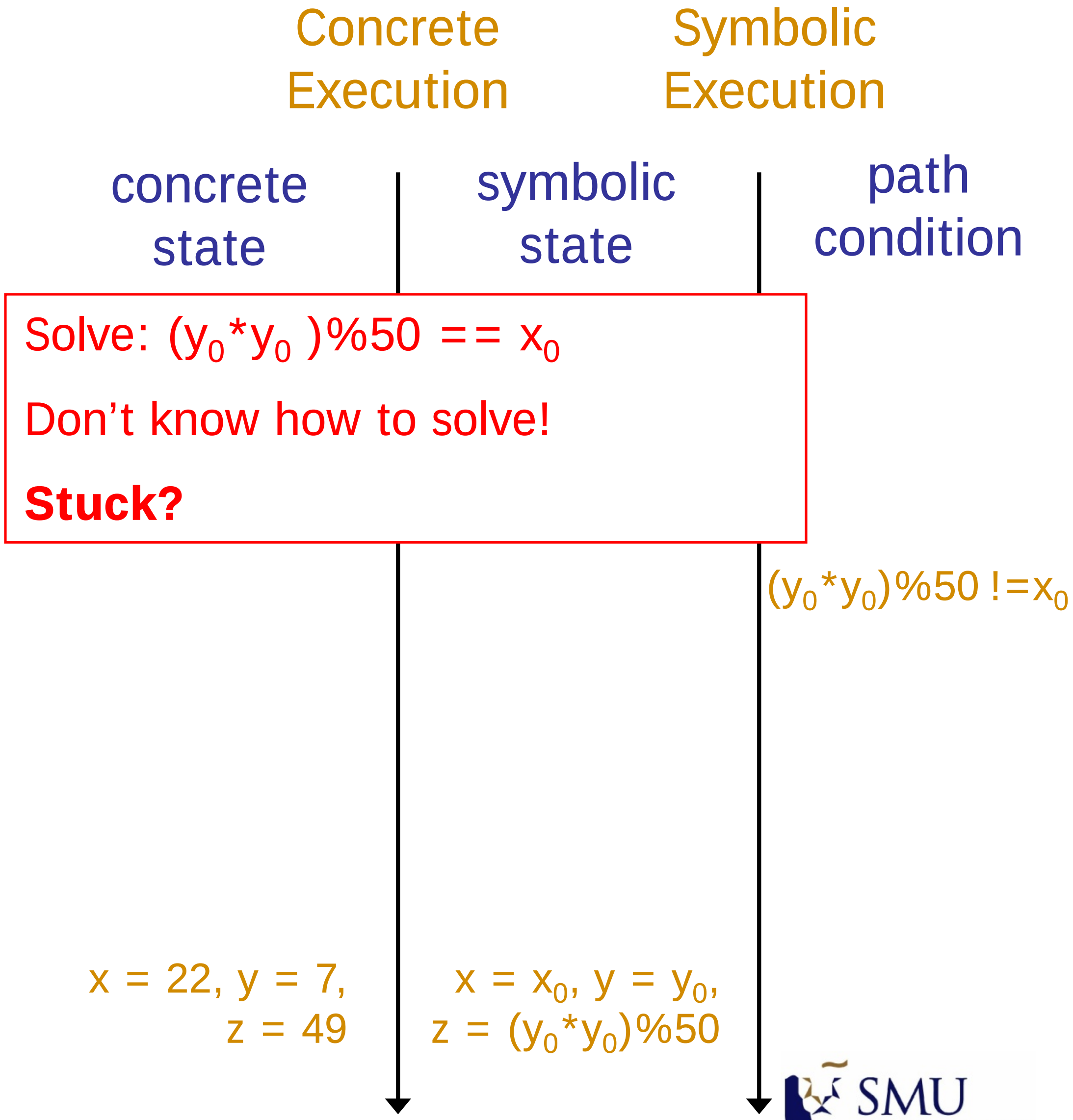
Concrete Execution

Symbolic Execution

concrete state	symbolic state	path condition
x = 22, y = 7	$x = x_0, y = y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```



Solve: $(y_0*y_0)\%50 == x_0$

Don’t know how to solve!

Stuck?

$(y_0*y_0)\%50 != x_0$

$x = 22, y = 7,$
$z = 49$

$x = x_0, y = y_0,$
$z = (y_0*y_0)\%50$

Concolic Testing: Symbolic & Concrete Executions

Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

```
void testme (int x, int y) {

     z = foo (y);

     if (z == x) {

          if (x > y+10) {

               ERROR;
          }
     }

}
```

Solve: foo (y_0) == x_0

Don't know how to solve!

Stuck?

foo (y_0) !=x_0

x = 22, y = 7,
z = 49

x = x_0, y = y_0,
z = (y_0*y_0)%50

Concolic Testing: Symbolic & Concrete Executions

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```

Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $(y_0 * y_0)\%50 == x_0$

Don't know how to solve!

Not Stuck!

Use concrete state

Replace y_0 by 7

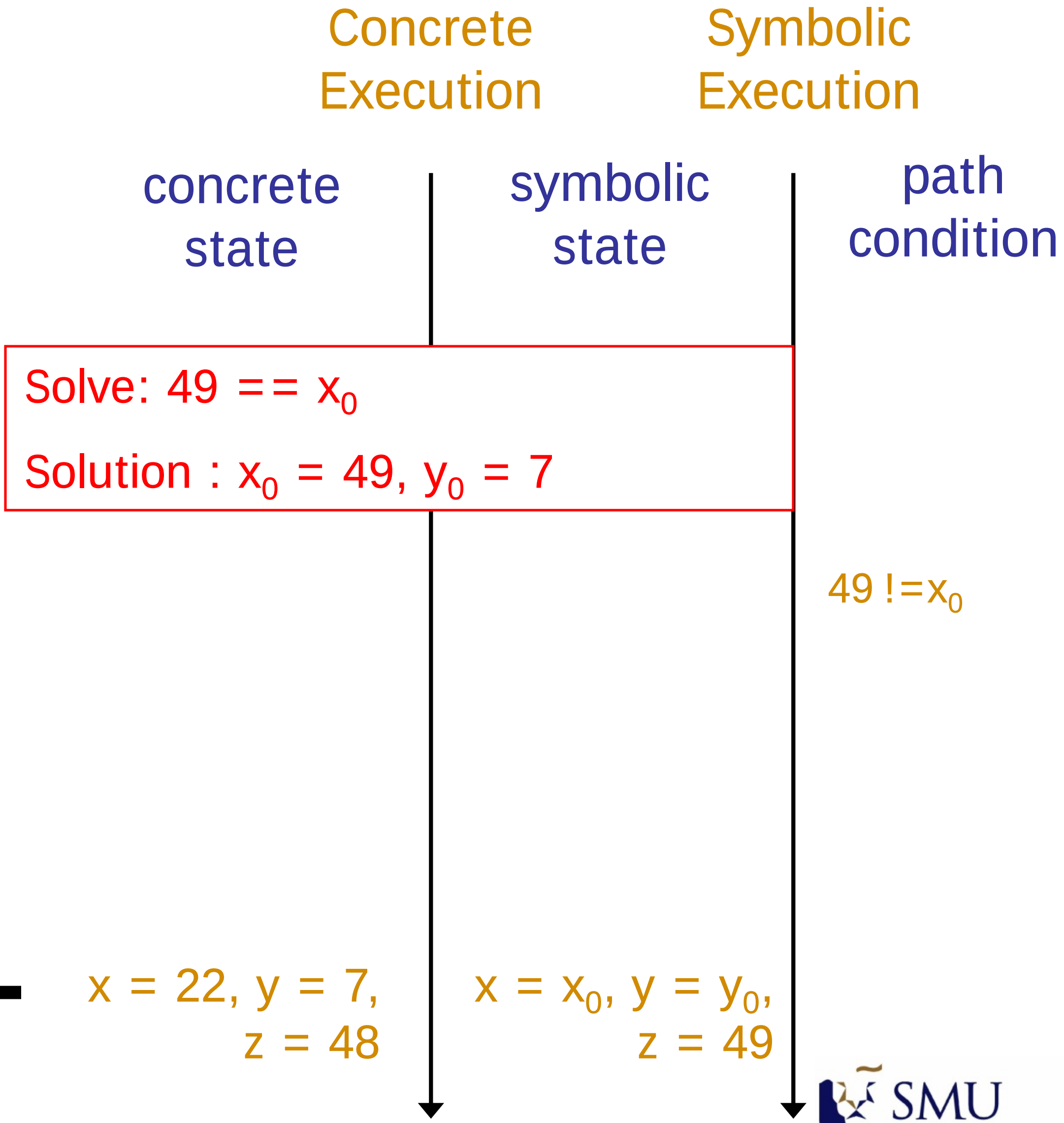
$(y_0 * y_0)\%50 != x_0$

$x = 22, y = 7, z = 49$

$x = x_0, y = y_0, z = (y_0 * y_0)\%50$

Concolic Testing: Symbolic & Concrete Executions

```
int foo (int v) {

    return (v*v) % 50;
}

void testme (int x, int y) {

    z = foo (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```



Concrete Execution

Symbolic Execution

concrete state

symbolic state

path condition

Solve: $49 == x_0$

Solution : $x_0 = 49$, $y_0 = 7$

$49 != x_0$

$x = 22$, $y = 7$, $z = 48$

$x = x_0$, $y = y_0$, $z = 49$

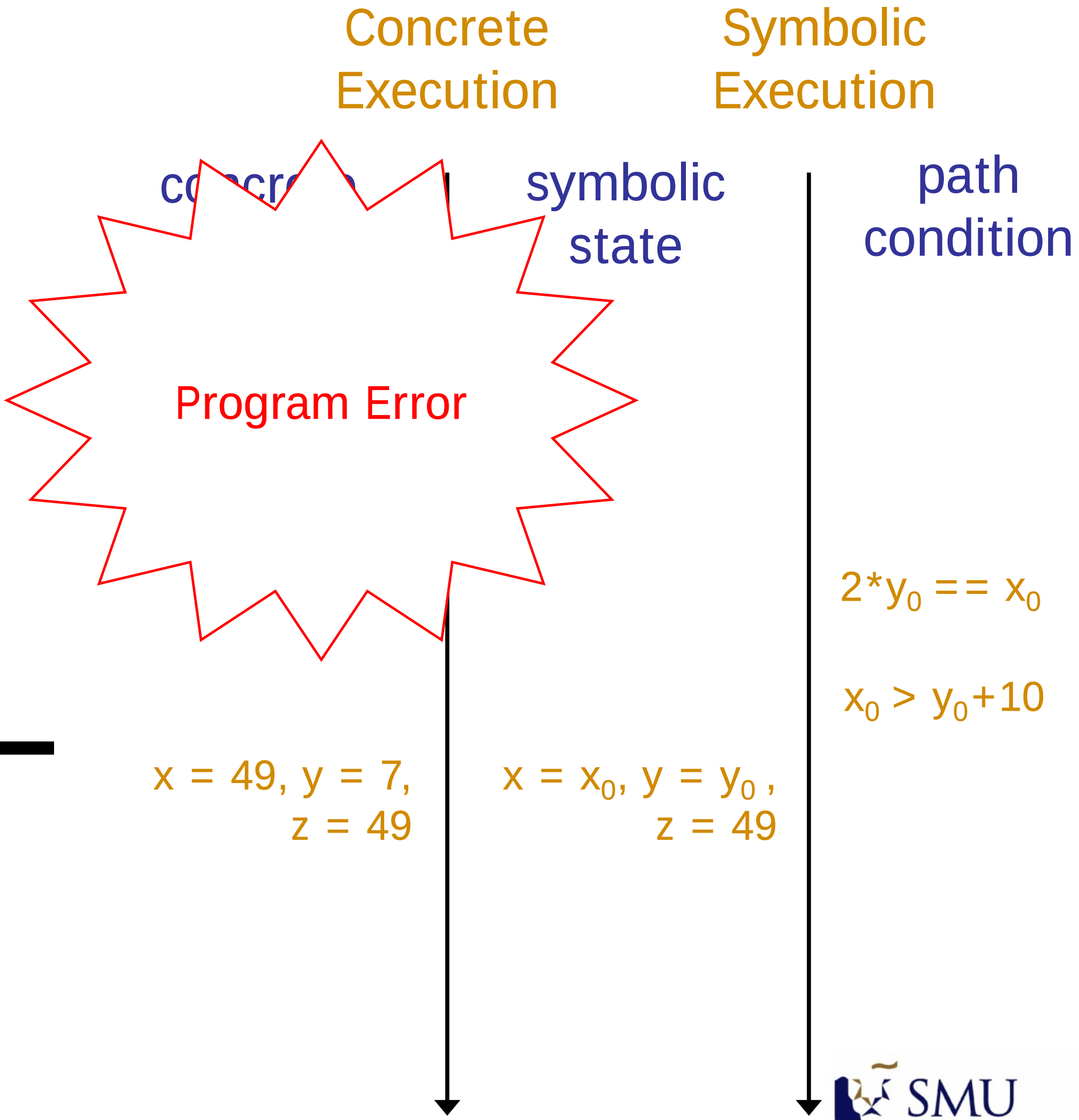
Concolic Testing: Symbolic & Concrete Executions

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {

    z = foo (y);

    if (z == x) {

        if (x > y+10) {

            ERROR;
        }
    }

}
```

Concrete Execution	Symbolic Execution	
concrete state	symbolic state	path condition
x = 49, y = 7	$x = x_0, y = y_0$	

Concolic Testing: Symbolic & Concrete Executions

```
int foo (int v) {
    return (v*v) % 50;
}

void testme (int x, int y) {
    z = foo (y);
    if (z == x) {
        if (x > y+10) {
            ERROR;
        }
    }
}
```



Concrete Execution		Symbolic Execution
c[illegible]cr[illegible]	symbolic state	path condition
		$2*y_0 == x_0$
		$x_0 > y_0+10$
x = 49, y = 7, z = 49	$x = x_0$, $y = y_0$, z = 49	

Program Error

Tools

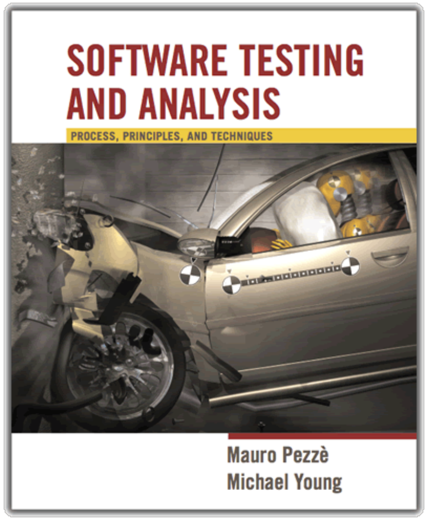
- Program instrumentation and analysis frameworks
  - CIL: http://cil.sourceforge.net/
  - Valgrind: http://valgrind.org/
  - Daikon: http://groups.csail.mit.edu/pag/daikon/
  - Jikes: http://jikes.sourceforge.net/
  - QEMU: http://wiki.qemu.org/Main_Page
  - Pin: http://www.pintool.org/
  - Omega: http://www.cs.umd.edu/projects/omega/
- Test case generation
  - Korat: http://korat.sourceforge.net/
  - CUTE: http://srl.cs.berkeley.edu/~ksen/doku.php
  - CREST: http://crest.googlecode.com/

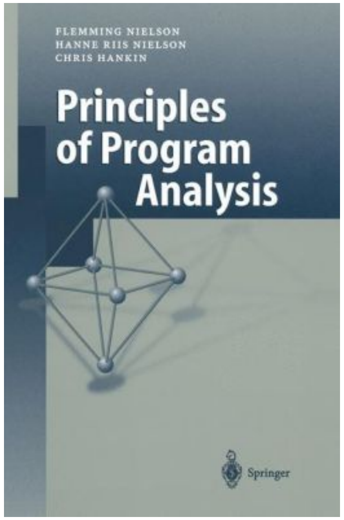
Conclusion

- Part I: Challenges & Problems
  - High software cost
  - Ensuring reliability of systems
- Part II: Research Topics
  - Software testing and reliability
  - Software maintenance and evolution
  - Software analytics
- Part III: Data Sources
  - Code, traces, history, bug reports, developer activities, forums, microblogs, etc.
- Part IV: Basic Program Analysis Tools
  - Static analysis
  - Dynamic analysis

Additional References & Acknowledgements

- Some slides and images are taken or adapted from:
  - Ying Zou’s, Ahmed Hassan’s and Tao Xie’s slides
  - Lingxiao Jiang’s slides
    (from SMU’s IS706 slides that we co-taught together)
  - Mauro Pezze’s and Michal Young’s slides
    (from the resource slides of their book)





Thank you!

Questions? Comments?

davidlo@smu.edu.sg

Data Mining for Software Engineering

Genius is 1% inspiration and 99% perspiration!

-Thomas A. Edison

Slide Outline

- Part I: Pattern Mining
  - Techniques
  - Applications
- Part II: Clustering
  - Techniques
  - Applications
- Part III: Classification
  - Techniques
  - Applications

Part I: Pattern Mining

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs many times in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?

Structure

- Techniques
  - Association Rule Mining
  - Sequential Pattern Mining
  - Subgraph Mining
- Applications
  - Allatin: Mining Alternative Patterns
  - Other Applications

I(A): Pattern Mining Techniques

Association Rule Mining

Definition: Frequent Itemsets

- Frequent pattern mining: find all frequent itemsets in a database

- Itemset: a set of items
  - E.g., acm={a, c, m}
- Support of itemsets
  - Sup(acm)=3
- Given min_sup = 3, acm is a frequent pattern

Transaction database TDB

TID	Items bought
100	f, a, c, d, g, i, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

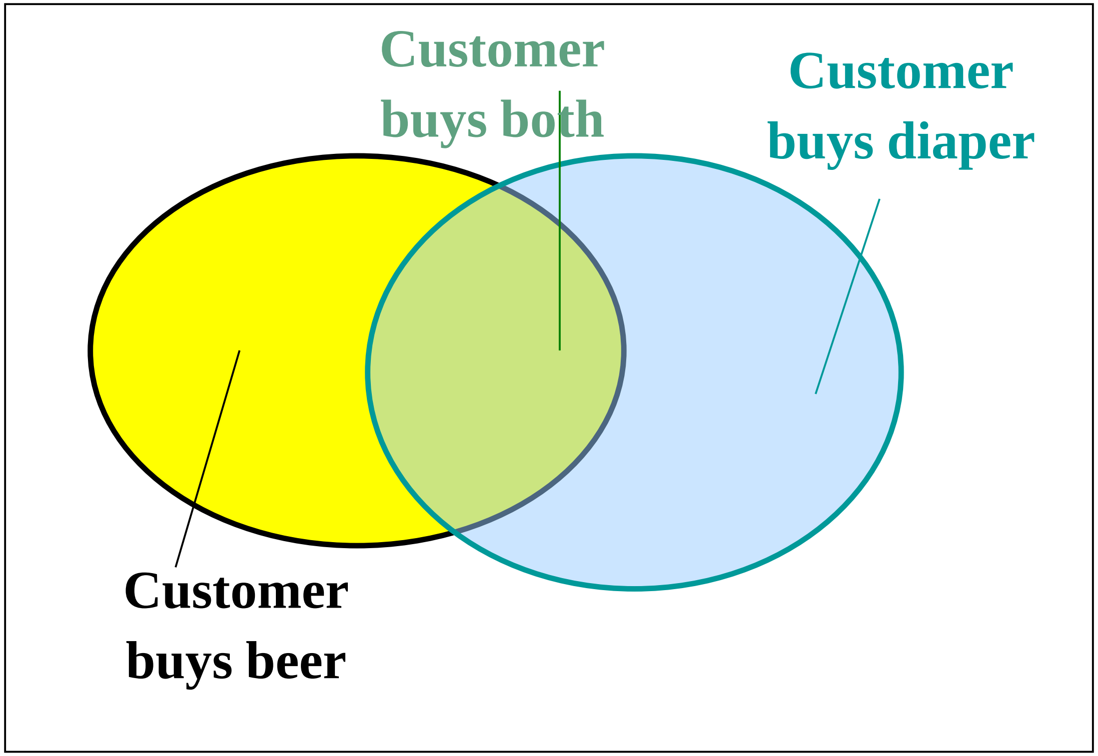
Definition: Association Rules

- Find all the rules X → Y with minimum support and confidence
    - support, s, number of transactions contain $X \cup Y$
    - confidence, c, conditional probability that a transaction having X also contains Y

- Itemsets should be frequent
    - It can be applied extensively
- Rules should be confident
    - With strong prediction capability

Definition: Association Rules

- buy(diaper) → buy(beer)
  - Dads taking care of babies in weekends drink beer



Definition: Association Rules

Transaction-id	Items bought
10	A, B, D
20	A, C, D
30	A, D, E
40	B, E, F
50	B, C, D, E, F

- *Let min-sup = 3, min-conf = 50%*
- *Freq. Pat.:* {*A:3, B:3, D:4, E:3, AD:3*}
- Association rules:
  - *A* → *D* (3, 100%)
  - *D* → *A* (3, 75%)

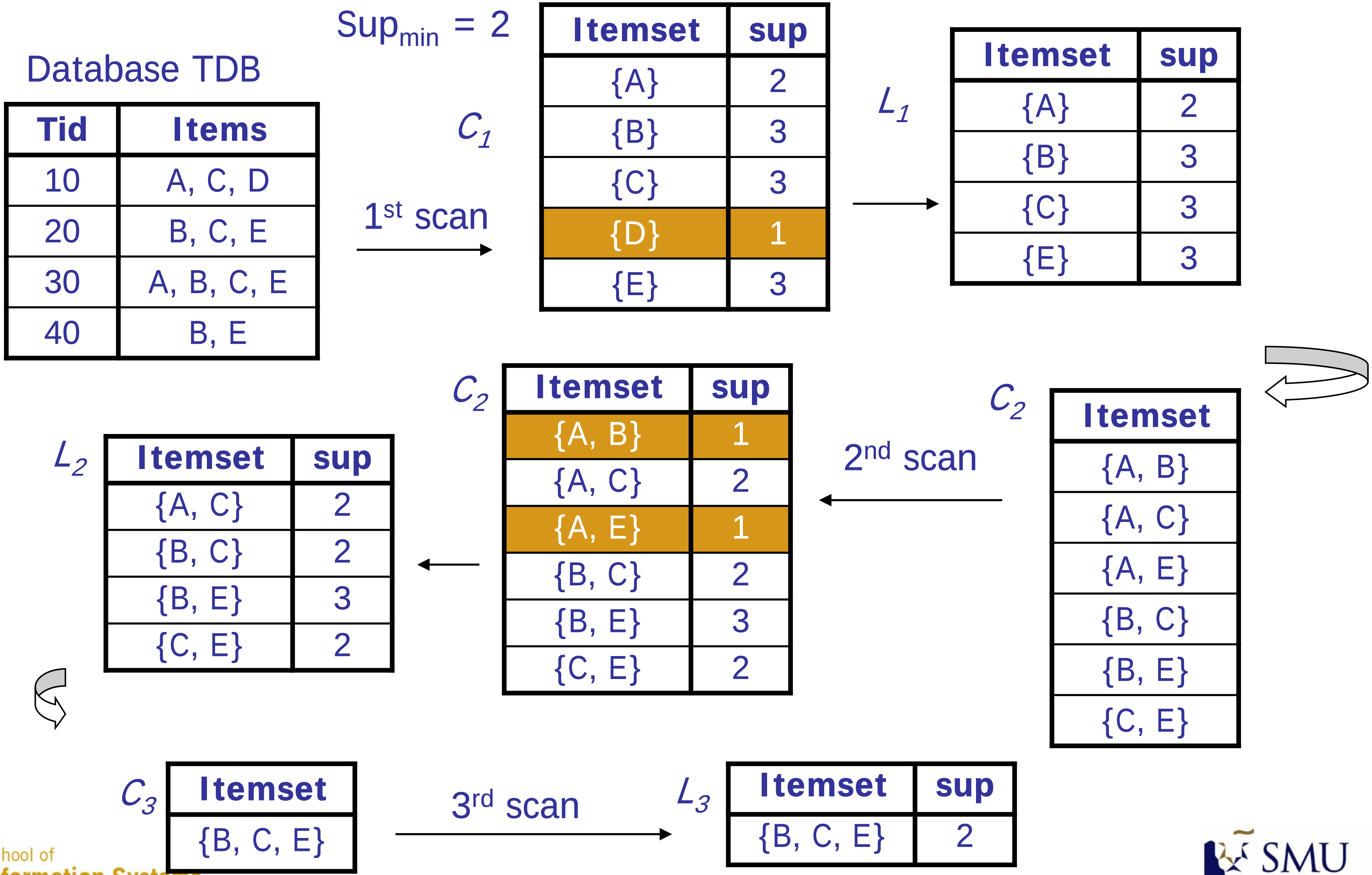
Methodology

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If {beer, diaper, nuts} is frequent, so is {beer, diaper}
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods:
  - Apriori (Agrawal & Srikant@VLDB’94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD’00)
  - Vertical data format approach (Charm—Zaki & Hsiao @SDM’02)

Methodology: Apriori Algorithm

- Apriori pruning principle:
  - If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

Apriori Algorithm—An Example



$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

Apriori Algorithm

- Pseudo-code:

```
$C_k$: Candidate itemset of size k
$L_k$ : frequent itemset of size k

$L_1$ = {frequent items};
for ($k$ = 1; $L_k$ !=$\emptyset$; $k$++) do begin
    $C_{k+1}$ = candidates generated from $L_k$;
    for each transaction $t$ in database do
        increment the count of all candidates in $C_{k+1}$
        that are contained in $t$
    $L_{k+1}$ = candidates in $C_{k+1}$ with min_support
    end
return $\cup_k L_k$;
```

Apriori Algorithm: Details

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of Candidate-generation
  - $L_3=\{abc, abd, acd, ace, bcd\}$
  - Self-joining: $L_3 * L_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - $C_4=\{abcd\}$

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ɔ X, *with the same support* as X (proposed by Pasquier, et al. @ ICDT'99)
  - Closed pattern is a lossless compression of freq. patterns
    - Reducing the # of patterns and rules
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ɔ X (proposed by Bayardo @ SIGMOD'98)

Closed Patterns and Max-Patterns

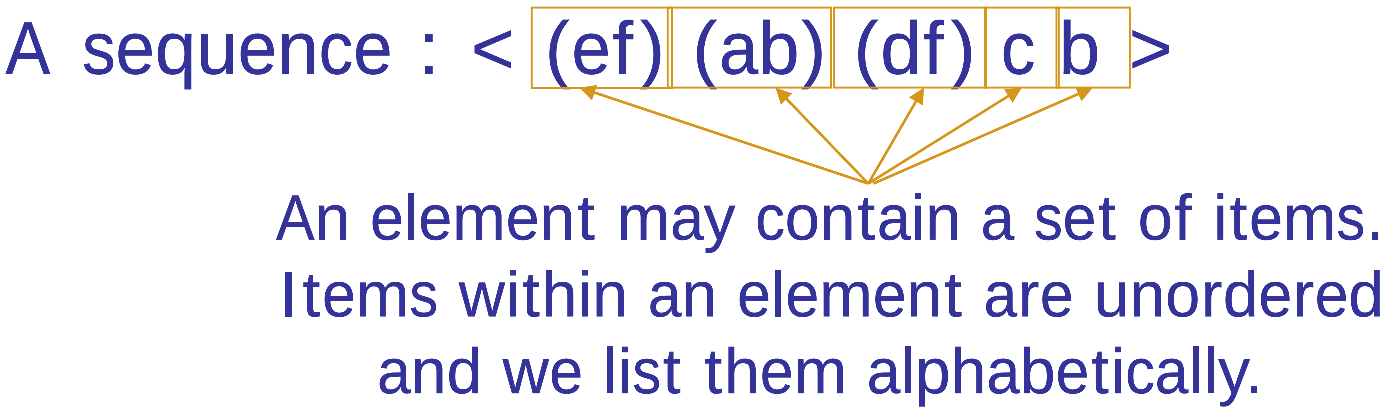
- Exercise. DB = $\{a_1, \ldots, a_{100}\}, \{a_1, \ldots, a_{50}\}$
  - Min_sup = 1.
- What is the set of closed itemset?
  - $\{a_1, \ldots, a_{100}\}$: 1
  - $\{a_1, \ldots, a_{50}\}$: 2
- What is the set of max-pattern?
  - $\{a_1, \ldots, a_{100}\}$: 1
- What is the set of all patterns?
  - !!

I(A): Pattern Mining Techniques

Sequential Pattern Mining

Definition: Sequential Pattern Mining

- Given a set of sequences, find the complete set of *frequent* subsequences



A sequence : < (ef) (ab) (df) c b >

An element may contain a set of items.
Items within an element are unordered
and we list them alphabetically.

Definition: Sequential Pattern Mining

A sequence database

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

Given support threshold
min_sup =2,
<(ab)c> is a frequent
sequential pattern

<(ab)c>
is a subsequence of
<a(abc)(ac)d(cf)>

<(ab)c>
is a subsequence of
<(ef)(ab)(df)cb>

Methodology

- A basic property: Apriori (Agrawal & Srikant'94)
    - If a sequence S is not frequent
    - Then none of the super-sequences of S is frequent
    - E.g, <hb> is infrequent → so do <hab> and <(ah)b>
- Many algorithms:
    - Apriori (Agrawal & Srikant'94): GSP
    - PrefixSpan
    - BIDE, etc.

GSP—Generalized Sequential Pattern Mining

- Proposed by Agrawal and Srikant, EDBT'96
- Outline of the method
  - Initially, every item in DB is a candidate of length-1
  - For each level (i.e., sequences of length-k) do
    - Scan database to collect support count for each candidate sequence
    - Generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori
  - Repeat until no frequent sequence or no candidate can be found

PrefixSpan: Definition

- Prefix and Suffix (Projection)
  - <a>, <aa>, <a(ab)> and <a(abc)> are prefixes of sequence <a(abc)(ac)d(cf)>
  - Given sequence <a(abc)(ac)d(cf)>

Prefix	Suffix (Prefix-Based Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

PrefixSpan: Approach

- Step 1: find length-1 sequential patterns
  - <a>, <b>, <c>, <d>, <e>, <f>

- Step 2: divide search space. The complete set of seq. pat. can be partitioned into 6 subsets:
  - The ones having prefix <a>;
  - The ones having prefix <b>;
  - …
  - The ones having prefix <f>

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

PrefixSpan: Approach

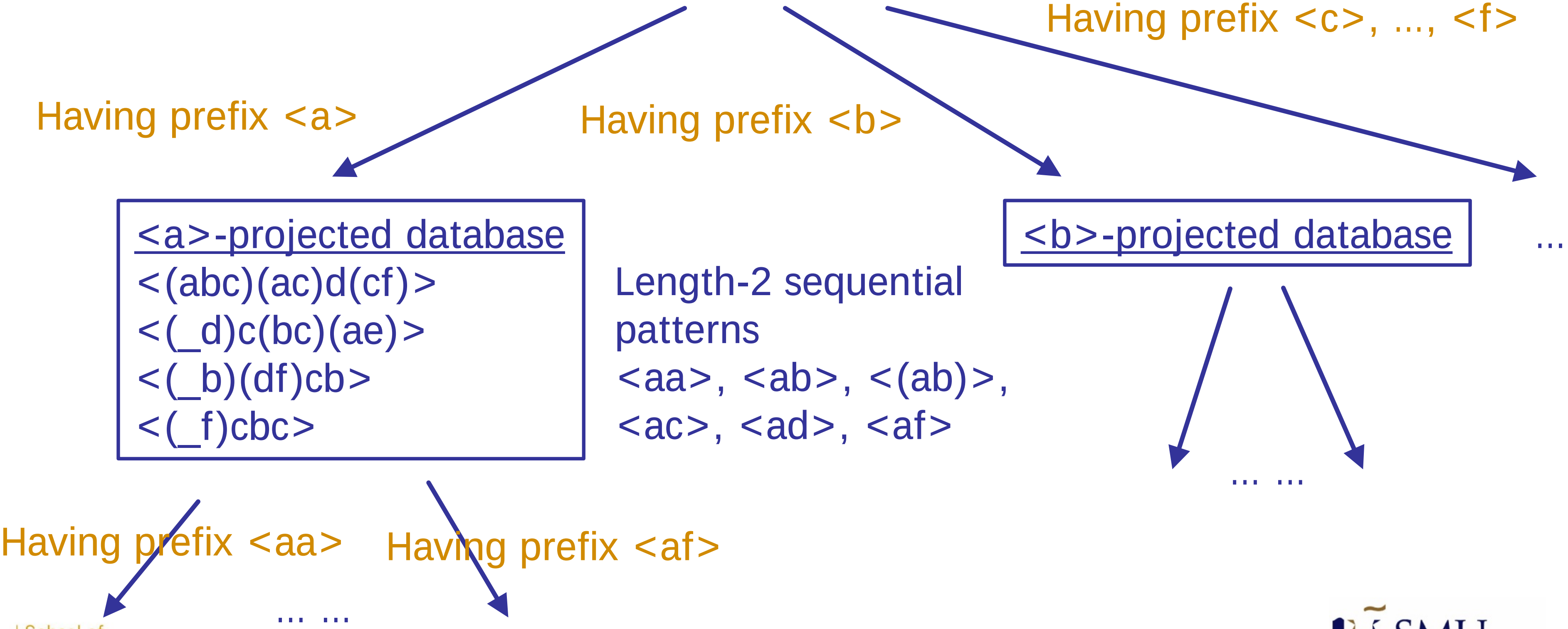
- Only need to consider projections w.r.t. <a>
  - <a>-projected database: <(abc)(ac)d(cf)>, <(_d)c(bc)(ae)>, <(_b)(df)cb>, <(_f)cbc>

- Find all the length-2 seq. pat. having prefix <a>: <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>
  - Further partition into 6 subsets
    - Having prefix <aa>;
    - ...
    - Having prefix <af>

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

PrefixSpan: Approach

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>



Having prefix <a>

Having prefix <b>

Having prefix <c>, ..., <f>

<a>-projected database
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

Length-2 sequential patterns
<aa>, <ab>, <(ab)>,
<ac>, <ad>, <af>

<b>-projected database

…

… …

Having prefix <aa>

Having prefix <af>

… …

Closed Frequent Sequences

- Motivation: Handling sequential pattern explosion problem
- Closed frequent sequence
  - A frequent (sub) sequence S is closed if there exists no supersequence of S that carries the same support as S
  - If some of S’s subsequences have the same support, it is unnecessary to output these subsequences (nonclosed sequences)
  - Lossless compression: still ensures that the mining result is complete

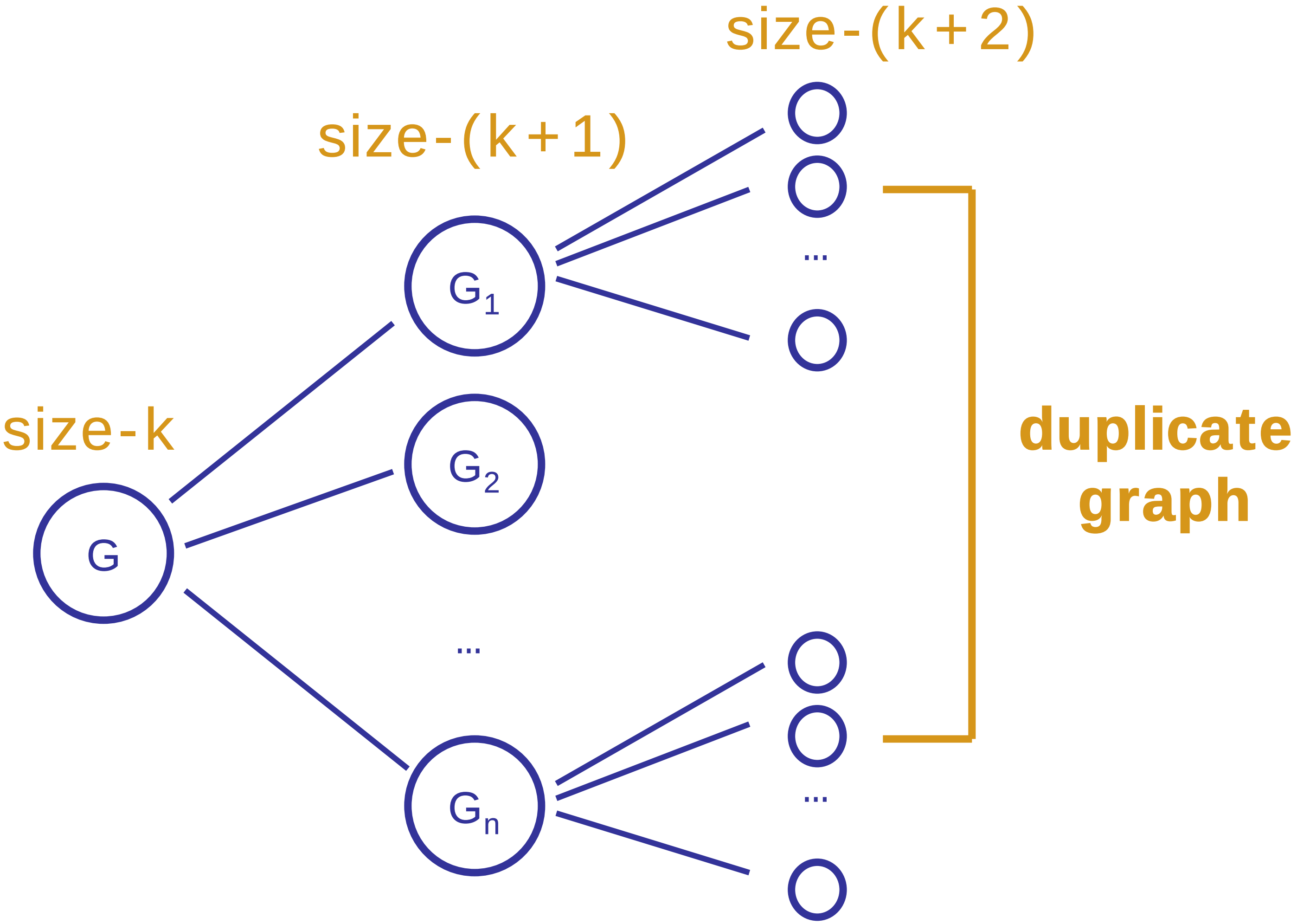
I(A): Pattern Mining Techniques

Subgraph Mining

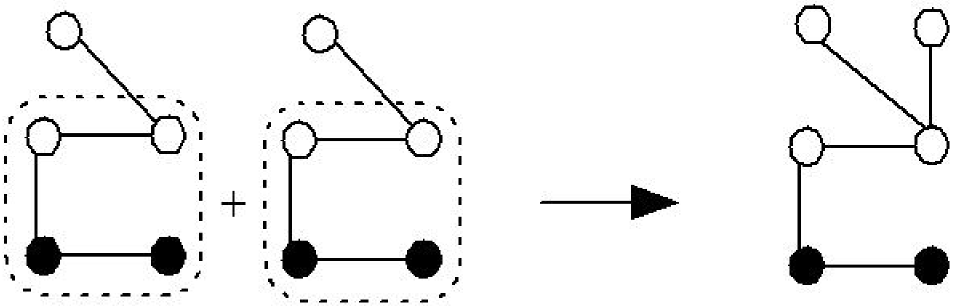
Definition : Subgraph Mining

- Frequent subgraphs
  - A (sub)graph is frequent if its support (occurrence frequency) in a given dataset is no less than a minimum support threshold
- Applications of graph pattern mining
  - Mining biochemical structures
  - Program control flow analysis
  - Building blocks for graph classification, clustering, compression, etc.

Methodology



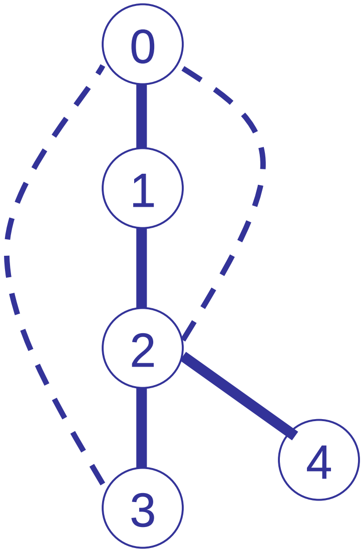
Methodology: *Joining two graphs*



- AGM (Inokuchi, et al. PKDD'00)
  - generates new graphs with one more node
- FSG (Kuramochi and Karypis ICDM'01)
  - generates new graphs with one more edge

Graph Mining and Sequence Mining

- Flatten a graph into a sequence using depth first search



e0: (0,1)

e1: (1,2)

e2: (2,0)

e3: (2,3)

e4: (3,1)

e5: (2,4)

Closed Frequent Graphs

- Motivation: Handling graph pattern explosion problem
- Closed frequent graph
  - A frequent graph G is closed if there exists no supergraph of G that carries the same support as G
  - If some of G's subgraphs have the same support, it is unnecessary to output these subgraphs (nonclosed graphs)
  - Lossless compression: still ensures that the mining result is complete

I(B): Pattern Mining Applications

Allatin: Mining Alternating Patterns for Defect Detection

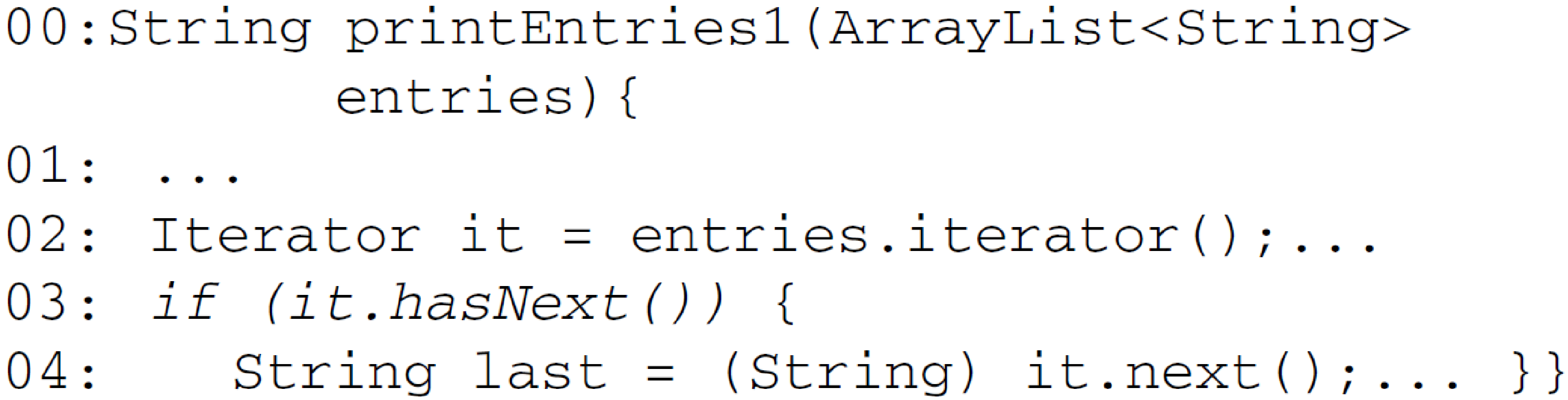
Allatin: Mining Alternative Patterns

- Suresh Thummalapenta, IBM Research
- Tao Xie, North Carolina State University

- Published in Automated Software Engineering Journal, 2011

Introduction

- Programming rules often exists for APIs



```
String printEntries1(ArrayList<String>
             entries){
 ...
 Iterator it = entries.iterator();...
 if (it.hasNext()) {
    String last = (String) it.next();... }}
```

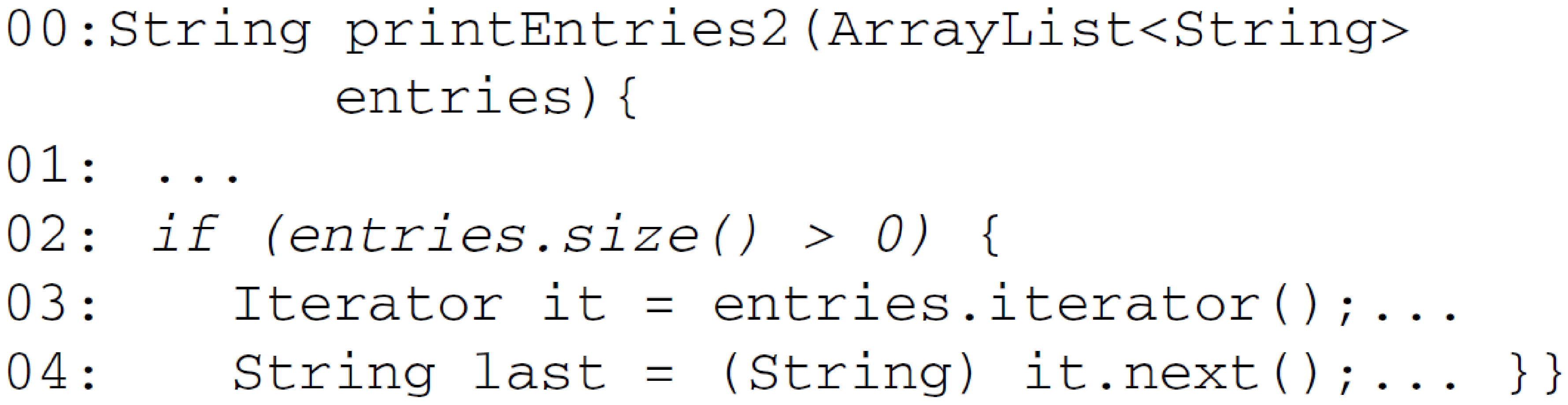
- These programming rules are often not documented well

Introduction

- Allatin recovers common usages of APIs
- Expressed as simple patterns:
  - *P1 = Boolean-check* on *return* of *Iterator.hasNext* before *Iterator.next*
- Patterns are mined by looking to many code pieces that use the API before in the internet.

Introduction

- There might be various acceptable usages



```
String printEntries2(ArrayList<String>
            entries){
  ...
  if (entries.size() > 0) {
      Iterator it = entries.iterator();...
      String last = (String) it.next();... }}
```

- Alternative pattern: *P2 = Constant-check* on *return* of *ArrayList.size* before *Iterator.next*

Introduction

- Allatin recovers a combination of simple patterns
  - And patterns: P1 AND P2
  - Or patterns: P1 OR P2
  - XOR patterns: P1 XOR P2
  - Combo patterns: (P1 AND P2) XOR P3

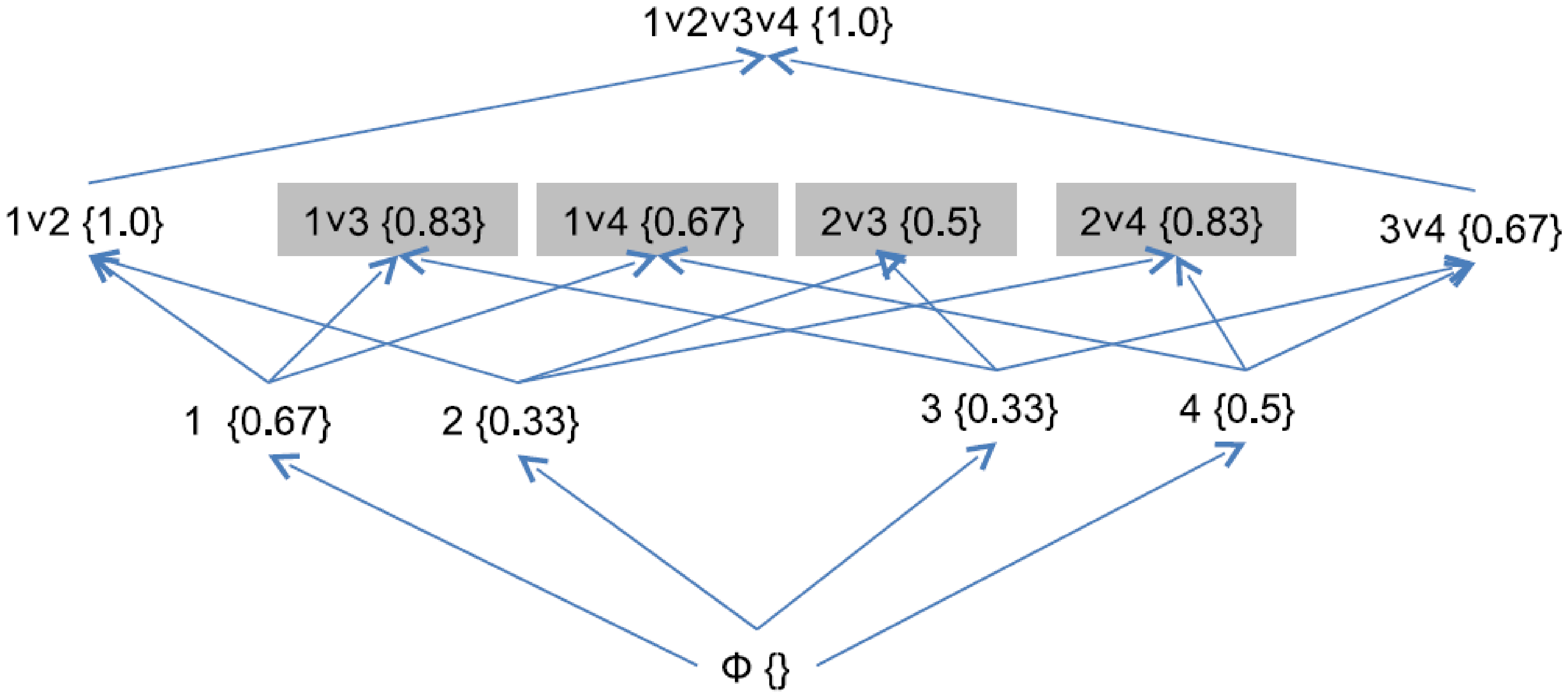
- The patterns are used to detect neglected conditions

Methodology

- Phase 1: Gathering code examples
- Phase 2: Generating pattern candidates
  - Focus on condition checks before and after API method invocations
- Phase 3: Mining alternative patterns
- Phase 4: Detect neglected conditions

Methodology

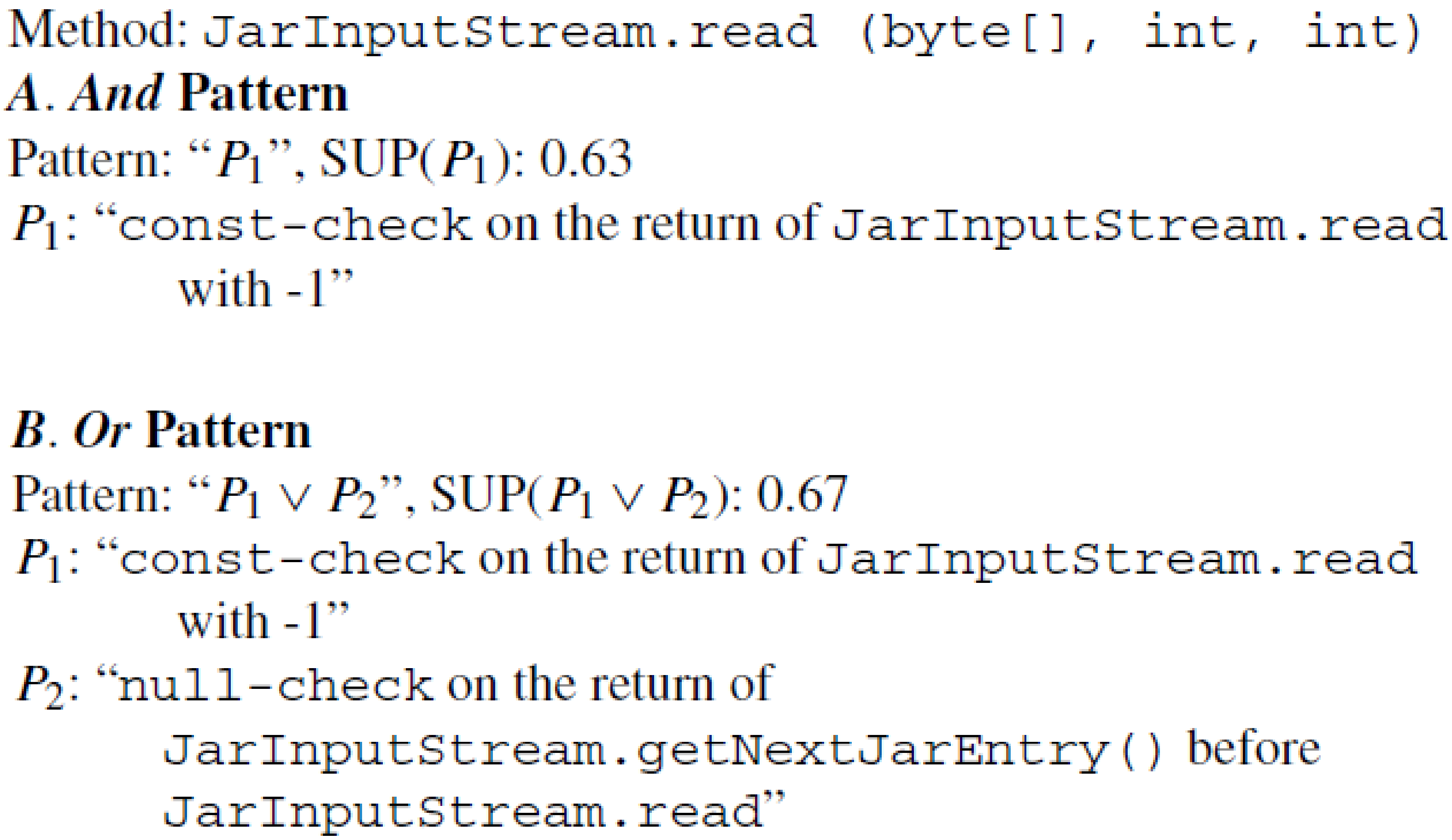
- Algorithm
  - Starts with small pattern
  - Combines them by various operators



Methodology

- Limitation
  - Employ a number of ad-hoc heuristics
    - If “A AND B” and “A XOR B” have support >= min-sup then the right pattern is “A OR B”
  - No guarantee that a complete set of patterns are mined

Experiment



Method: `JarInputStream.read (byte[], int, int)`

A. *And* Pattern

Pattern: “P_1”, SUP(P_1): 0.63

P_1: “`const-check` on the return of `JarInputStream.read` with -1”

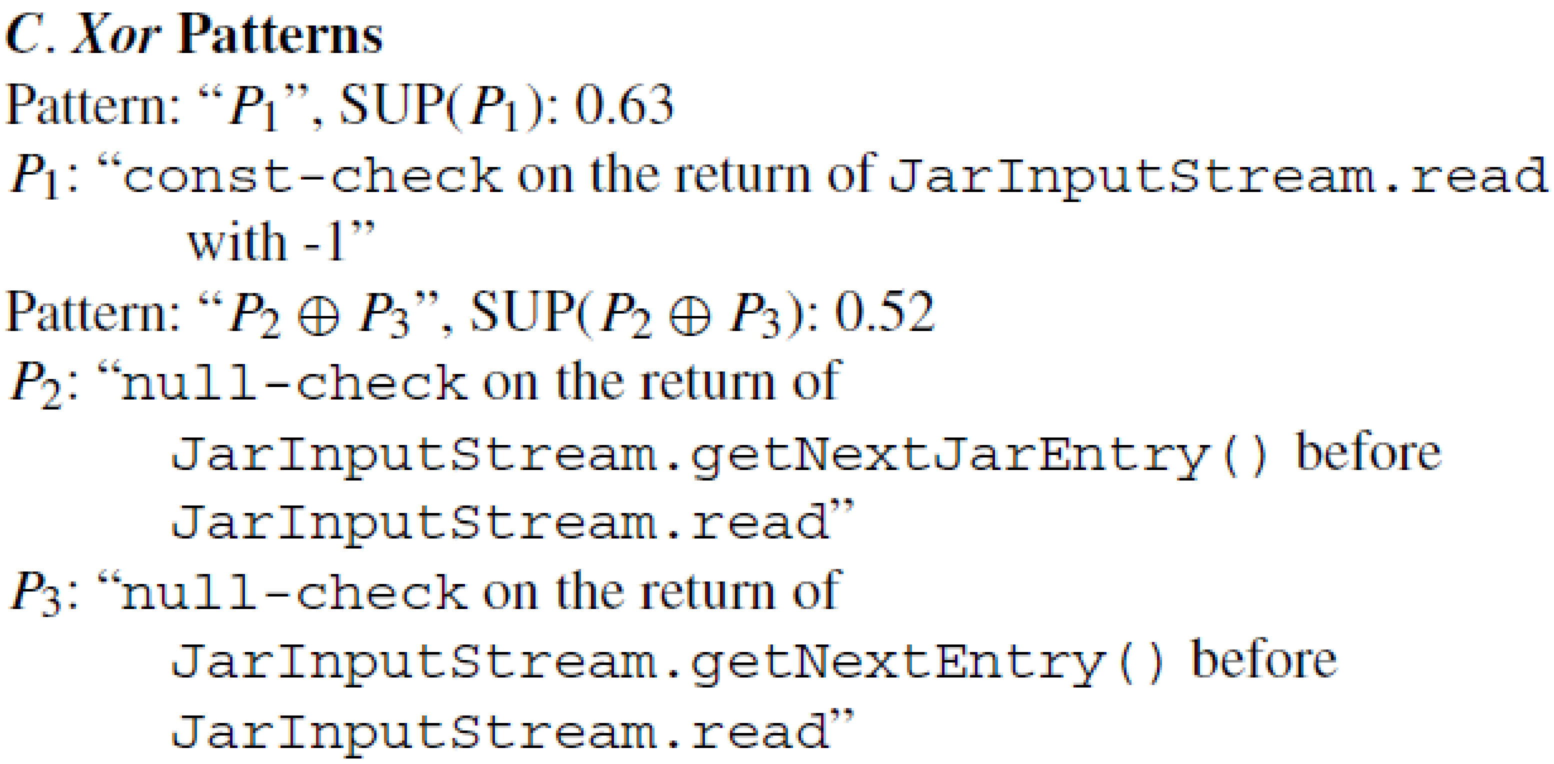
B. *Or* Pattern

Pattern: “$P_1 \vee P_2$”, SUP($P_1 \vee P_2$): 0.67

P_1: “`const-check` on the return of `JarInputStream.read` with -1”

P_2: “`null-check` on the return of `JarInputStream.getNextJarEntry()` before `JarInputStream.read`”

Experiment



C. *Xor* Patterns

Pattern: "P_1", SUP(P_1): 0.63

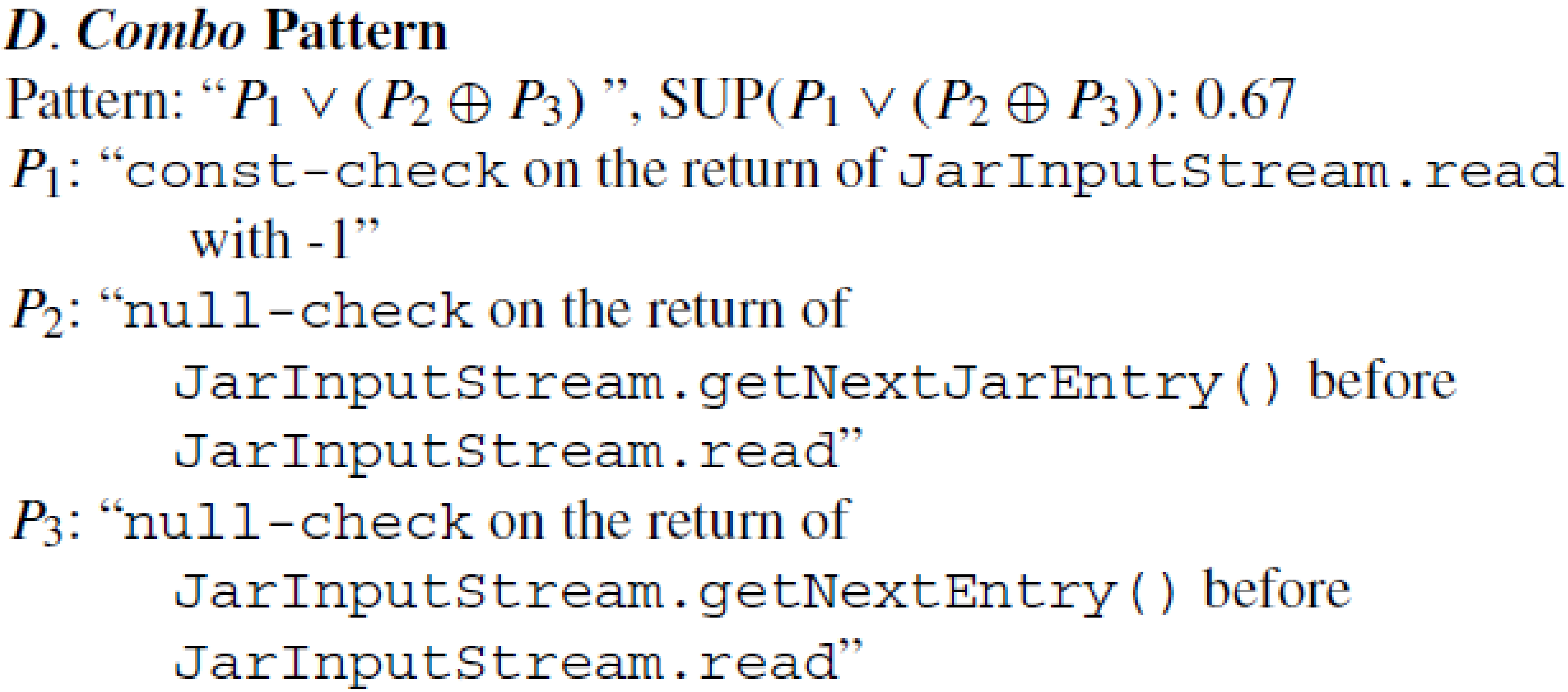
P_1: "`const-check` on the return of `JarInputStream.read` with -1"

Pattern: "$P_2 \oplus P_3$", SUP($P_2 \oplus P_3$): 0.52

P_2: "`null-check` on the return of `JarInputStream.getNextJarEntry()` before `JarInputStream.read`"

P_3: "`null-check` on the return of `JarInputStream.getNextEntry()` before `JarInputStream.read`"

Experiment



D. Combo Pattern

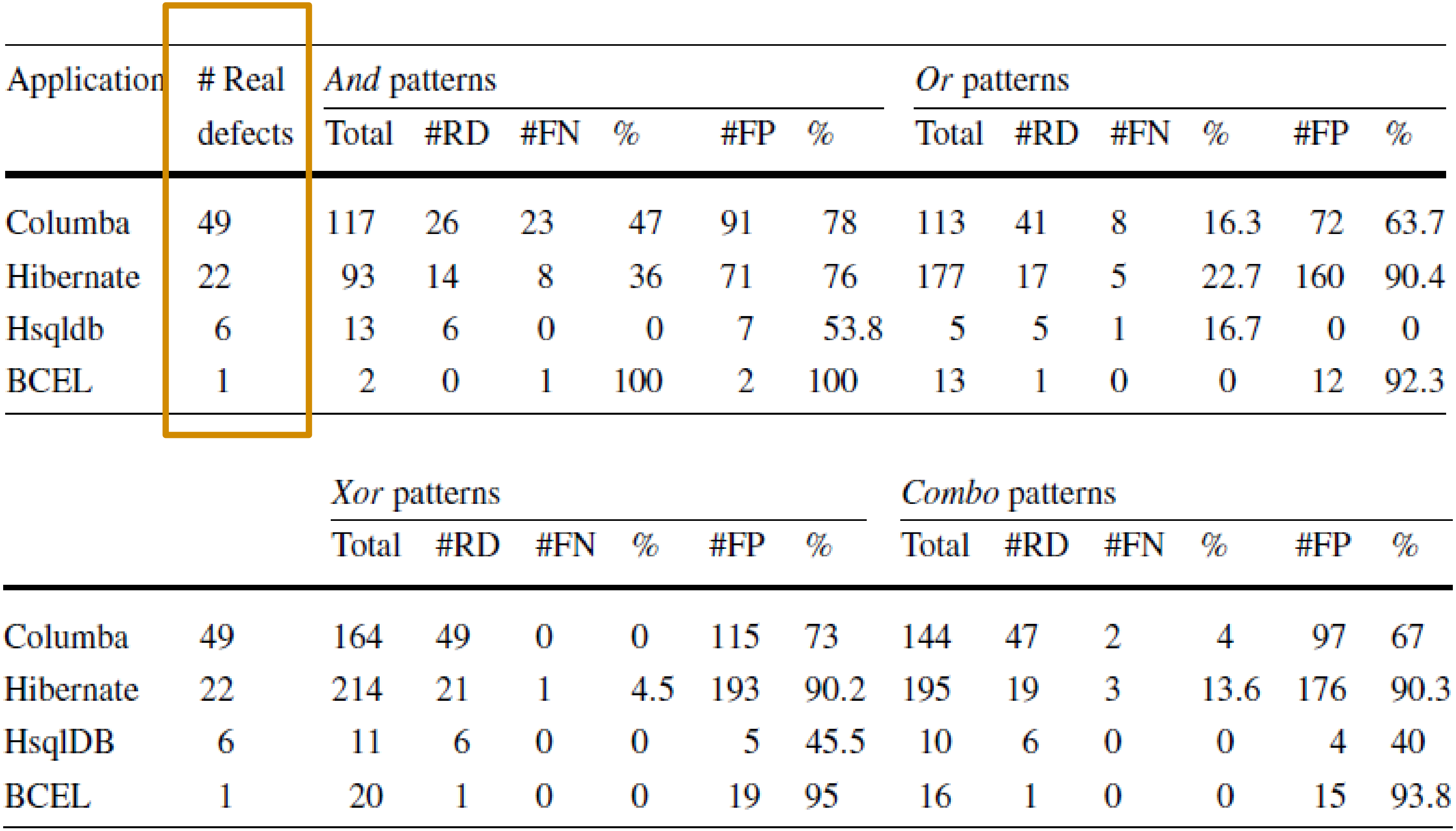
Pattern: "$P_1 \vee (P_2 \oplus P_3)$ ", SUP($P_1 \vee (P_2 \oplus P_3)$): 0.67

P_1: "`const-check` on the return of `JarInputStream.read` with -1"

P_2: "`null-check` on the return of `JarInputStream.getNextJarEntry()` before `JarInputStream.read`"

P_3: "`null-check` on the return of `JarInputStream.getNextEntry()` before `JarInputStream.read`"

Experiment



Application	# Real defects	*And* patterns						*Or* patterns					
		Total	#RD	#FN	%	#FP	%	Total	#RD	#FN	%	#FP	%
Columba	49	117	26	23	47	91	78	113	41	8	16.3	72	63.7
Hibernate	22	93	14	8	36	71	76	177	17	5	22.7	160	90.4
Hsqldb	6	13	6	0	0	7	53.8	5	5	1	16.7	0	0
BCEL	1	2	0	1	100	2	100	13	1	0	0	12	92.3

		Xor patterns						*Combo* patterns					
		Total	#RD	#FN	%	#FP	%	Total	#RD	#FN	%	#FP	%
Columba	49	164	49	0	0	115	73	144	47	2	4	97	67
Hibernate	22	214	21	1	4.5	193	90.2	195	19	3	13.6	176	90.3
HsqlDB	6	11	6	0	0	5	45.5	10	6	0	0	4	40
BCEL	1	20	1	0	0	19	95	16	1	0	0	15	93.8

Other Applications

- Mining temporal specifications
    - Zhenmin Li, Yuanyuan Zhou: PR-Miner: automatically extracting implicit programming rules and detecting violations in large software code. ESEC/SIGSOFT FSE 2005: 306-315
    - David Lo, Siau-Cheng Khoo, Chao Liu: Efficient mining of iterative patterns for software specification discovery. KDD 2007: 460-469
    - David Lo, Bolin Ding, Lucia, Jiawei Han: Bidirectional mining of non-redundant recurrent rules from a sequence database. ICDE 2011: 1043-1054

Other Applications

- Mining temporal specifications (cont)
  - David Lo, Jinyan Li, Limsoon Wong, Siau-Cheng Khoo: Mining Iterative Generators and Representative Rules for Software Specification Discovery. IEEE Trans. Knowl. Data Eng. 23(2): 282-296 (2011)

- Detecting duplicate bug reports
  - David Lo, Hong Cheng, Lucia: Mining closed discriminative dyadic sequential patterns. EDBT 2011: 21-32

Other Applications

- Bug and failure identification
  - Hwa-You Hsu, James A. Jones, Alessandro Orso: Rapid: Identifying Bug Signatures to Support Debugging Activities. ASE 2008: 439-442
  - Hong Cheng, David Lo, Yang Zhou, Xiaoyin Wang, Xifeng Yan: Identifying bug signatures using discriminative graph mining. ISSTA 2009: 141-152
  - David Lo, Hong Cheng, Jiawei Han, Siau-Cheng Khoo, Chengnian Sun: Classification of software behaviors for failure detection: a discriminative pattern mining approach. KDD 2009: 557-566

Other Applications

- Predicting project outcome
  - Didi Surian, Yuan Tian, David Lo, Hong Cheng, Ee-Peng Lim: Predicting Project Outcome Leveraging Socio-Technical Network Patterns. CSMR 2013: 47-56

- Detecting co-occurring changes
  - Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, Andreas Zeller: Mining Version Histories to Guide Software Changes. ICSE 2004: 563-572

Part II: Clustering

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters

- Cluster analysis
  - Finding similarities among data objects according to their characteristics
  - Grouping similar data objects into clusters

Part II: Clustering

- Typical applications
    - As a stand-alone tool to get insight into data
    - As a preprocessing step for other algorithms

Quality: What Is Good Clustering?

- A good clustering method will produce clusters with:
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns

Structure

- Techniques
  - k-Means
  - k-Medoids
  - Hierarchical Clustering
- Applications
  - Performance Debugging in the Large via Mining Millions of Stack Traces
  - Other applications

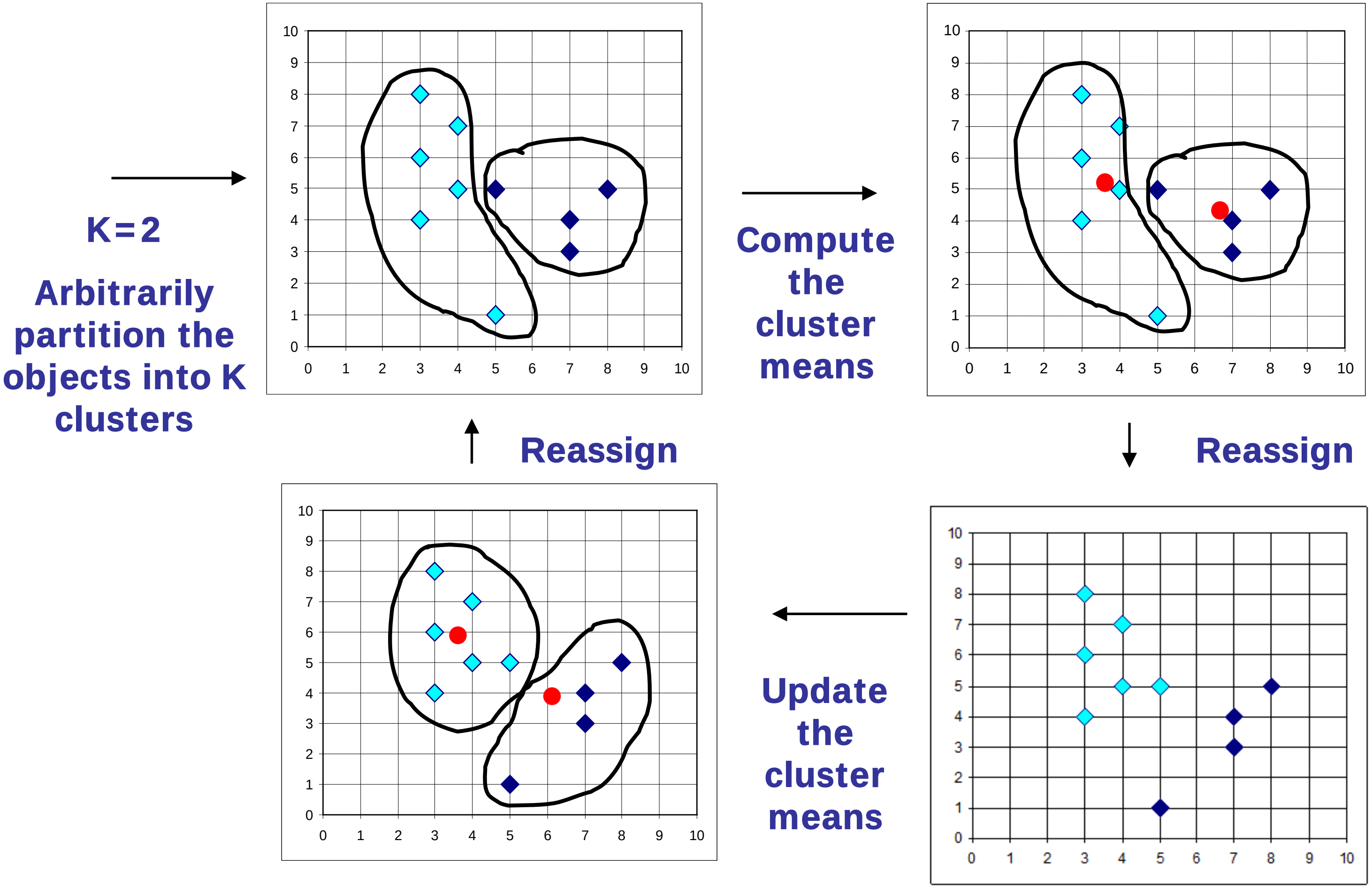
II(A): Clustering Techniques

k-Means

The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  1. Partition objects into k nonempty subsets
  2. Compute the means of the clusters of the current partition (the mean is the center of the cluster)
  3. Re-assign each object to the cluster with the nearest mean
  4. Go back to Step 2, stop when no more new assignment

The *K-Means* Clustering Method



K=2

Arbitrarily partition the objects into K clusters

Compute the cluster means

Reassign

Update the cluster means

Reassign

Limitations

- Applicable only when mean is defined
- Need to specify *k*, the number of clusters, in advance
- Unable to handle noisy data and outliers

II(A): Clustering Techniques

k-Medoids

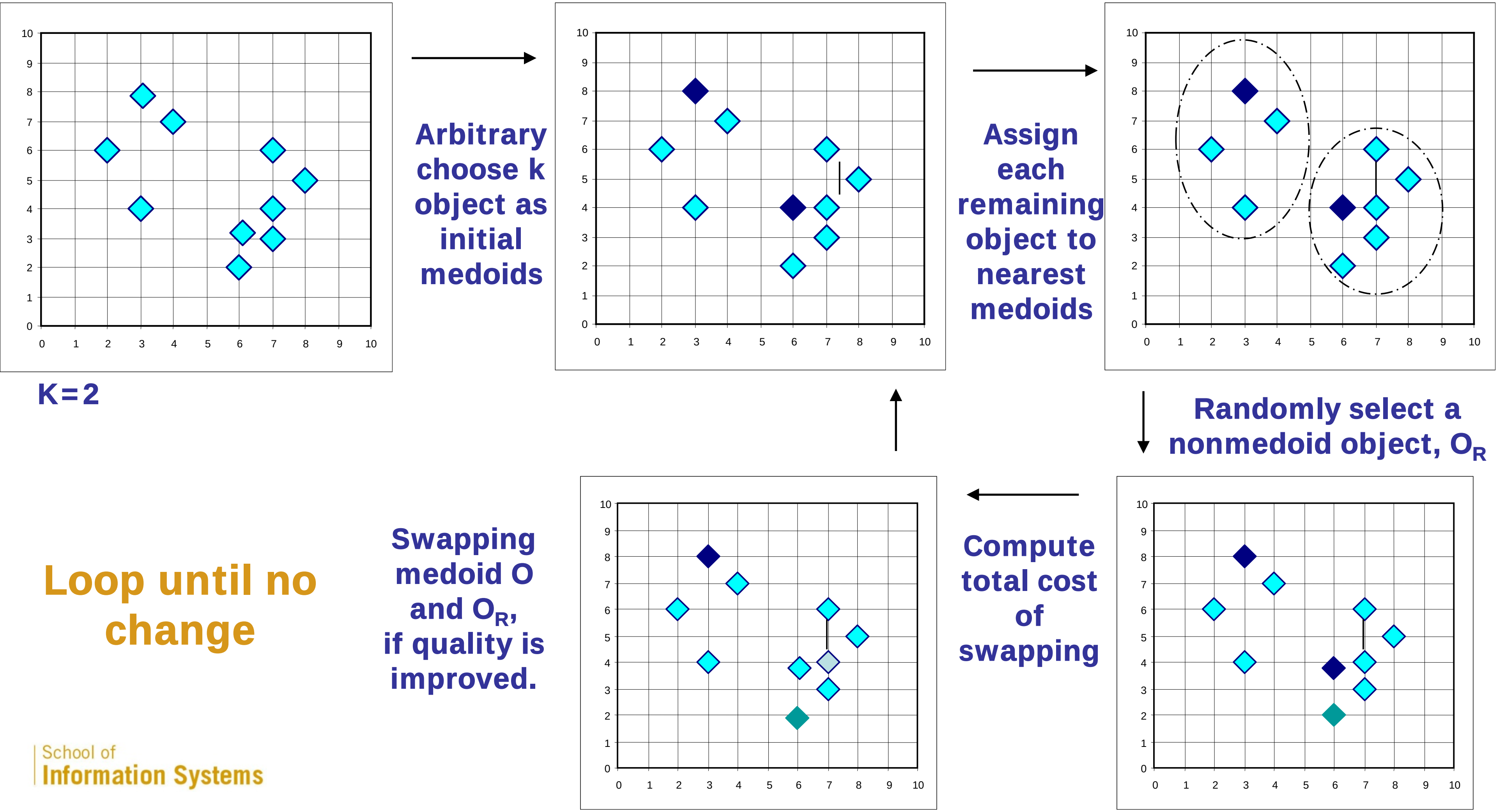
k-Medoids

- Find representative objects, called medoids, in clusters
- *Many algorithms:*
  - *PAM* (Partitioning Around Medoids, 1987)
  - *CLARA* (Kaufmann & Rousseeuw, 1990)
  - *CLARANS* (Ng & Han, 1994): Randomized sampling
  - Focusing + spatial data structure (Ester et al., 1995)

PAM (Partitioning Around Medoids) (1987)

- PAM (Kaufman and Rousseeuw, 1987)
- Use real object to represent the cluster
  - Select *k* representative objects arbitrarily
  - For each pair of non-selected object *h* and selected object *i*, calculate the total swapping cost TC_{ih}
  - For each pair of *i* and *h*,
    - If $TC_{ih} < 0$, *i* is replaced by *h*
    - Then reassign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change

PAM (Partitioning Around Medoids) (1987)



K=2

Arbitrary choose k object as initial medoids

Assign each remaining object to nearest medoids

Randomly select a nonmedoid object, O_R

Compute total cost of swapping

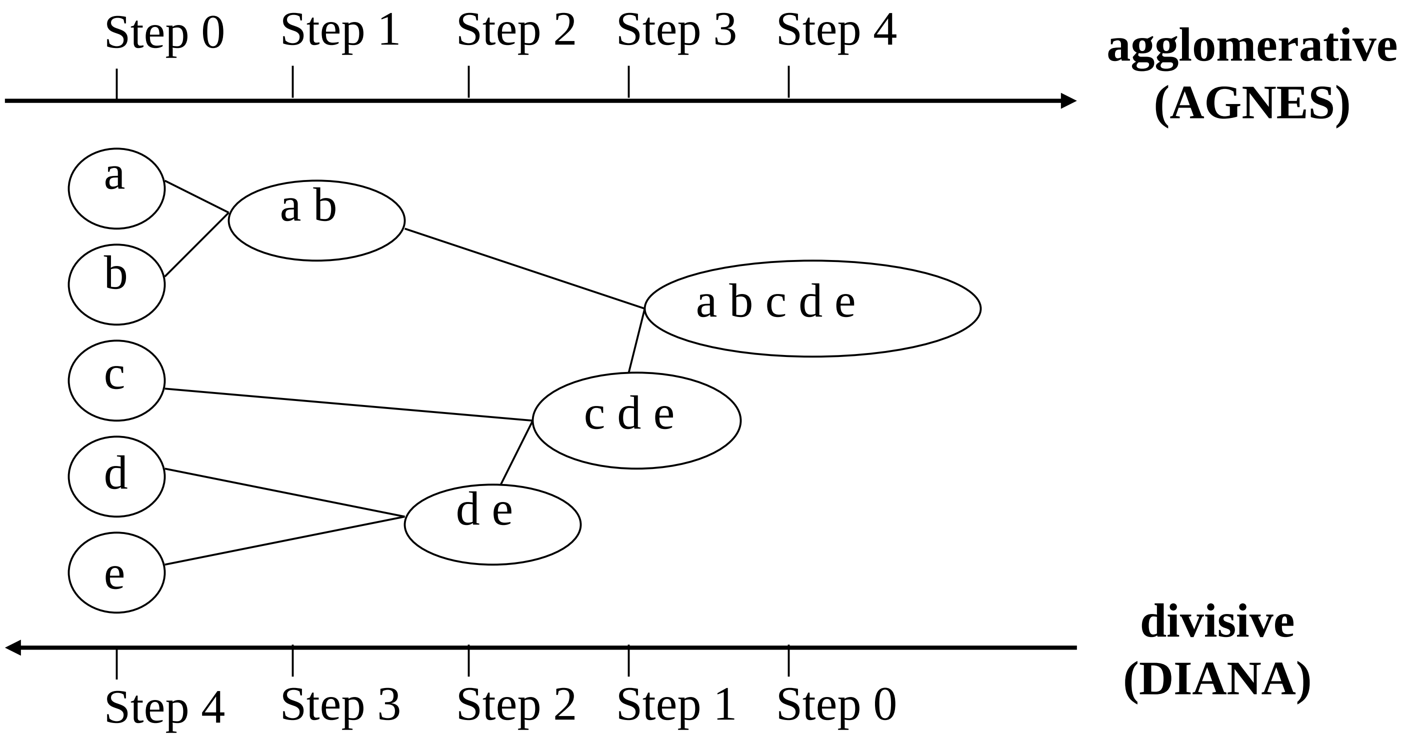
Swapping medoid O and O_R, if quality is improved.

Loop until no change

II(A): Clustering Techniques

Hierarchical Clustering

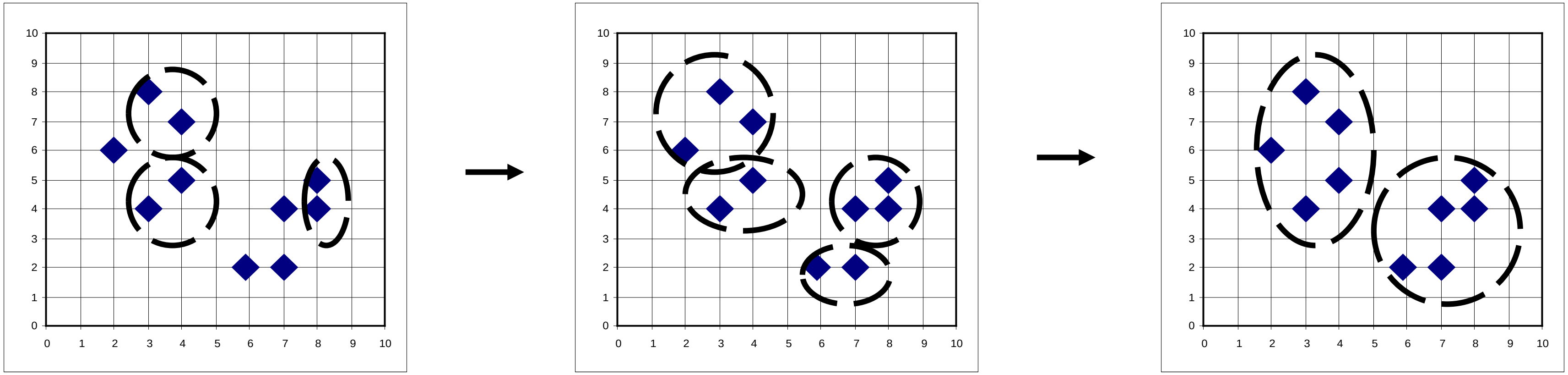
Hierarchical Clustering



- This method does not require the number of clusters k as an input, but needs a termination condition

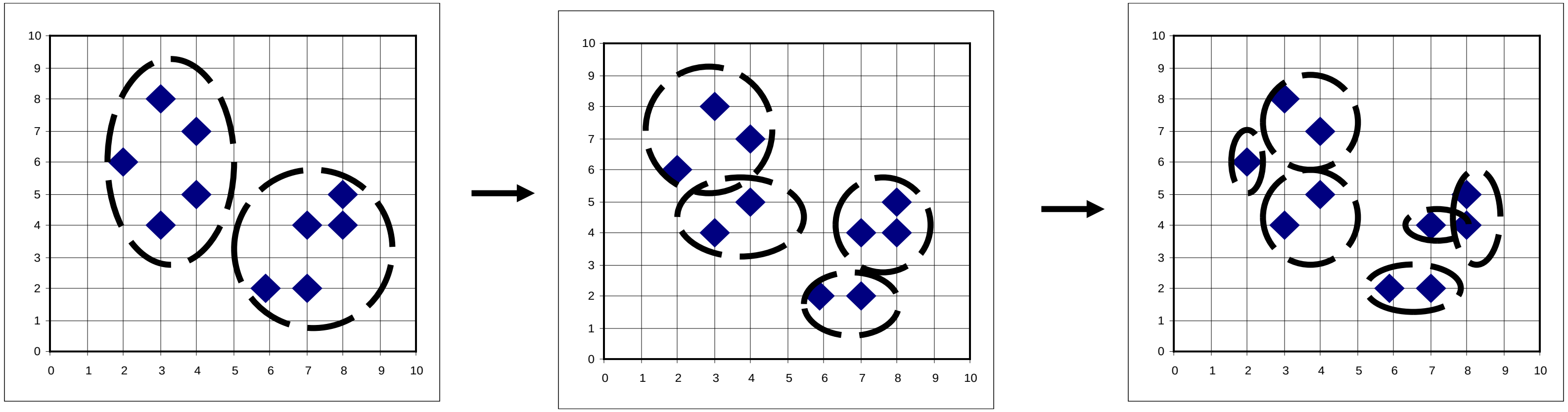
AGNES (Agglomerative Nesting)

- Introduced in Kaufmann and Rousseeuw (1990)
- Merge nodes that have the least dissimilarity
- Go on until eventually all nodes belong to the same cluster

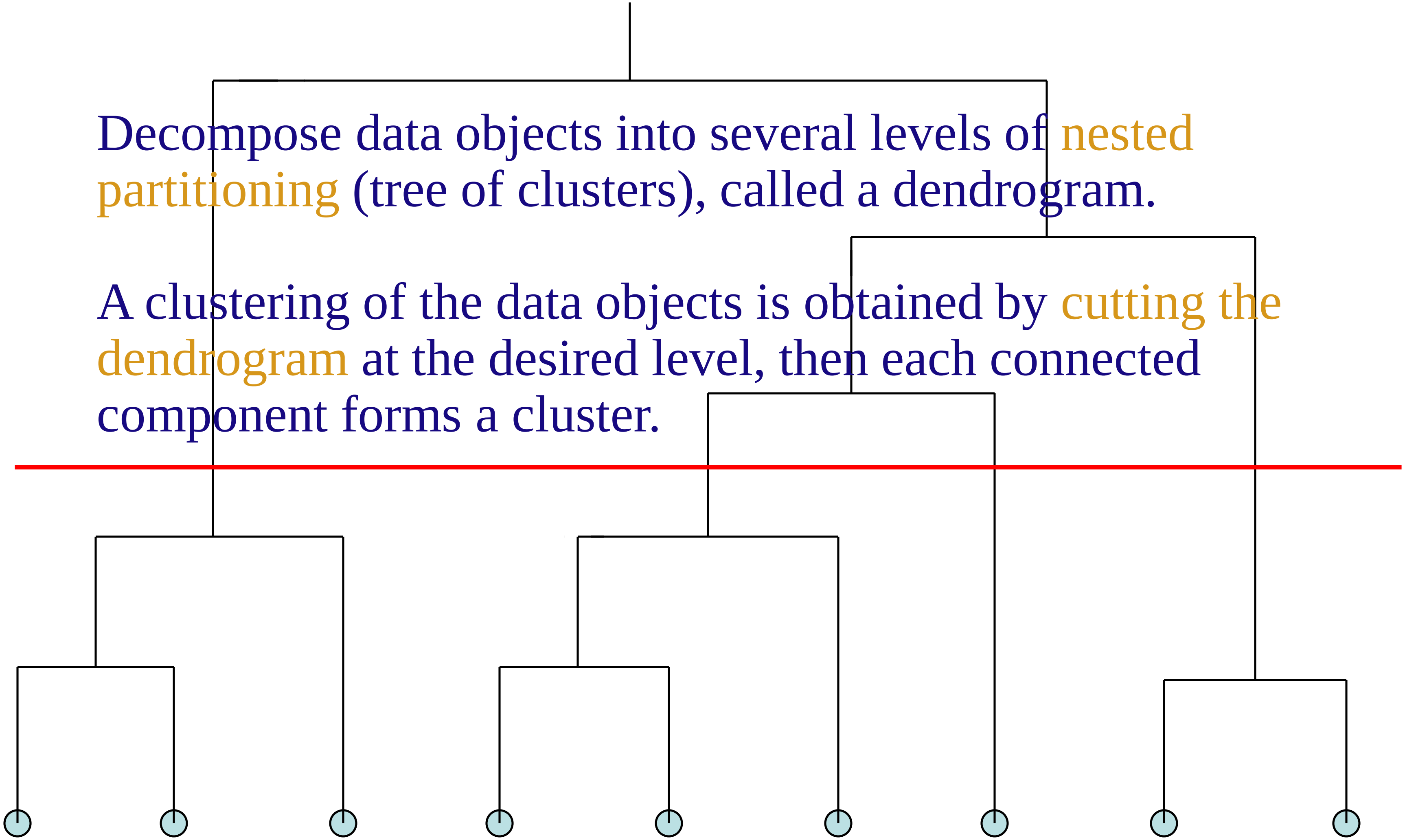


DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Inverse order of AGNES
- Eventually each node forms a cluster on its own



Dendrogram: Shows How the Clusters are Merged



Decompose data objects into several levels of nested partitioning (tree of clusters), called a dendrogram.

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.

II(B): Clustering Applications

Performance Debugging in the Large via Mining Millions of Stack Traces

Performance Debugging by Mining Stack Traces

- Shi Han, Yingnong Dang, Song Ge, Dongmei Zhang, Microsoft Research
- Tao Xie, North Carolina State University

- Published in International Conference on Software Engineering, 2012

Introduction

- Performance of software system is important
- Performance bugs leads to unbearably slow system
- To debug performance issues, Windows has the facility to collect execution traces

Introduction

- Manual investigation needs to be performed
- Very tedious and time-consuming
  - Many execution traces
  - Each of them can be very long
- Semi/Fully automated support needed
- Proposed solution:
  - Group related execution traces together

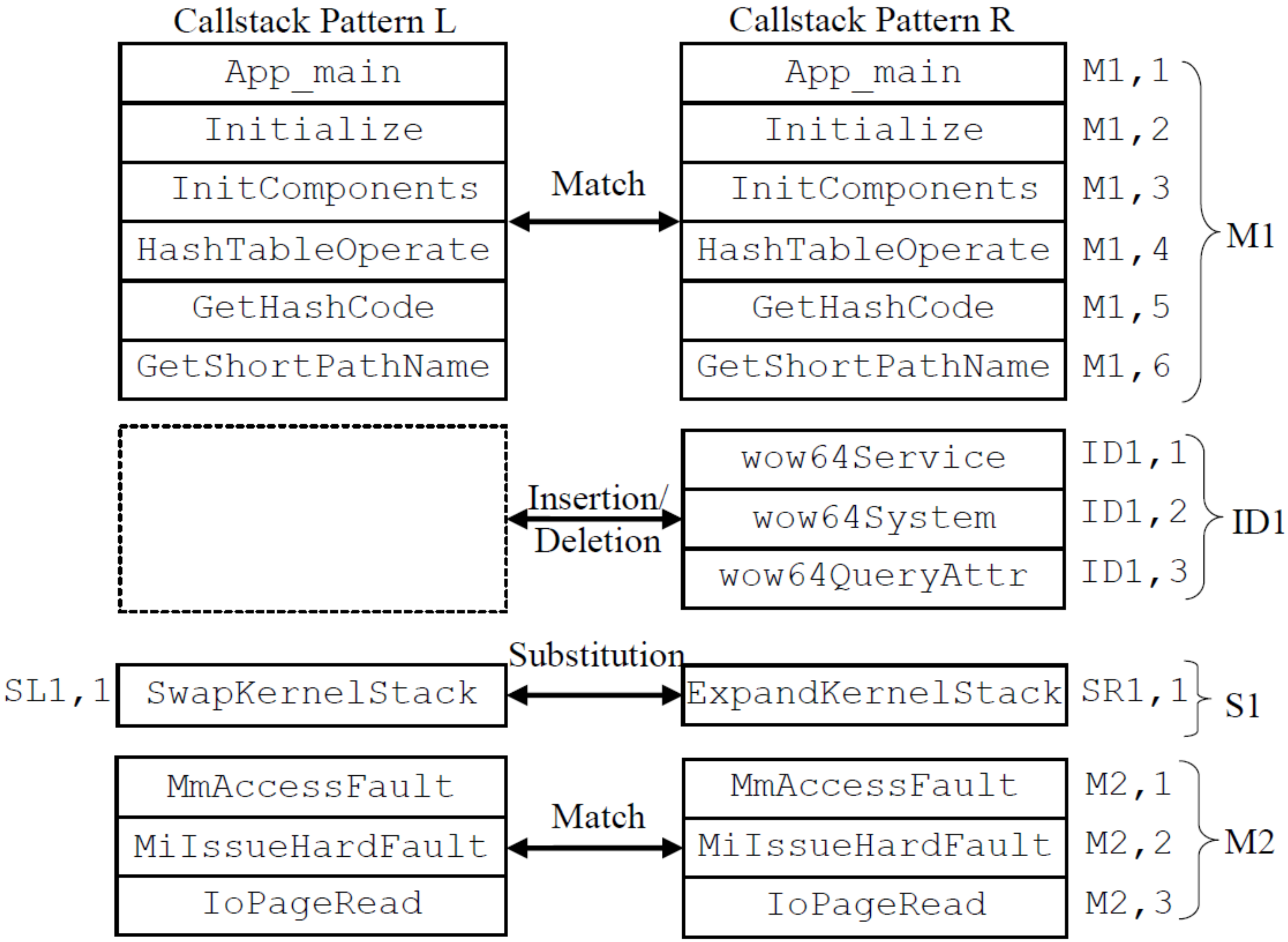
Methodology

- Phase 1: Extract area of interest
    - Not all collected execution traces are interesting
    - Focus on events that wait for other events in the traces
    - Use developers domain knowledge to localize this area of interest
- Phase 2: Extract maximal sequential patterns
- Phase 3: Cluster the patterns together

Methodology

- Hierarchical clustering is performed
- Key: similarity measure
- Similarity measure:
  - Alignment of two patterns
  - Computation of similarity

Methodology



Experiment

- Finding hidden performance bugs
    - on Windows Explorer UI
- Input: 921 trace streams
    - 140 million call stacks
- Output: 1,215 pattern clusters
- Pattern mining and clustering time: 10 hours

Experiment

- Developer manually investigate the clusters
- Eight hours -> produce 93 signatures
- Twelve of them are highly impactful performance bugs

Other Applications

- Testing multi-threaded applications
  - Adrian Nistor, Qingzhou Luo, Michael Pradel, Thomas R. Gross, Darko Marinov: Ballerina: Automatic generation and clustering of efficient random unit tests for multithreaded code. ICSE 2012: 727-737

- Defect prediction
  - Nicolas Bettenburg, Meiyappan Nagappan, Ahmed E. Hassan: Think locally, act globally: Improving defect and effort prediction models. MSR 2012: 60-69

Other Applications

- Ontology inference
    - Shaowei Wang, David Lo, Lingxiao Jiang: Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. ICSM 2012: 604-607

- Detecting malicious apps
    - Alessandra Gorla, Ilaria Tavecchia, Florian Gross, Andreas Zeller: Checking app behavior against app descriptions. 1025-1035

Other Applications

- Software remodularization
  - Nicolas Anquetil, Timothy Lethbridge: Experiments with Clustering as a Software Remodularization Method. WCRE 1999: 235-255

Part III: Classification

- Assigns data to some predefined categories
- It performs this
  - By constructing a model
  - Based on:
    - the training set
    - the values (class labels) in a classifying attribute
  - Uses it in classifying new data
- Two steps process:
  - Model construction
  - Model usage

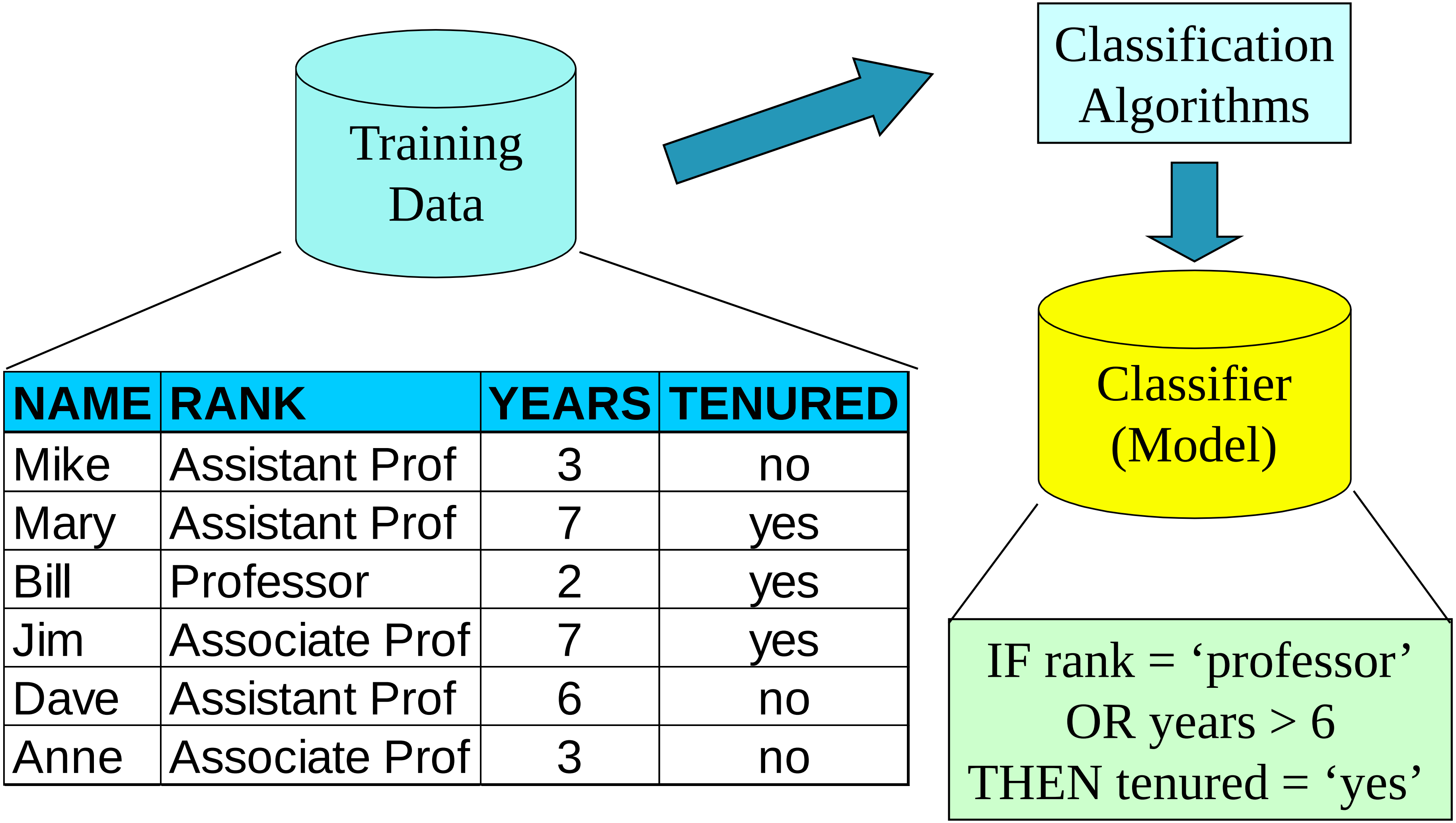
Classification – Model Construction

- Model construction: describing the set of predetermined class/categories
    - The set of tuples used for model construction is called the training set
    - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
    - The model is represented as classification rules, decision trees, or mathematical formulae

Classification – Model Usage

- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Process (1): Model Construction


Training Data

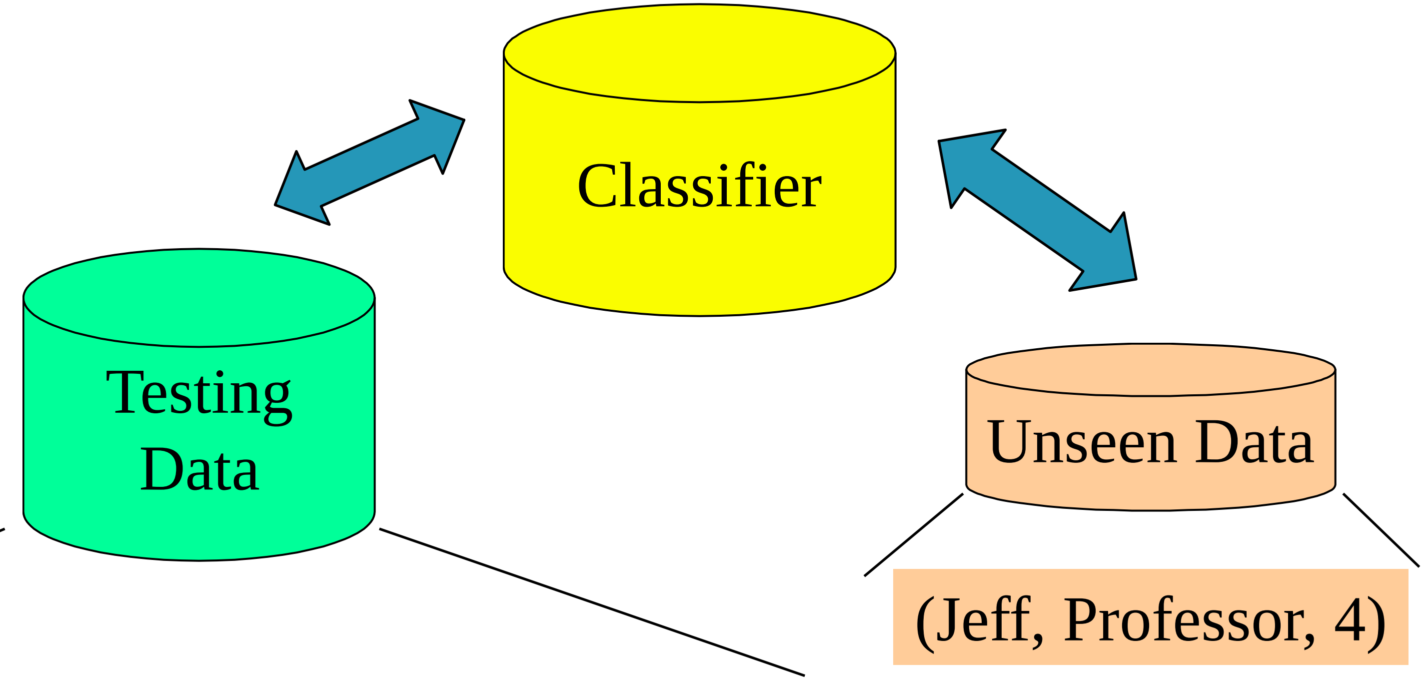
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



(Jeff, Professor, 4)

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

Structure

- Techniques
  - Decision Tree
  - Support Vector Machine
  - k-Nearest Neighbor
- Applications
  - An Industrial Study on the Risk of Software Changes
  - Other Applications

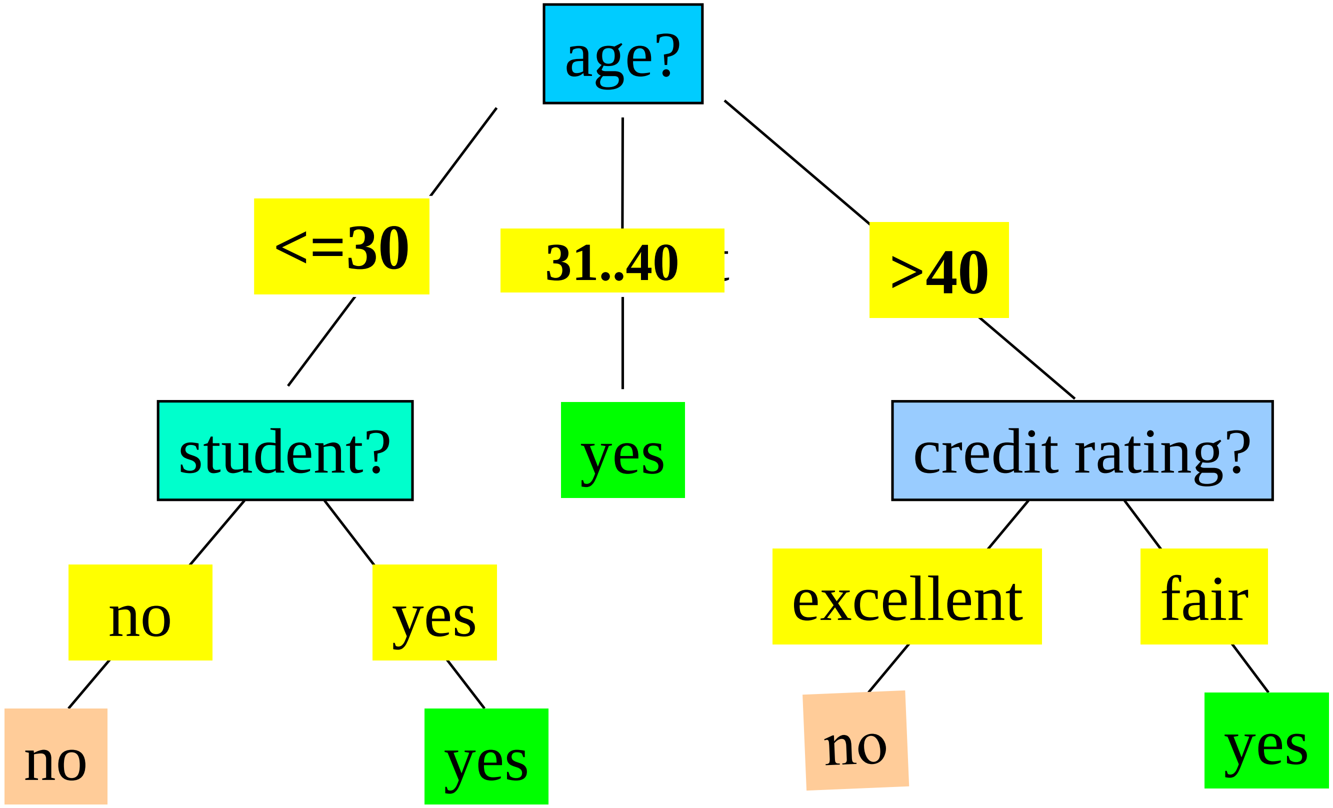
III(A): Classification Techniques

Decision Tree

Decision Tree Induction: Training Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for "*buys_computer*"



High-Level Methodology

- Tree is constructed in a top-down recursive divide-and-conquer manner
    - At start, all the training examples are at the root
    - Examples are partitioned recursively based on selected attributes
    - Attributes are selected on the basis of a heuristic or statistical measure

High-Level Methodology

- Conditions for stopping partitioning
    - All samples for a given node belong to the same class
    - There are no remaining attributes for further partitioning
    - There are no samples left
- Majority voting is employed for classifying the leaf

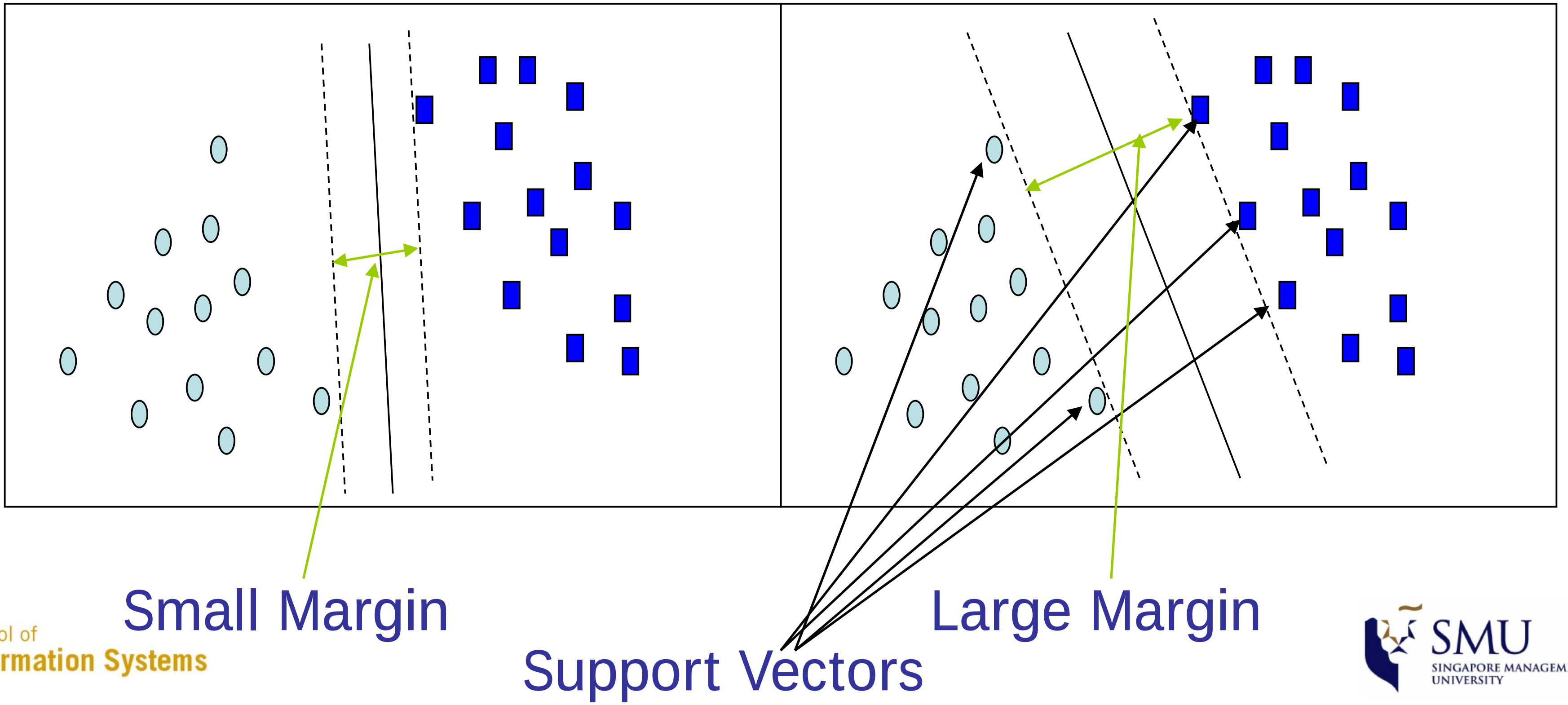
III(A): Classification Techniques

Support Vector Machine (SVM)

High-Level Methodology

- Searches for the linear optimal separating hyperplane (i.e., “decision boundary”)

SVM searches for the hyperplane with the largest margin, i.e., maximum marginal hyperplane (MMH)



High-Level Methodology

- How if not separable by a linear hyperplane?
  - It uses a nonlinear mapping to transform the original training data into a higher dimension
  - With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane

High-Level Methodology

- Features:
  - Training can be slow
  - Accuracy is often high owing to their ability to model complex nonlinear decision boundaries
- Applications:
  - Handwritten digit recognition, object recognition, speaker identification, etc

III(A): Classification Techniques

k-Nearest Neighbors

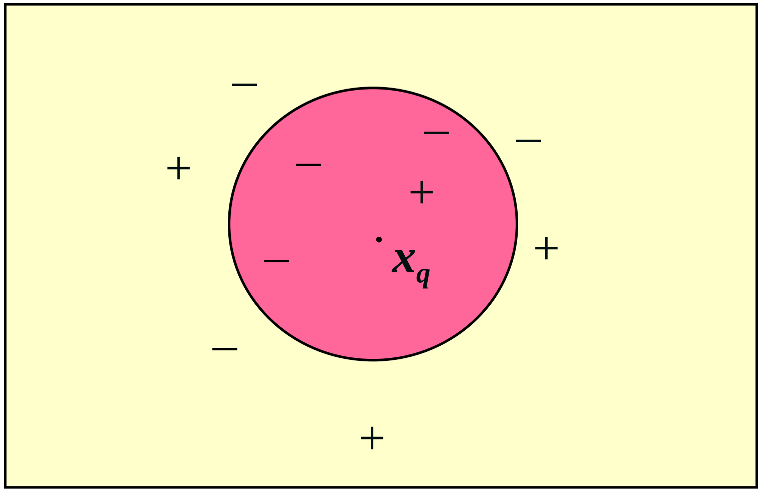
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified

- Typical approaches
  - k-nearest neighbor approach
  - Locally weighted regression
  - Case-based reasoning

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- New instance label is predicted based on its k-NN
- If the predicted label is discrete:
  - Return the most common value among the neighbors in the training data
- If the predicted label is a real number:
  - Return the mean values of the *k* nearest neighbors



Discussion on the *k*-NN Algorithm

- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the k neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Problem:
  - Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
    - To overcome it: elimination of least relevant attributes

III(A): Classification Applications

An Industrial Study on the Risk of Software Changes

Predicting Risk of Software Changes

- Emad Shihab, Ahmed E. Hassan, Queen’s University, Canada
- Bram Adams, Ecole Polytechnique de Montreal, Canada
- Zhen Ming Jiang, Research in Motion, Canada

- Published in ACM Symposium on Foundations of Software Engineering (FSE), 2012

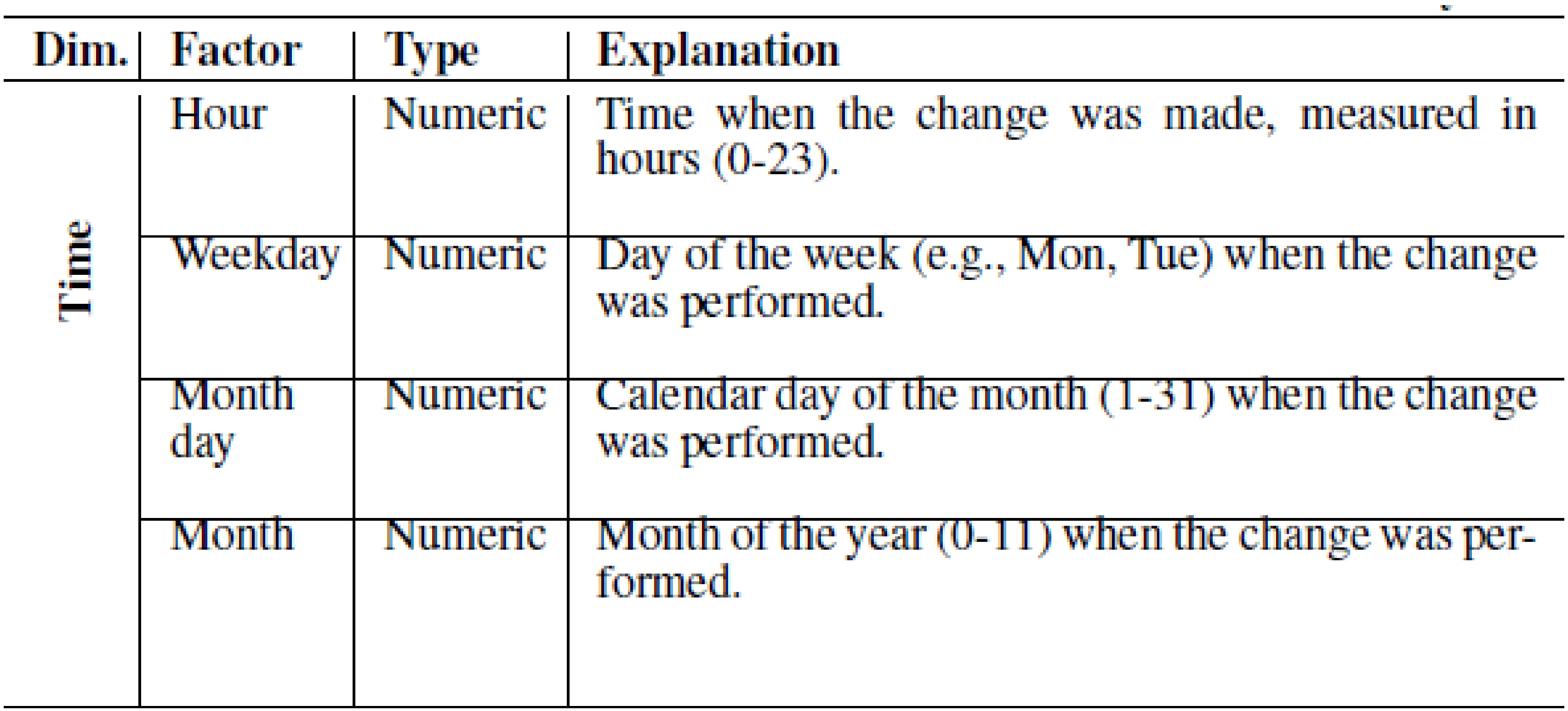
Introduction

- Many companies care about risk
    - Negative impact on products and processes
- Some software changes are risky to be implemented
    - Risky changes = “changes for which developers believe that additional attention is needed in the form of careful code or design reviewing and/or more testing”

Approach

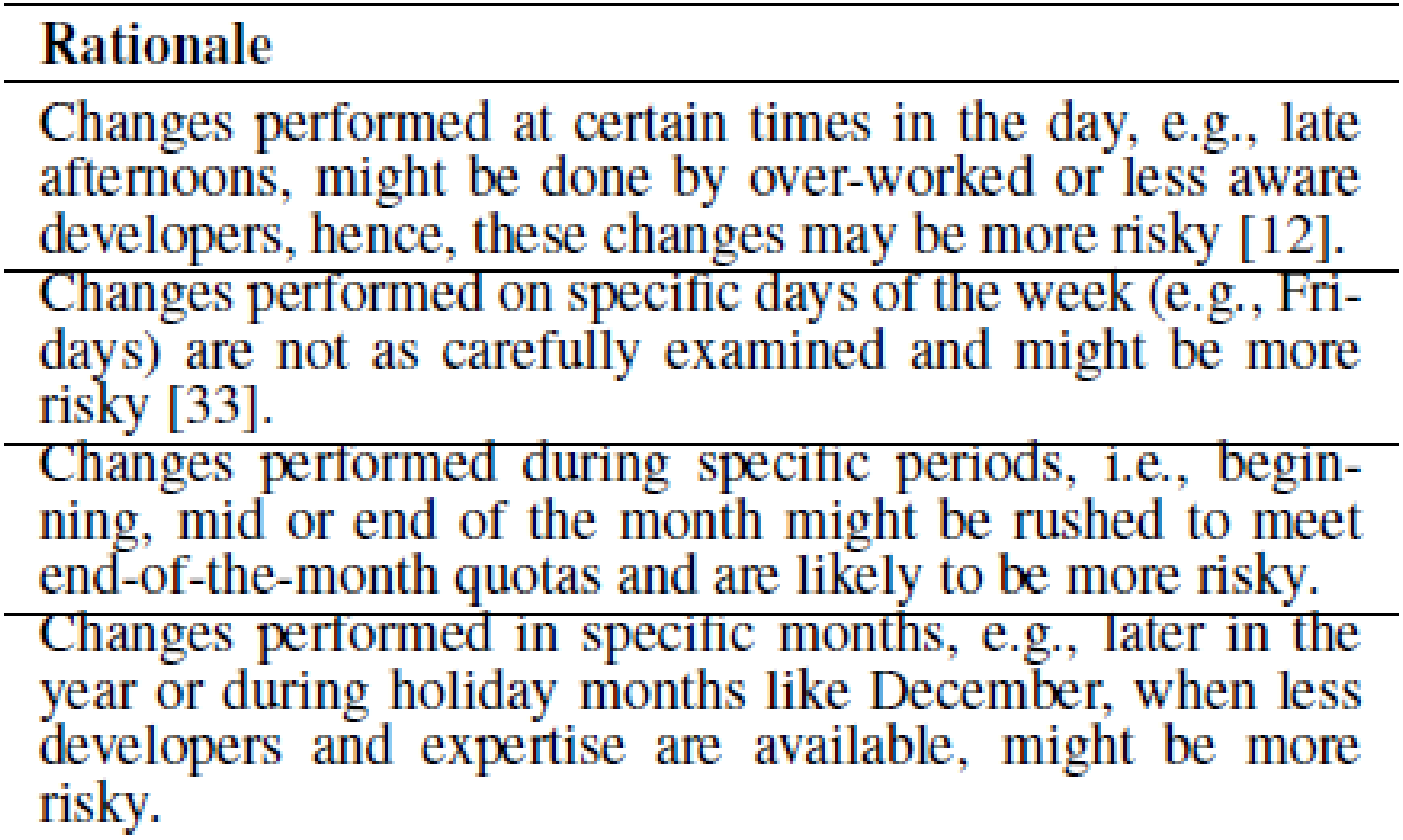
- Feature Extraction (Key Step)
- Classifier Construction
- Classifier Application

Approach



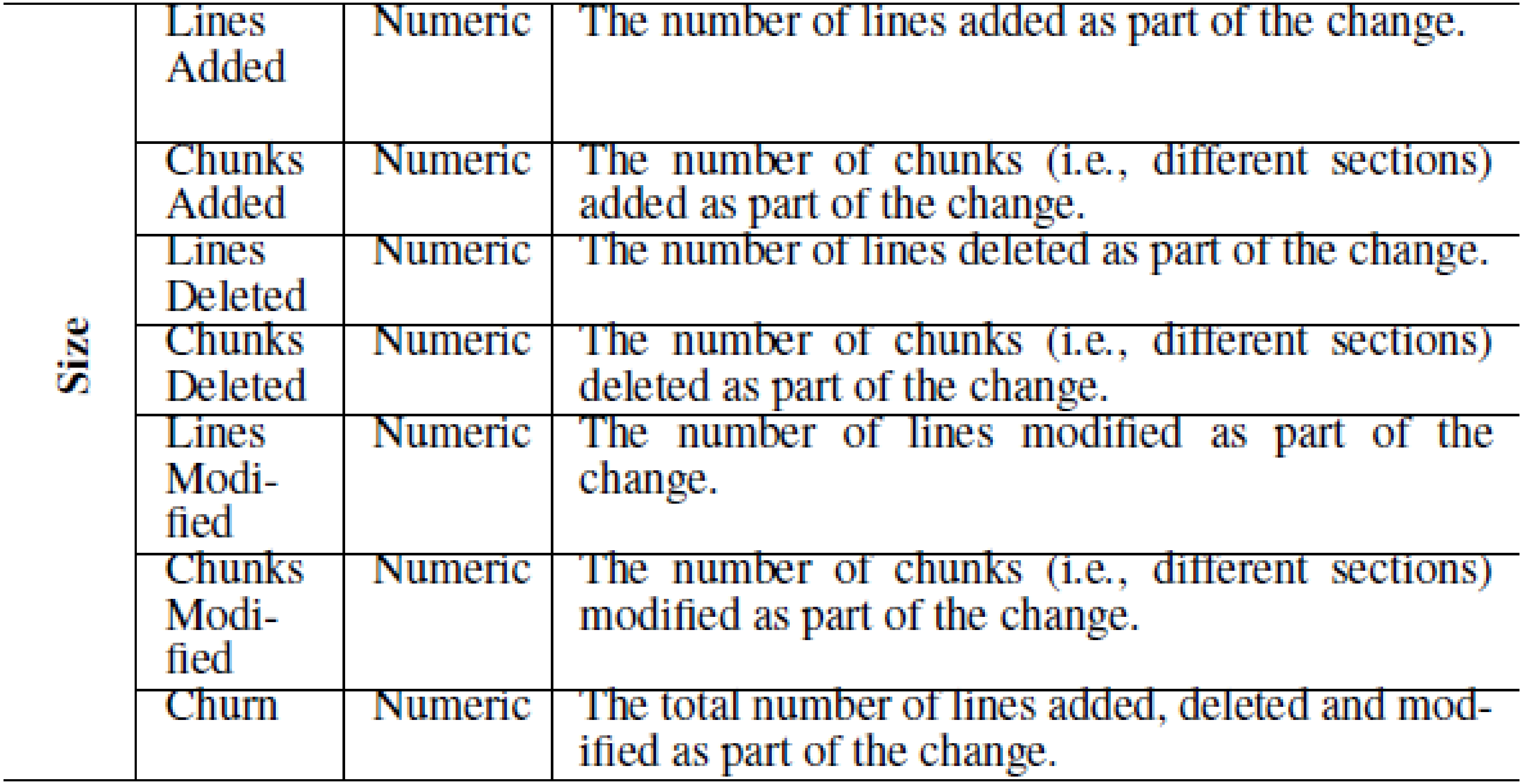
Dim.	Factor	Type	Explanation
Time	Hour	Numeric	Time when the change was made, measured in hours (0-23).
	Weekday	Numeric	Day of the week (e.g., Mon, Tue) when the change was performed.
	Month day	Numeric	Calendar day of the month (1-31) when the change was performed.
	Month	Numeric	Month of the year (0-11) when the change was performed.

Approach



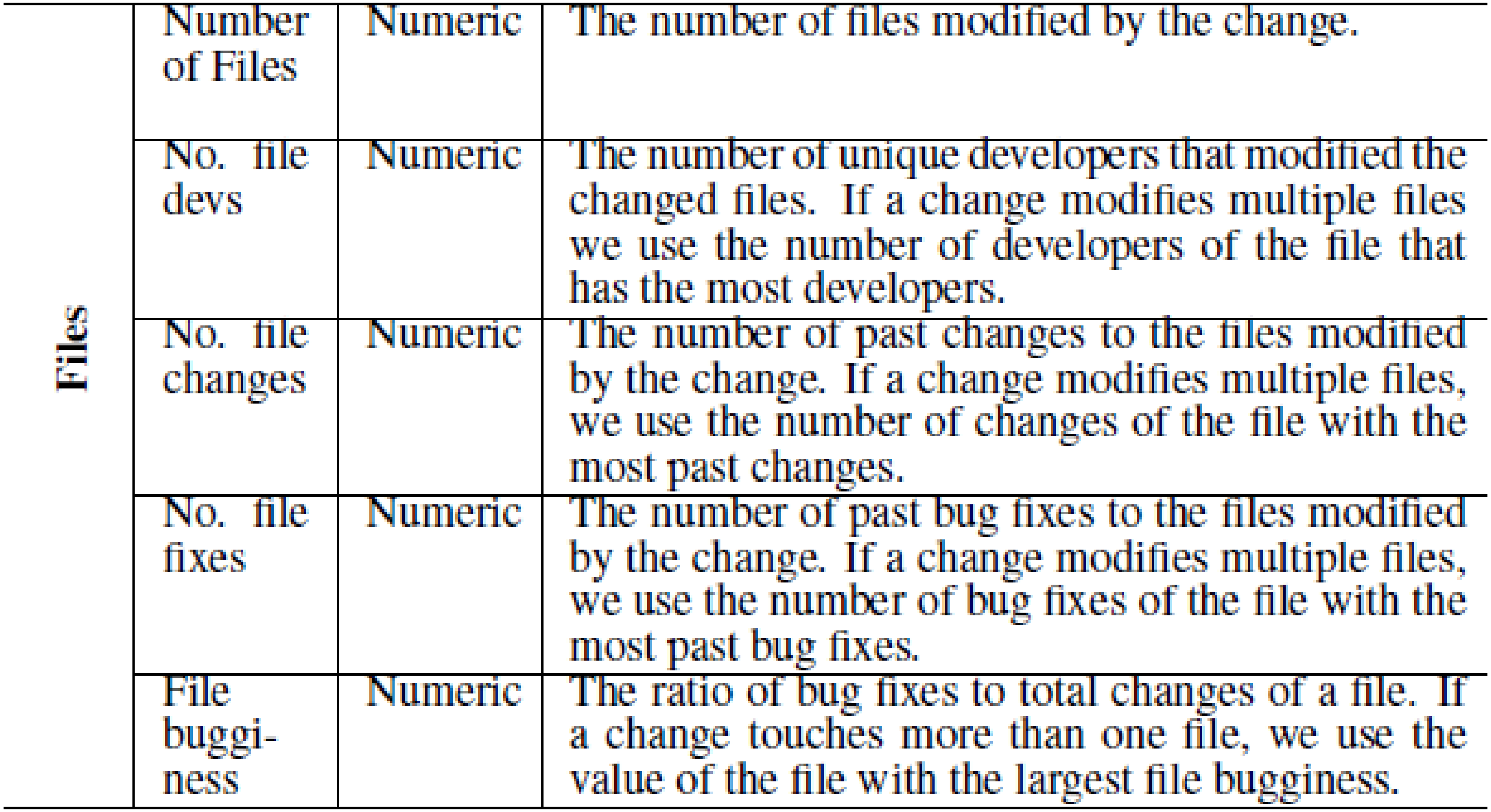
Rationale
Changes performed at certain times in the day, e.g., late afternoons, might be done by over-worked or less aware developers, hence, these changes may be more risky [12].
Changes performed on specific days of the week (e.g., Fridays) are not as carefully examined and might be more risky [33].
Changes performed during specific periods, i.e., beginning, mid or end of the month might be rushed to meet end-of-the-month quotas and are likely to be more risky.
Changes performed in specific months, e.g., later in the year or during holiday months like December, when less developers and expertise are available, might be more risky.

Approach



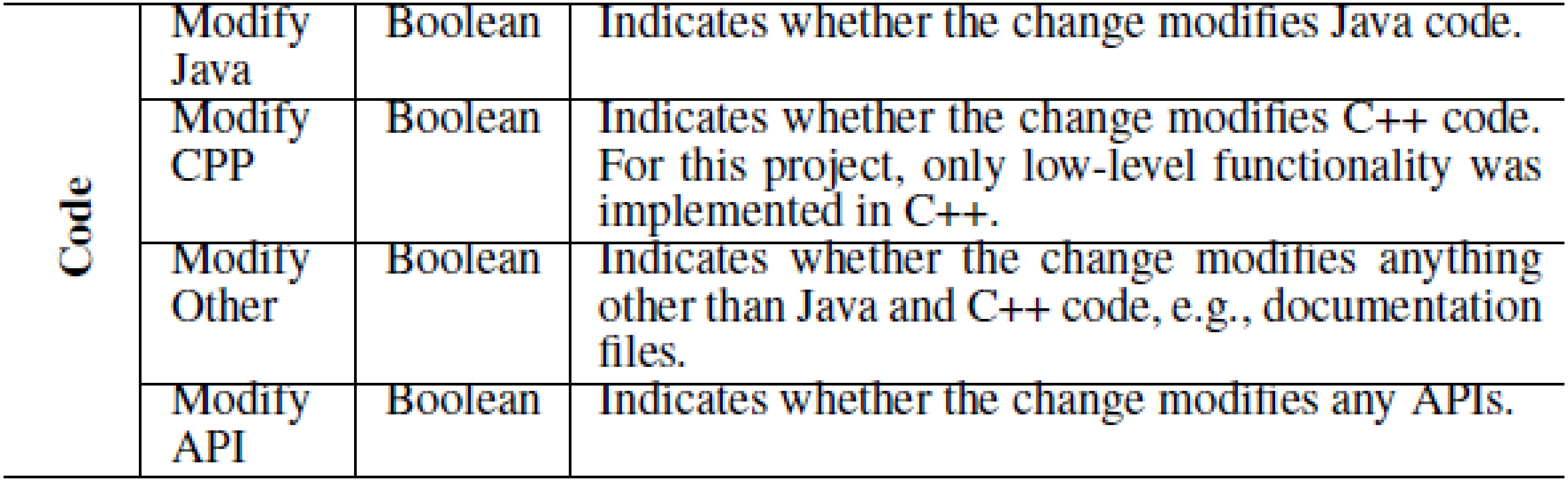
Size	Lines Added	Numeric	The number of lines added as part of the change.
	Chunks Added	Numeric	The number of chunks (i.e., different sections) added as part of the change.
	Lines Deleted	Numeric	The number of lines deleted as part of the change.
	Chunks Deleted	Numeric	The number of chunks (i.e., different sections) deleted as part of the change.
	Lines Modified	Numeric	The number of lines modified as part of the change.
	Chunks Modified	Numeric	The number of chunks (i.e., different sections) modified as part of the change.
	Churn	Numeric	The total number of lines added, deleted and modified as part of the change.

Approach



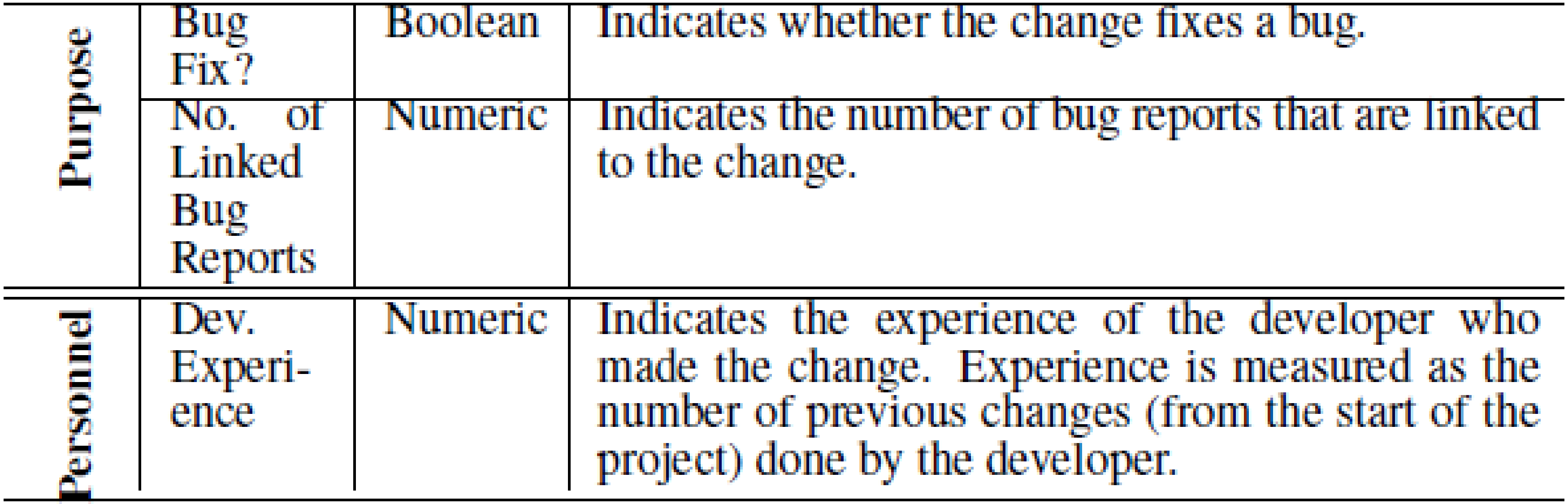
Files	Number of Files	Numeric	The number of files modified by the change.
	No. file devs	Numeric	The number of unique developers that modified the changed files. If a change modifies multiple files we use the number of developers of the file that has the most developers.
	No. file changes	Numeric	The number of past changes to the files modified by the change. If a change modifies multiple files, we use the number of changes of the file with the most past changes.
	No. file fixes	Numeric	The number of past bug fixes to the files modified by the change. If a change modifies multiple files, we use the number of bug fixes of the file with the most past bug fixes.
	File bugginess	Numeric	The ratio of bug fixes to total changes of a file. If a change touches more than one file, we use the value of the file with the largest file bugginess.

Approach



Code	Modify Java	Boolean	Indicates whether the change modifies Java code.
	Modify CPP	Boolean	Indicates whether the change modifies C++ code. For this project, only low-level functionality was implemented in C++.
	Modify Other	Boolean	Indicates whether the change modifies anything other than Java and C++ code, e.g., documentation files.
	Modify API	Boolean	Indicates whether the change modifies any APIs.

Approach



Purpose	Bug Fix?	Boolean	Indicates whether the change fixes a bug.
	No. of Linked Bug Reports	Numeric	Indicates the number of bug reports that are linked to the change.
Personnel	Dev. Experience	Numeric	Indicates the experience of the developer who made the change. Experience is measured as the number of previous changes (from the start of the project) done by the developer.

Approach

- Find that risky changes classification is subjective
- Thus they add two additional features for two kinds of models:
    - Developers based: Add developer name
    - Team base: Add team name

Result

- Ten fold cross validation
- Recall
    - Developer based: 67.6%
    - Team based: 67.9%
- Relative precision
    - Compared with random model
    - Developer based: 1.87x
    - Team based: 1.37x

Other Applications

- Predicting faulty commits
  - Tian Jiang, Lin Tan, Sunghun Kim: Personalized defect prediction. ASE 2013: 279-289
- Refining anomaly reports
  - Lucia, David Lo, Lingxiao Jiang, Aditya Budi: Active refinement of clone anomaly reports. ICSE 2012: 397-407
- Automated fixing of bugs in SQL-like queries
  - Divya Gopinath, Sarfraz Khurshid, Diptikalyan Saha, Satish Chandra: Data-guided repair of selection statements. ICSE 2014: 243-253

Other Applications

- Class diagram summarization
  - Ferdian Thung, David Lo, Mohd Hafeez Osman, Michel R. V. Chaudron: Condensing class diagrams by analyzing design and network metrics using optimistic classification. ICPC 2014: 110-121

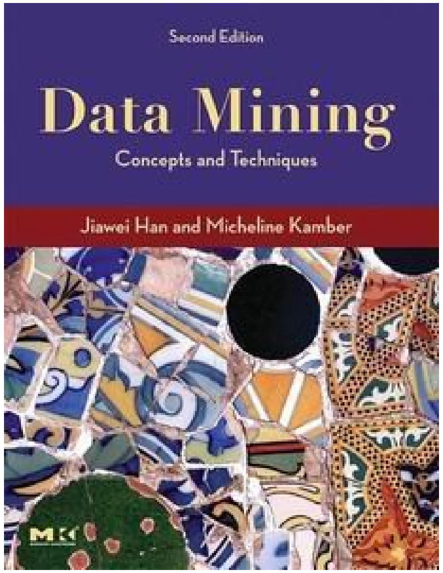
- Predicting effectiveness of automated fault localization tools
  - Tien-Duy B. Le, David Lo: Will Fault Localization Work for These Failures? An Automated Approach to Predict Effectiveness of Fault Localization Tools. ICSM 2013: 310-319

Conclusion

- Part I: Pattern Mining
  - Extract frequent structures from database
  - Structures: Set, Sequence, Graph
  - Application: Find common API patterns
- Part II: Clustering
  - Group similar things together
  - Approaches: k-Means, k-Medoids, Hierarchical, etc.
  - Application: Group traces to reduce inspection cost
- Part III: Classification
  - Predict class label of unknown data
  - Approaches: Decision tree, SVM, kNN, etc.
  - Application: Predict risk of software changes

Acknowledgements & Additional References

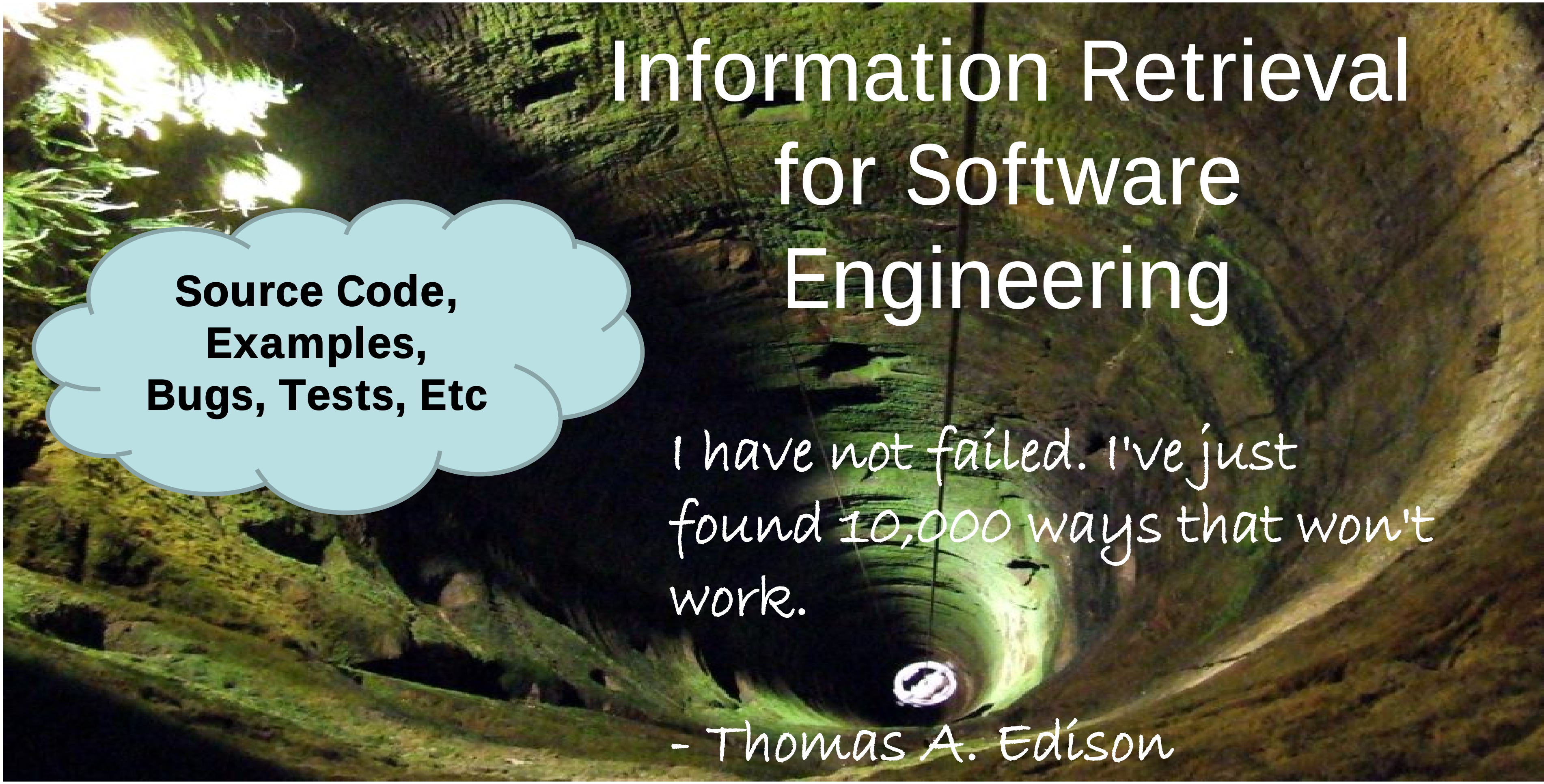
- Many slides and images are taken or adapted from:
  - Resource slides of: Data mining: Concepts and Techniques, 2nd Ed., by Han et al., 2006
  - Ahmed Hassan's and Tao Xie's slides
  - The three research papers mentioned in the slides.



Thank you!

Questions? Comments?

davidlo@smu.edu.sg



Information Retrieval for Software Engineering

Source Code, Examples, Bugs, Tests, Etc

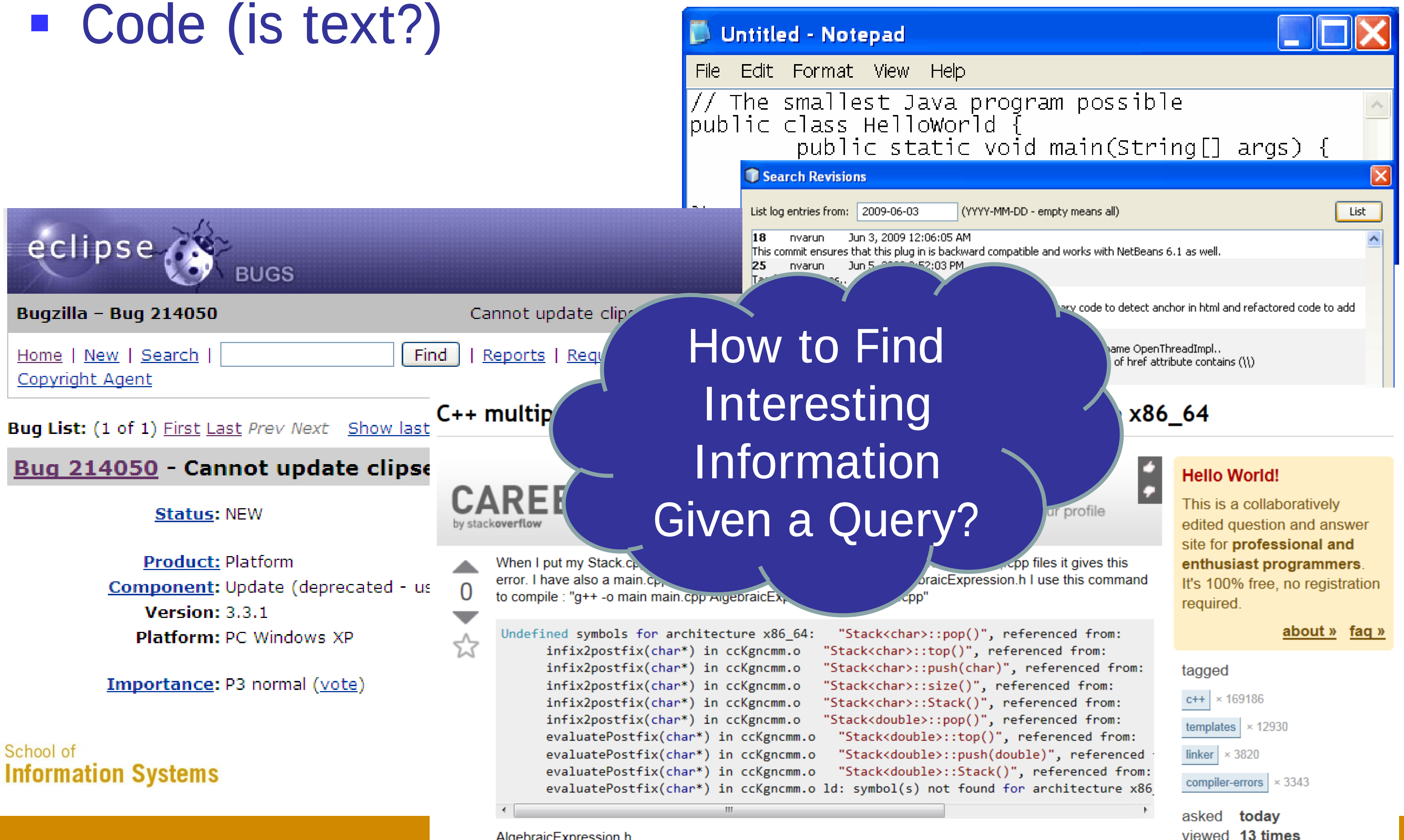
> I have not failed. I've just found 10,000 ways that won't work.
>
> - Thomas A. Edison

Definition

- Information retrieval (IR) is finding material
  - (usually documents)
  - of an unstructured nature (usually text)
  - that satisfies an information need
  - from within large collections

Software Engineering Corpora

- Real text
- Code (is text?)



```
// The smallest Java program possible
public class HelloWorld {
       public static void main(String[] args) {
```

List log entries from: 2009-06-03 (YYYY-MM-DD - empty means all)

18 nvarun Jun 3, 2009 12:06:05 AM
This commit ensures that this plug in is backward compatible and works with NetBeans 6.1 as well.
25 nvarun Jun 5, …:52:03 PM

…ry code to detect anchor in html and refactored code to add

…ame OpenThreadImpl..
… of href attribute contains (\\)

Bugzilla – Bug 214050 Cannot update clip…

Bug List: (1 of 1) First Last Prev Next Show last

Bug 214050 - Cannot update clipse

Status: NEW

Product: Platform

Component: Update (deprecated - us

Version: 3.3.1

Platform: PC Windows XP

Importance: P3 normal (vote)

How to Find Interesting Information Given a Query?

C++ multip… x86_64

When I put my Stack.cp… .cpp files it gives this error. I have also a main.cp… braicExpression.h I use this command to compile : "g++ -o main main.cpp AlgebraicEx… .cpp"

```
Undefined symbols for architecture x86_64:   "Stack<char>::pop()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::top()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::push(char)", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::size()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<char>::Stack()", referenced from:
      infix2postfix(char*) in ccKgncmm.o   "Stack<double>::pop()", referenced from:
      evaluatePostfix(char*) in ccKgncmm.o   "Stack<double>::top()", referenced from:
      evaluatePostfix(char*) in ccKgncmm.o   "Stack<double>::push(double)", referenced 
      evaluatePostfix(char*) in ccKgncmm.o   "Stack<double>::Stack()", referenced from:
      evaluatePostfix(char*) in ccKgncmm.o ld: symbol(s) not found for architecture x86
```

AlgebraicExpression.h

Hello World!

This is a collaboratively edited question and answer site for **professional and enthusiast programmers**. It's 100% free, no registration required.

tagged

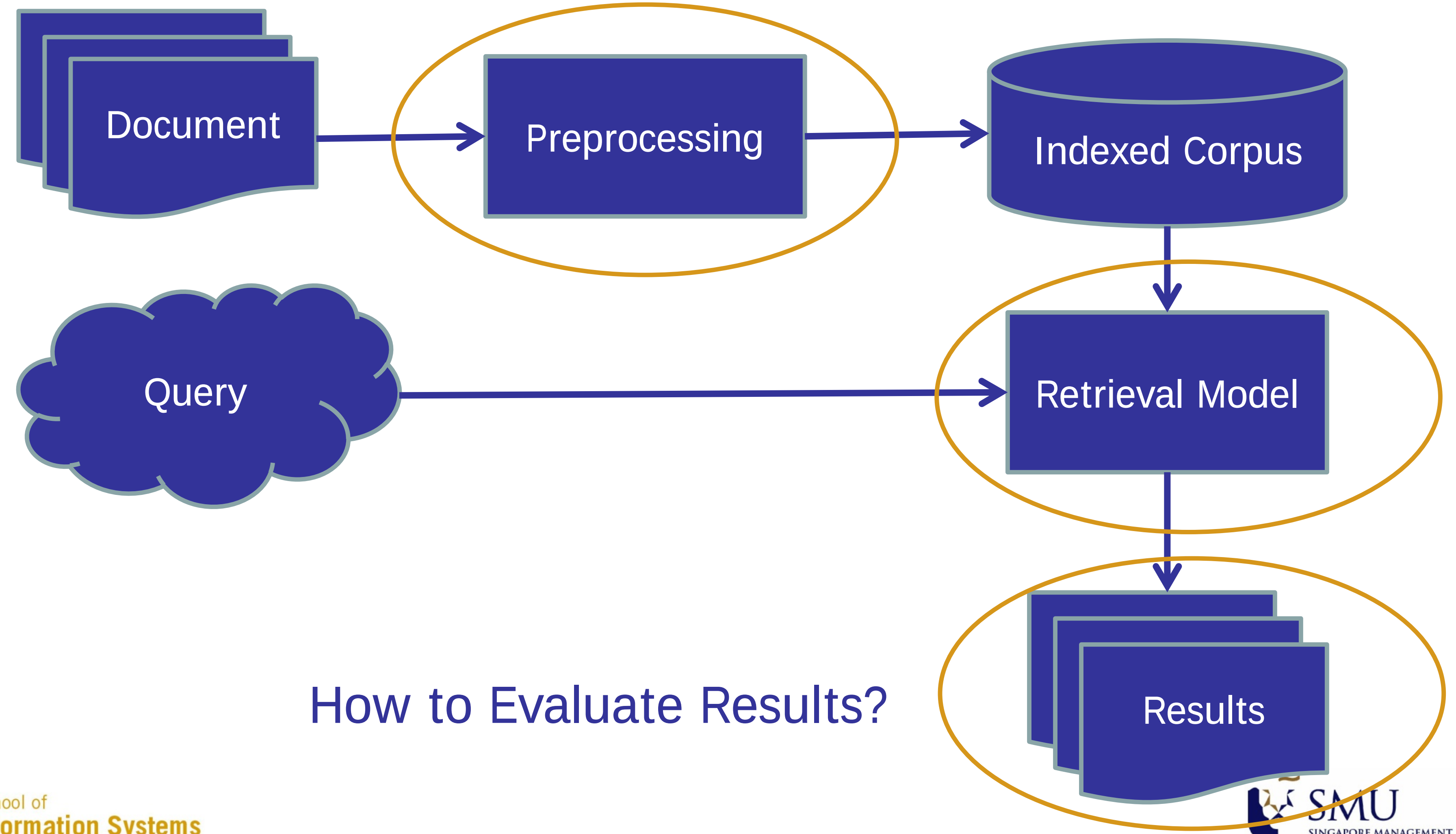
c++	× 169186
templates	× 12930
linker	× 3820
compiler-errors	× 3343

asked **today**
viewed **13 times**

Outline

- I. Preliminaries
  - Preprocessing
  - Retrieval
  - Recent Studies in SE
- II. Vector Space Model
  - Techniques
  - Applications
- III. Language Model
  - Techniques
  - Applications
- IV. Topic Model
  - Techniques
  - Applications
- V. Text Classification
  - Techniques
  - Applications

Part I: Preliminaries



How to Evaluate Results?

Structure

- Preprocessing:
  - Document Boundary & Format
  - Text Preprocessing
  - Code Preprocessing
- Retrieval:
  - Retrieval Model
  - Evaluation Criteria
- Recent Studies in SE

I(A): Preprocessing

Document Boundary & Format

Text Preprocessing

Code Preprocessing

Document Boundary

- What is the document unit ?
  - A file?
  - An email?
  - An email with 5 attachments?
  - A group of files (ppt or latex in HTML)?
  - A method ? A class ?
- Requires some design decisions.

Document Format

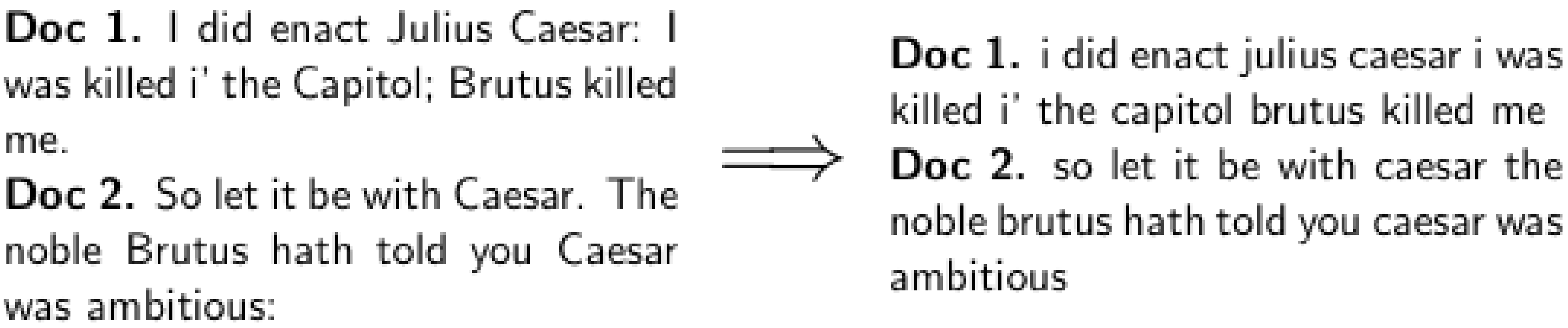
- We need to deal with format and language of each document.
- What format is it in?
  - pdf, word, excel, html, etc.
- What language is it in?
  - English, Java, C#, Chinese, Hindi, etc.
- What character set is in use?

Text Preprocessing

- Tokenization
- Stop-word Removal
- Normalization
- Stemming
- Indexing

Text: Tokenization

- Breaking a document into its constituent tokens or terms
  - In a textual document, a token is typically a word.
- Example (Shakespeare's Play):



Doc 1. I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.
Doc 2. So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

$\Longrightarrow$

Doc 1. i did enact julius caesar i was killed i' the capitol brutus killed me
Doc 2. so let it be with caesar the noble brutus hath told you caesar was ambitious

Text: Stop-Word Removal

- stop words = extremely common words
  - little value in helping select documents matching a user need
  - Examples: a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will, with
- Stop word elimination used to be standard in older IR systems.
  - However, stop words needed for phrase queries, e.g. “president of Singapore”
  - Most web search engines index stop words

Text: Normalization

- Need to normalize terms in indexed text as well as query terms into the same form.
- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.

Text: Stemming

- Definition of stemming: Crude heuristic process that chops off the ends of words to reduce related words to their root form
- Language dependent
- Example: *automate, automatic, automation* all reduce to *automat*

Text: Stemming

- Porter's Algorithm
    - Most commonly used algorithm
    - Five phases of reductions
        - Phases are applied sequentially
    - Each phase consists of a set of commands.
        - Sample command: Delete final *ement* if what remains is longer than 1 character
        - replacement → replac
        - cement → cement

Text: Stemming

- Stemming can increase effectiveness for some queries, and decrease effectiveness for others
- Queries where stemming is likely to help:
  - [wool sweaters], [sightseeing tour singapore]
  - (equivalence classes: {sweater,sweaters}, {tour,tours})
- Queries where stemming hurts:
  - [operational AND research], [operating AND system], [operative AND dentistry]

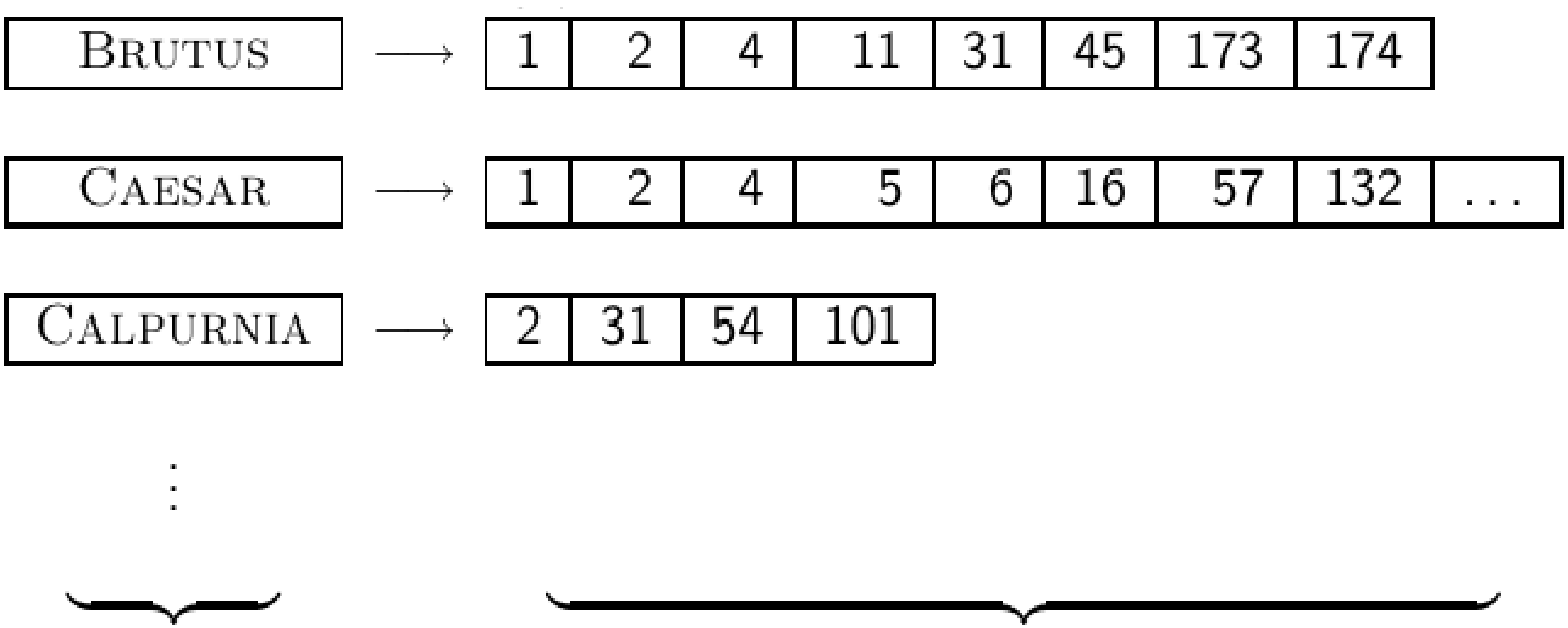
Text: Stemming

- Other stemming algorithms:
  - Lovins stemmer
  - Paice stemmer
  - Etc.

Text: Indexing

- Inverted Index

For each term *t*, we store a list of all documents that contain *t*.



BRUTUS → 1 | 2 | 4 | 11 | 31 | 45 | 173 | 174

CAESAR → 1 | 2 | 4 | 5 | 6 | 16 | 57 | 132 | ...

CALPURNIA → 2 | 31 | 54 | 101

⋮

dictionary

documents

Text: Indexing

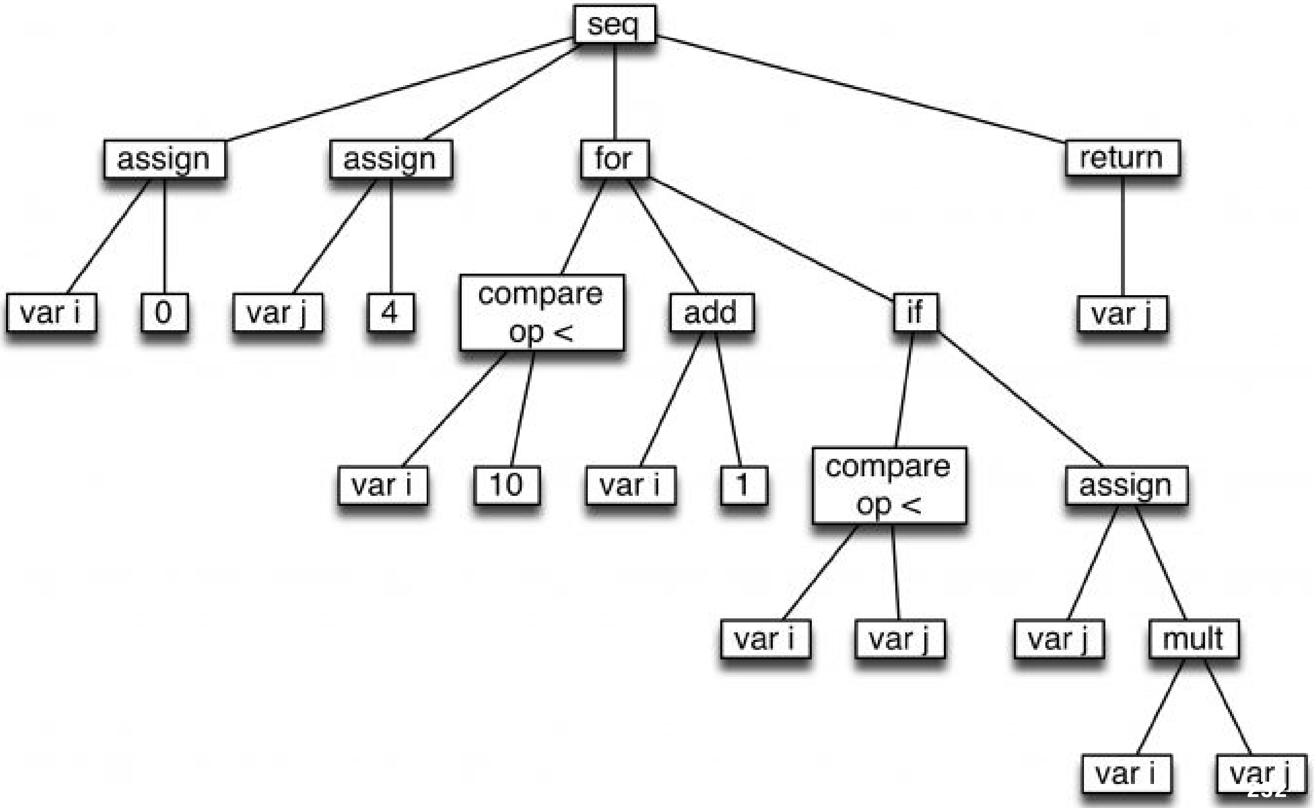
- Bi-word index
  - Index every consecutive pair of terms in the text as a phrase.
- k-gram index
- Positional index
- etc.

Code Preprocessing

- Parsing
- Identifier Extraction
- Identifier Tokenization

Code: Parsing

- Creating an abstract syntax tree of the code.
- Identify which ones are variable names, which ones are method calls, etc.
- Difficulties: Multiple languages, partial code
- Tools:
  - ANTLR
  - WALA



Code: Identifier Extraction

- Extract the names of identifiers in the code.
  - Method names
  - Variable names
  - Parameter names
  - Class names
- Extract the comments in the code
- Extract string literals in the code
- How about if/loop/switch structures ?

Code: Identifier Tokenization

- Break identifier names into tokens.
  - printLine => print line
  - System.out.println => system out println
- Many identifier names are in camel casing

- Why do we need to break identifier names?
- Do all identifiers need to be broken?

I(B): Retrieval

Retrieval Model
Evaluation Metrics

Retrieval Model

- Vector Space Model
  - Model documents and queries as a vector of values
- Language Model
  - Model documents and/or queries by a probability distribution
    - Probability for it to generate a word, a sequence of words, etc.
- Topic Model
  - Model documents and queries by a set of topics, where a topic is a set of words

Evaluation Metrics - 1

- Unranked evaluation
- Precision (P) is the fraction of retrieved documents that are relevant

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

- Recall (R) is the fraction of relevant documents that are retrieved

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

Evaluation Metrics - 2

- Ranked evaluation
- P-R Curve
  - Compute precision and recall for each “prefix”
    - top 1, top 2, top 3, top 4 etc results
  - Produces a precision-recall curve.
- Mean Average Precision (MAP)
  - Average precision for the top k documents
    - each time a relevant doc is retrieved
  - Averaged over all queries

$$\mathrm{MAP}(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\mathrm{Precision}(R_{jk})$$

I(C): Recent Studies in SE

Identifier Expansion

Recent Studies in SE

- Dawn Lawrie, David Binkley: Expanding identifiers to normalize source code vocabulary. ICSM 2011: 113-122
- Dave Binkley, Dawn Lawrie, Christopher Uehlinger: Vocabulary normalization improves IR-based concept location. ICSM 2012: 588-591

Expanding Identifiers: Introduction

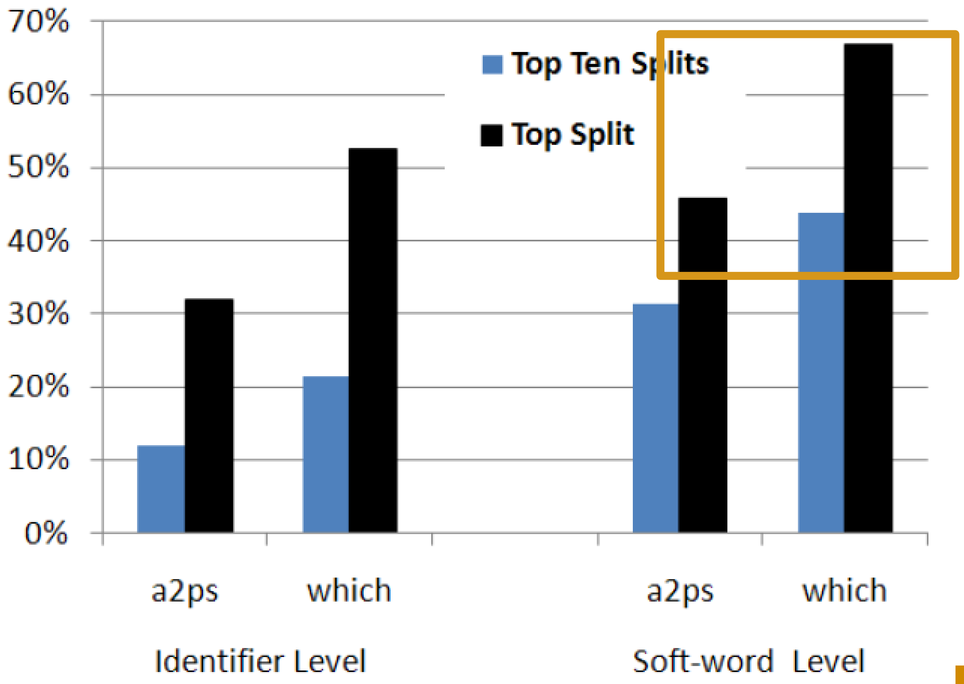
- Language used in code and other documents must be standardized for effective retrieval.
- A significant proportion of invented vocabulary.
- To standardize:
    - Split an identifier into parts
    - Expand the identifier into a word
- Closer to queries expressed in human language

Expanding Identifiers: Approach (Nutshell)

- Break an identifiers into many possible splits
- For each possible split, expand the identifier parts
  - Expand each part by adding wildcard characters
  - See if any of the resultant regular expression match any dictionary word surrounding the identifier
- Find the best possible split and expansion
  - Criterion: Maximize similarity of expanded parts
  - Measure word-similarity based on co-occurrence
    - Trained on a dataset of over one trillion words collected by Google

Expanding Identifiers: Accuracy

- Compared with manual expansion of identifiers
- Variants: Top-1 or Top-10 splits
- Accuracy criteria:
  - Identifier match: % of identifiers correctly expanded
  - Word match: % of identifier parts correctly expanded



Expanding Identifiers: Application in Retrieval

Feature Location:

Queries (Feature Description) -> Relevant Code Units

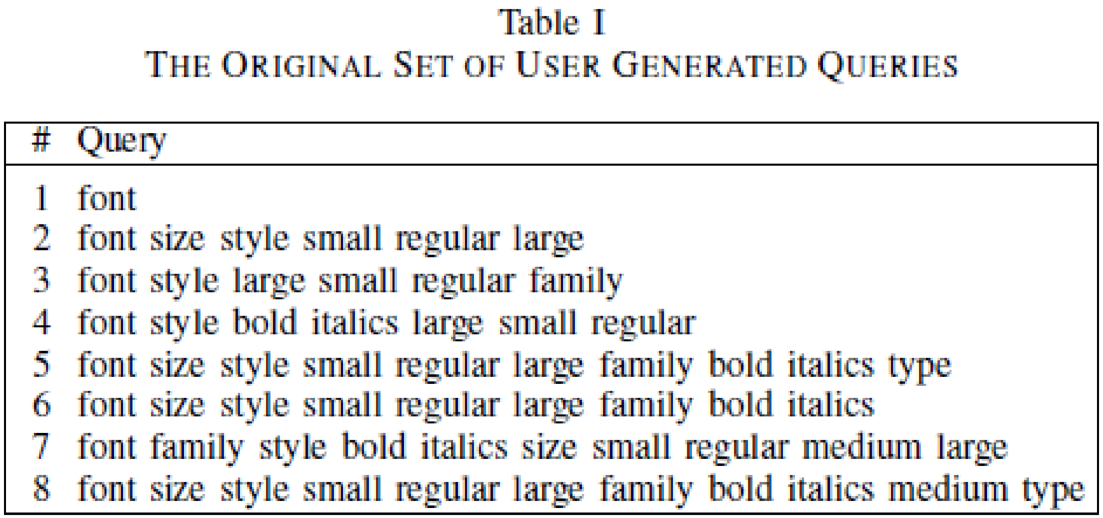


Table I
THE ORIGINAL SET OF USER GENERATED QUERIES

#	Query
1	font
2	font size style small regular large
3	font style large small regular family
4	font style bold italics large small regular
5	font size style small regular large family bold italics type
6	font size style small regular large family bold italics
7	font family style bold italics size small regular medium large
8	font size style small regular large family bold italics medium type

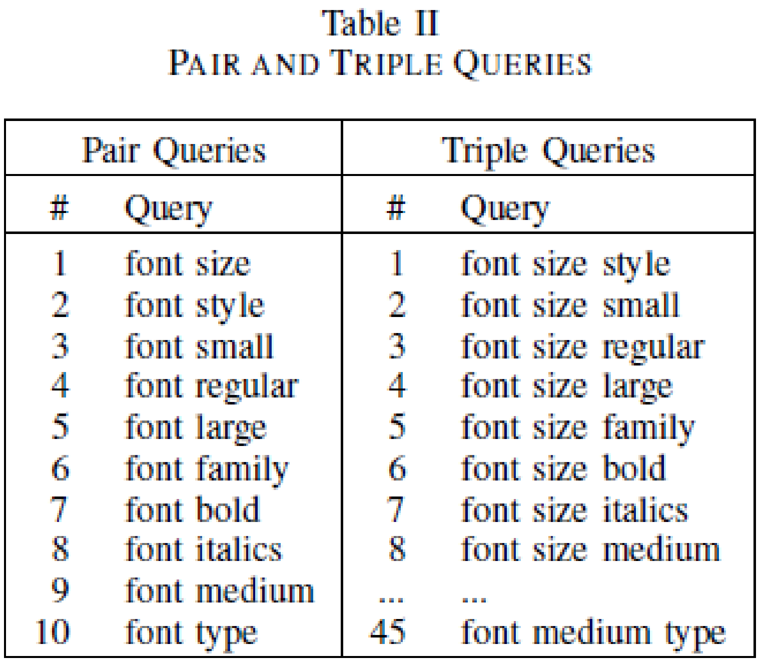
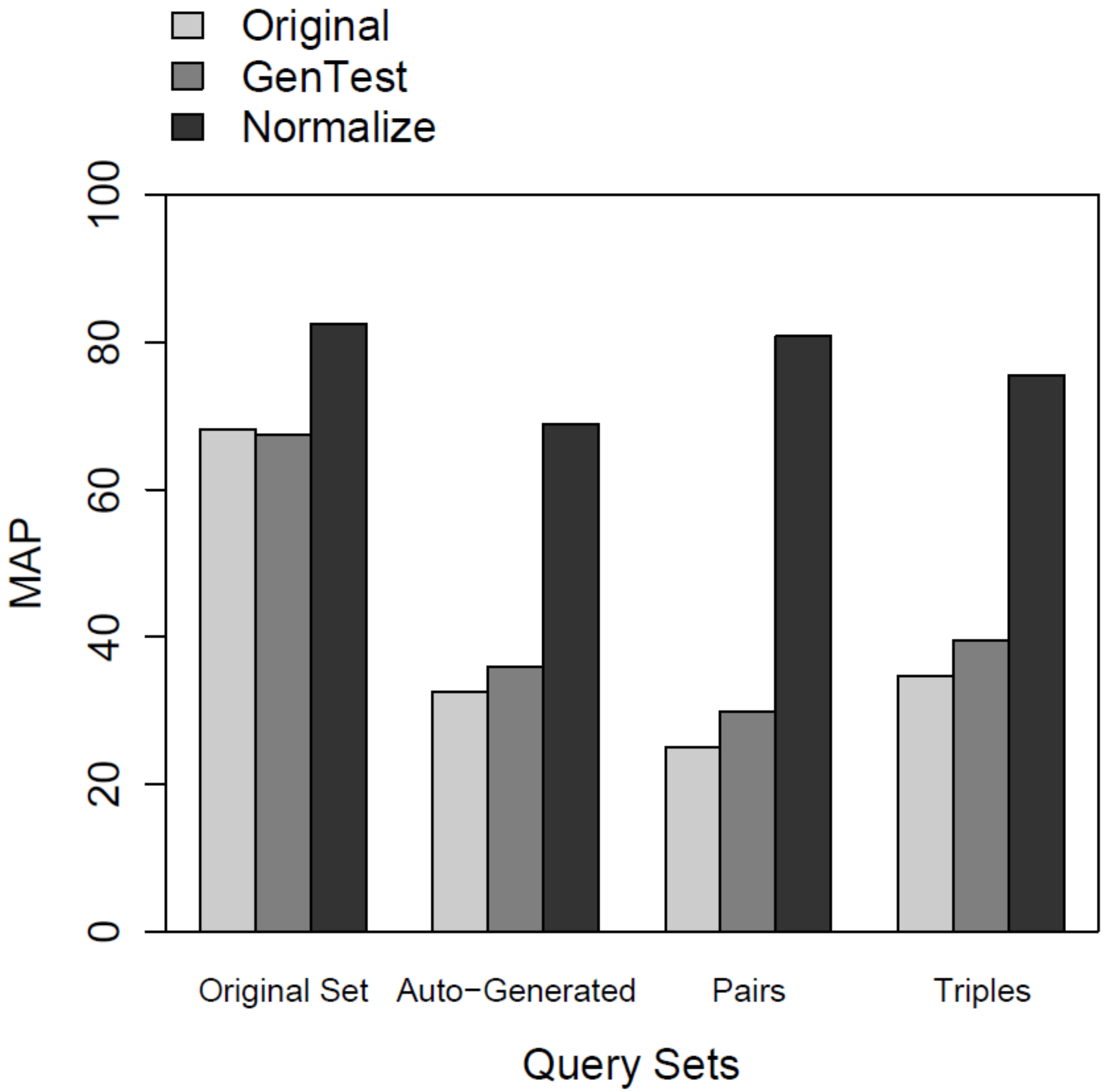


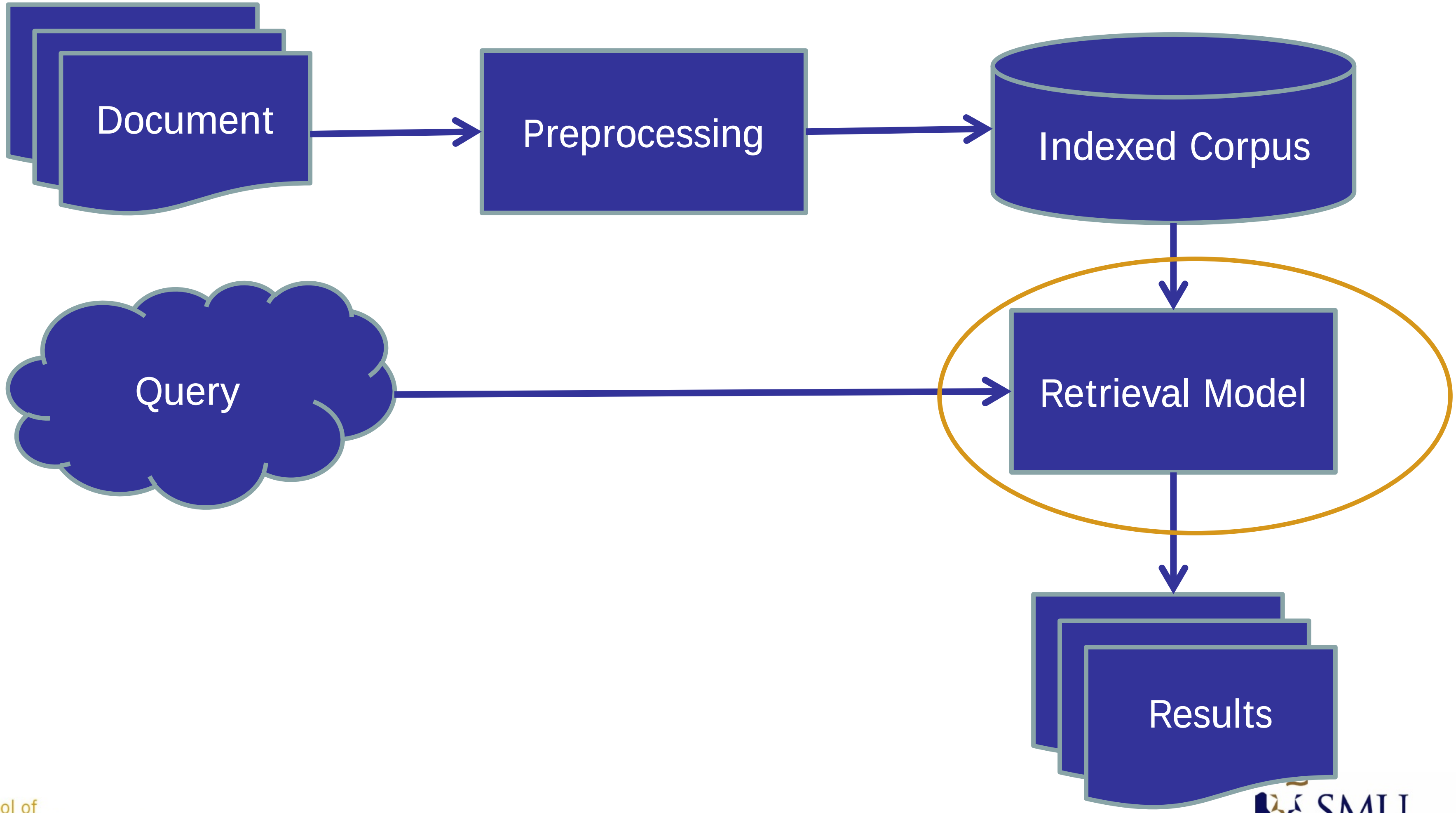
Table II
PAIR AND TRIPLE QUERIES

Pair Queries		Triple Queries	
#	Query	#	Query
1	font size	1	font size style
2	font style	2	font size small
3	font small	3	font size regular
4	font regular	4	font size large
5	font large	5	font size family
6	font family	6	font size bold
7	font bold	7	font size italics
8	font italics	8	font size medium
9	font medium	...	...
10	font type	45	font medium type

Expanding Identifiers: Application in Retrieval



Summary: Retrieval Process



Document

Preprocessing

Indexed Corpus

Query

Retrieval Model

Results

Part II - Vector Space Model (VSM)

- Model documents and queries as a vector of values
- Retrieval is done by computing similarities of:
  - Document and queries
  - In the vector space
- Questions:
  - What are the appropriate vectors of values that represent documents?
  - How to compute similarities between two vector-based representations of documents?

Structure

- Techniques
  - Document Representation
    - Bag-of-Word Model
    - Term Frequency (TF)
    - Inverse Document Frequency (IDF)
    - Other TF-IDF Variants
  - Retrieval using VSM
- Applications
  - Duplicate Bug Report Detection
  - Other Applications

II(A): Techniques

Bag-of-Word Model
Term Frequency (TF)
Inverse Document Frequency (IDF)
Other TF-IDF Variants
Retrieval using VSM

Bag of Words Consideration

- It considers a document as a multi-set of its constituent words (or terms).
- We do not consider the order of words in a document.
  - John is quicker than Mary, and
  - Mary is quicker than John

  are represented the same way.

VSM: Term Weighting (TF)

- Not all words/terms equally characterize a document.
  - If term t appears more times in a document d, that term is more relevant to d
- We denote the number of times that a term t occurs in a document d as $tf_{t,d}$
  - We refer to this as term frequency (TF)

VSM: Term Weighting (IDF)

- Rare terms are more informative than frequent terms
  - Recall stop words
- We want a high weight for rare terms
  - Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
  - A document containing this term is very likely to be relevant to the query *arachnocentric*

VSM: Term Weighting (IDF)

- To do this we make use of inverse document frequency (IDF):

$$\mathrm{idf}_t = N/\mathrm{df}_t$$

  - N is the total number of documents in the corpus
  - df_t is the document frequency of t
    - the number of documents that contain t
    - df_t is an inverse measure of the informativeness of t
    - $\mathrm{df}_t \leq N$

VSM: Term Weighting (TF-IDF)

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

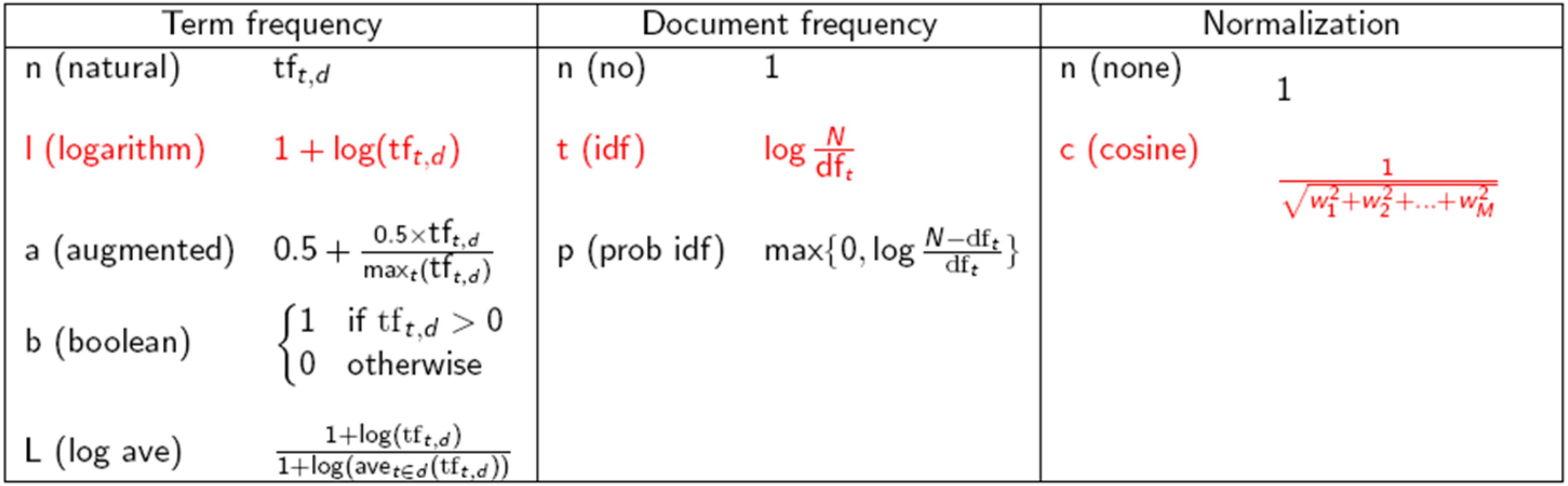
$$\mathrm{w}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

  - Increases with the number of occurrences within a document
  - Increases with the rarity of the term in the collection

VSM: Document Representation

- Each document and each query is characterized as a vector of terms weights
- So we have a |V|-dimensional vector space
  - V = set of all terms
  - Terms are dimensions of the space
  - Documents are points in this space

VSM: TF-IDF Variants



Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+...+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$		
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$				
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

VSM: Retrieval

- Represent documents and queries as vectors
- Compute the similarity between the vectors
- Cosine similarity is normally used:

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

qi is the tf-idf weight of term i in the query
di is the tf-idf weight of term i in the document

- Return top-k most similar documents

II(B): Applications

An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information

Duplicate Bug Report Detection

- Xiaoyin Wang, Lu Zhang, Jiasu Sun, Peking University, China
- Tao Xie, North Carolina State University, USA
- John Anvik, University of Victoria, Canada

- Published in ACM/IEEE International Conference on Software Engineering (ICSE), 2008

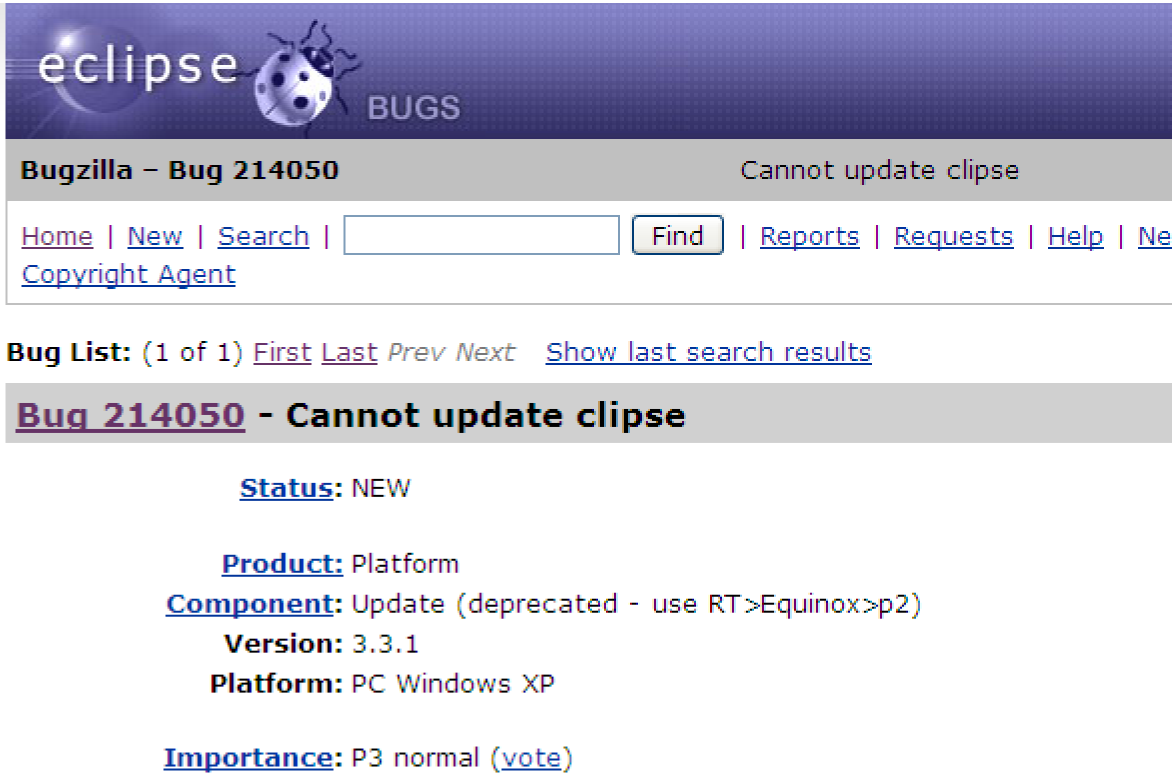
Duplicate Bug Reports: Motivation

- To improve quality of software systems, often developers allow users to report bugs.
- Bug reporting is inherently an uncoordinated distributed process.
    - A number of reports of the same defect/bug are often made by different users.
    - This lead to a problem of duplicate bug reports.

Duplicate Bug Reports: Motivation

- In practice, a special developer (a triager) is often assigned to detect duplicate reports.
  - Number of bug reports are often too many for developers to handle.
- A (semi) automated solution is needed.

Duplicate Bug Reports: Dataset



Bugzilla – Bug 214050 Cannot update clipse

Bug List: (1 of 1) First Last *Prev* *Next* Show last search results

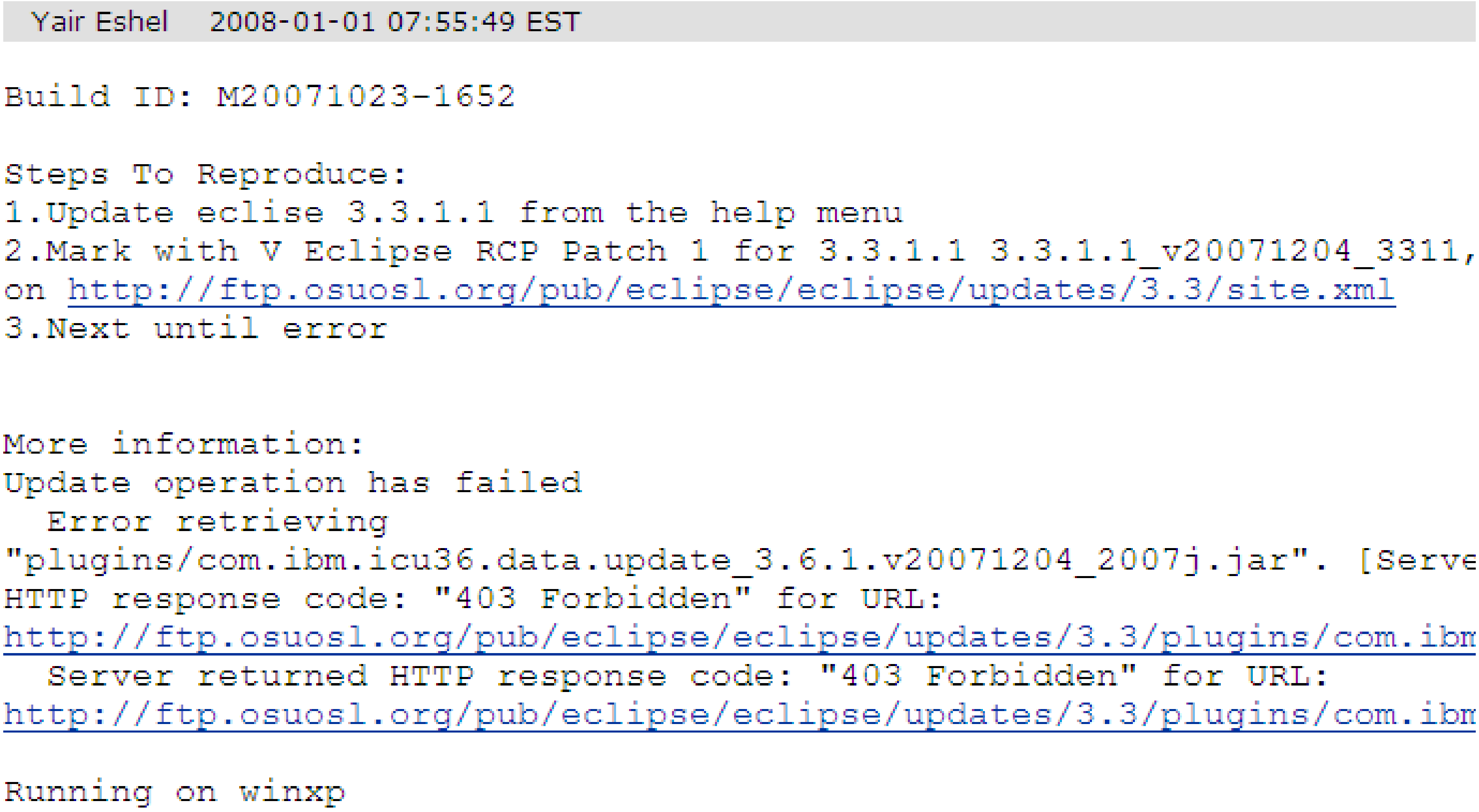
Bug 214050 - Cannot update clipse

Status: NEW

Product: Platform
Component: Update (deprecated - use RT>Equinox>p2)
Version: 3.3.1
Platform: PC Windows XP

Importance: P3 normal (vote)

Duplicate Bug Reports: Dataset



Yair Eshel 2008-01-01 07:55:49 EST

```
Build ID: M20071023-1652

Steps To Reproduce:
1.Update eclise 3.3.1.1 from the help menu
2.Mark with V Eclipse RCP Patch 1 for 3.3.1.1 3.3.1.1_v20071204_3311,
on http://ftp.osuosl.org/pub/eclipse/eclipse/updates/3.3/site.xml
3.Next until error


More information:
Update operation has failed
  Error retrieving
"plugins/com.ibm.icu36.data.update_3.6.1.v20071204_2007j.jar". [Serve
HTTP response code: "403 Forbidden" for URL:
http://ftp.osuosl.org/pub/eclipse/eclipse/updates/3.3/plugins/com.ibm
  Server returned HTTP response code: "403 Forbidden" for URL:
http://ftp.osuosl.org/pub/eclipse/eclipse/updates/3.3/plugins/com.ibm

Running on winxp
```

Duplicate Bug Reports: Dataset

- Text Data
    - Summary
        - Concise text
    - Description
        - Longer text
- Execution Traces
    - One execution trace for each bug that exhibits the error

Duplicate Bug Reports: Technique

- Modeling text information
  - Take summary and description of bug reports
  - Perform preprocessing
    - Tokenization
    - Stemming
    - Stop-word removal
  - Create a vector of term weights using:

$$w_i = tf_i \times idf_i$$

$$idf_i = log\ (Dsum\ /\ Dw_i)$$

Dsum = Total number of documents
Dwi = Number of documents containing term i

Duplicate Bug Reports: Technique

- Modeling trace information
  - Take method calls that appear in the execution trace
  - Treat each method as a word
    - Use canonical signature of a method
      - Differentiate overloaded methods
  - Model it in similar way as text information
    - Each method tf is either 0 or 1
      - Ignore repeated method calls
  - At the end, we have a vector of method weights

Duplicate Bug Reports: Technique

- Computing similarity
  - Use cosine similarity of two vectors:

$$Sim = \frac{\sum_{i=1}^{n} w_{1i} w_{2i}}{\sqrt{\sum_{i=1}^{n} w_{1i}^{2} \times \sum_{i=1}^{n} w_{2i}^{2}}}$$

- Need to combine textual and trace information:

$$SIM_{combined} = \frac{SIM_{nlp} + SIM_{exe}}{2}$$

Duplicate Bug Reports: Technique

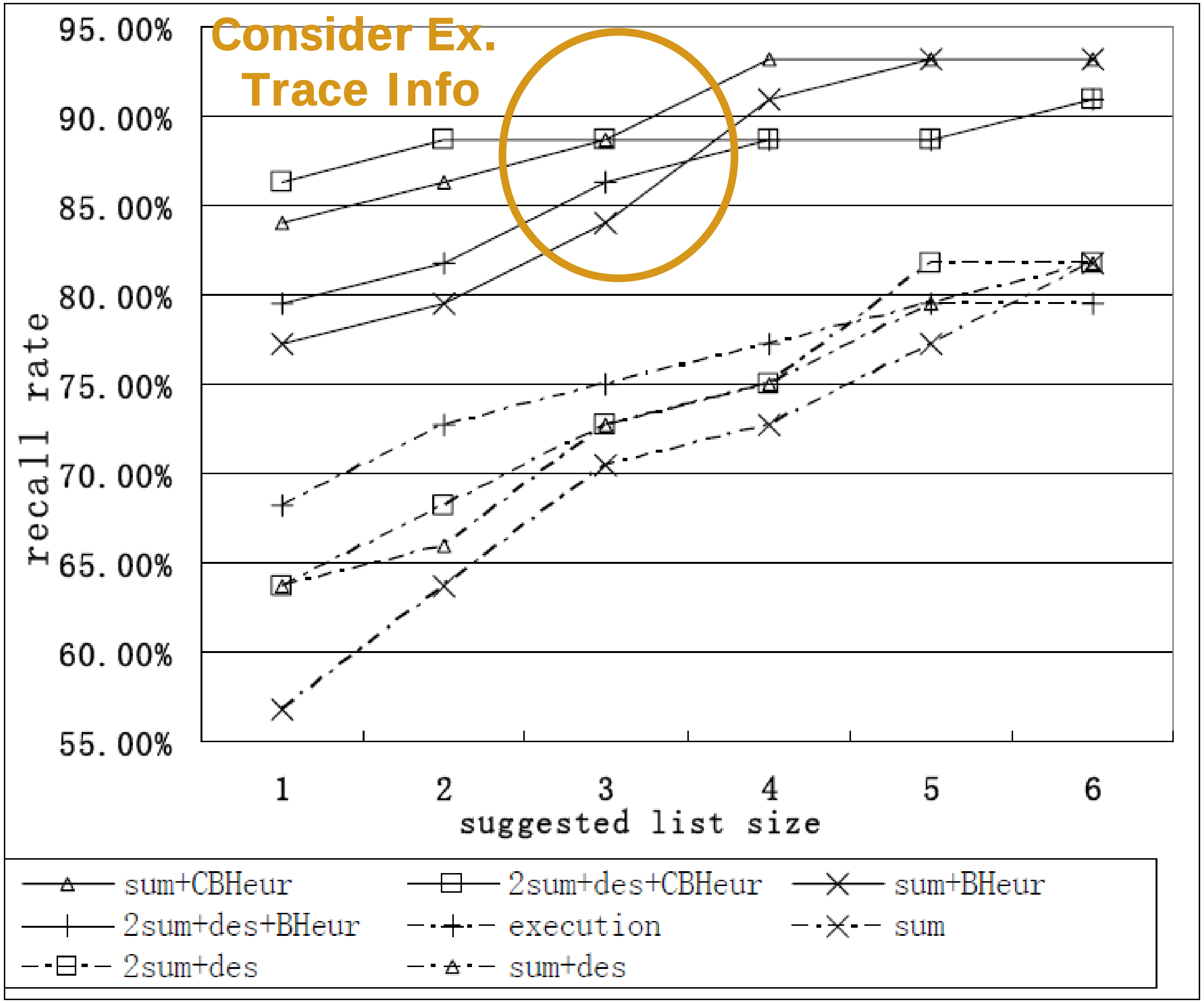
- Given a new bug report
- Return the top-k most similar bug reports that have been reported before

Duplicate Bug Reports: Experiments

$$recall\ rate = \frac{N_{recalled}}{N_{tatal}}$$

- $N_{recalled}$ = Number of duplicate reports whose duplicate is detected in the top-k list
- N_{total} = Number of duplicate reports considered

Duplicate Bug Reports: Experiments



Consider Ex. Trace Info

Other Applications

- Finding buggy files given bug descriptions
  - Shaowei Wang, David Lo: Version history, similar report, and structure: putting them together for improved bug localization. ICPC 2014: 53-63

- Tracing high-level to low-level requirements
  - Jane Huffman Hayes, Alex Dekhtyar, Senthil Karthikeyan Sundaram, Sarah Howard: Helping Analysts Trace Requirements: An Objective Look. RE 2004: 249-259

Other Applications

- Recommending relevant methods to use
  - Ferdian Thung, Shaowei Wang, David Lo, Julia L. Lawall: Automatic recommendation of API methods from feature requests. ASE 2013: 290-300

- Locating code that corresponds to a particular feature
  - Wei Zhao, Lu Zhang, Yin Liu, Jiasu Sun, Fuqing Yang: SNIAFL: Towards a static noninteractive approach to feature location. ACM Trans. Softw. Eng. Methodol. 15(2): 195-226 (2006)

Other Applications

- Semantic search engine to find answers from software forums
  - Swapna Gottipati, David Lo, Jing Jiang: Finding relevant answers in software forums. ASE 2011: 323-332

Part III - Language Model

- Model a document as a probability distribution
    - Able to compute the probability of a query to belong to the document
- Rank document based on the probability of the query to belong to the document

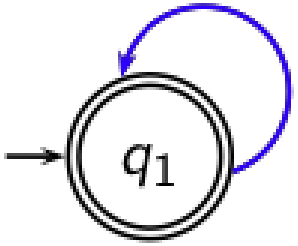
Structure

- Techniques
  - Unigram Language Model
  - Language Model for IR
  - Parameter Estimation
  - Smoothing
- Applications
  - Code Auto-Completion
  - Other Applications

III(A): Techniques

Unigram Language Model
Language Model for IR
Parameter Estimation
Smoothing

Unigram Language Model

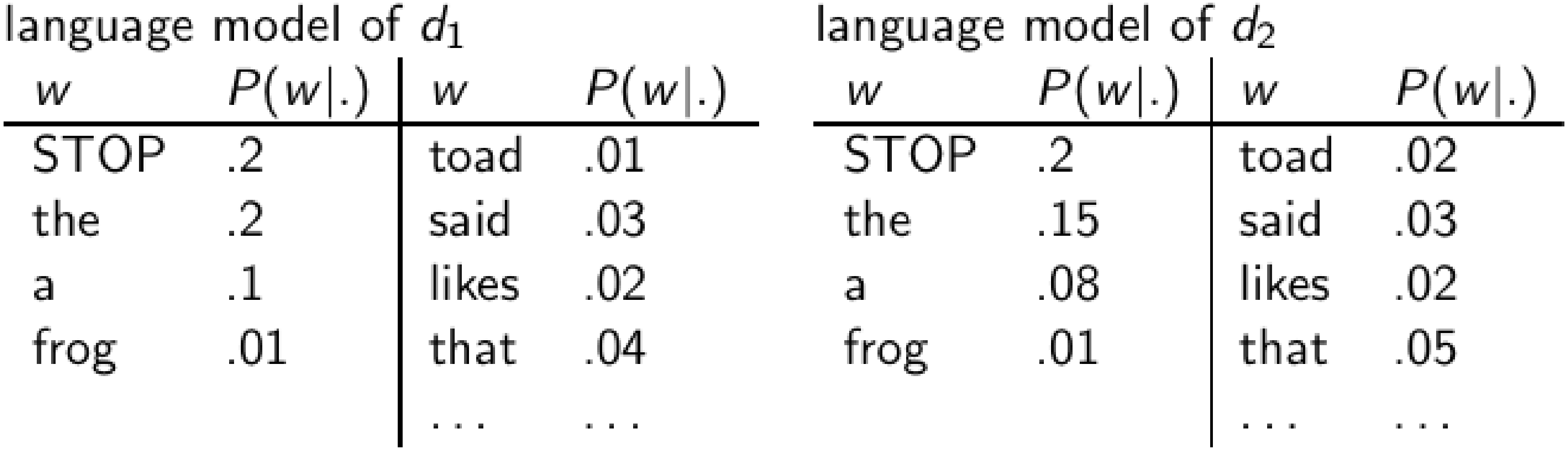


w	$P(w\|q_1)$	w	$P(w\|q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

- One-state probabilistic finite-state automaton
- State emission distribution for its one state q1
- STOP is a special symbol indicating that the automaton stops
- string = “frog said that toad likes frog STOP”

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.0000000000048$$

A different language model for each document



language model of d_1

w	$P(w\|.)$	w	$P(w\|.)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
		...	...

language model of d_2

w	$P(w\|.)$	w	$P(w\|.)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
		...	...

string = “frog said that toad likes frog STOP “

$P(string|Md1) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 4.8 \cdot 10^{-12}$

$P(string|Md2) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 12 \cdot 10^{-12}$

$P(string|Md1) < P(string|Md2)$

Thus, document d2 is “more relevant” to the string than d1 is.

Using language models in IR

- Each document d is represented by a language model Md
- Given a query q
  - Rank documents based on P(q|Md)
- How do we compute P(q|Md)?

How to compute P(q|Md)

- Make conditional independence assumption:

$$P(q|M_d) = P(\langle t_1, \ldots, t_{|q|} \rangle | M_d) = \prod_{1 \leq k \leq |q|} P(t_k|M_d)$$

(|q|: length of q; tk : the token occurring at position k in q)

- This is equivalent to:

$$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\mathrm{tf}_{t,q}}$$

  - $\mathrm{tf}_{t,q}$: term frequency (# occurrences) of t in q

Parameter estimation

- Missing piece: Where do the parameters P(t|Md) come from?
- Use the following estimate: $$\hat{P}(t|M_d) = \frac{\mathrm{tf}_{t,d}}{|d|}$$

  (|d|: length of d; $\mathrm{tf}_{t,d}$: # occurrences of t in d)

- We have a problem with zeros
  - A single t with P(t|Md) = 0 will make $P(q|M_d) = \prod P(t|M_d)$ zero
  - We would give a single term “veto power”

- We need to smooth the estimates to avoid zeros

Smoothing

- Key intuition: A non occurring term is possible (even though it didn't occur), . . .
- . . . but no more likely than would be expected by chance in the collection

- We will use $\hat{P}(t|M_c)$ to "smooth" P(t|Md) away from zero

$$\hat{P}(t|M_c) = cf_t/T$$

  - Mc: the collection model;
  - cf_t: the number of occurrences of t in the collection;
  - $T = \sum_t \mathrm{cf}_t$: the total number of tokens in the collection

Mixture Model

- $P_{mix}(t|Md) = \lambda P(t|Md) + (1 - \lambda)P(t|Mc)$
    - Mixes the probability considering the document with the probability considering the collection.

- High value of λ: “conjunctive-like” search – tends to retrieve documents containing all query words.
- Low value of λ: more disjunctive, suitable for long queries
- Correctly setting λ is very important for good performance.

Example

- Collection: d1 and d2
  - d1 : Jackson was one of the most talented entertainers of all time
  - d2 : Michael Jackson anointed himself King of Pop
- Query q: Michael Jackson

- Use mixture model with $\lambda = 1/2$
  - $P(q|d1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$
  - $P(q|d2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$
- Ranking: d2 > d1

Other Models

- Bigram model
- K-L model
- Other models

III(B): Applications

On the Naturalness of Software

Code Auto-Completion

- Abram Hindle, Earl T. Barr, Zhendong Su, Mark Gabel, Premkumar T. Devanbu, University of California, Davis, USA

- Published in ACM/IEEE International Conference on Software Engineering (ICSE), 2012

Naturalness of Software: Introduction

- Natural language is often repetitive and predictable
    - Can be modeled by a language model
- Is software code like natural language?
- If it is could we exploit the naturalness of code?

Naturalness of Software: Technique

- k-gram language model:
  - Token occurrences are influenced only by the previous k-1 tokens

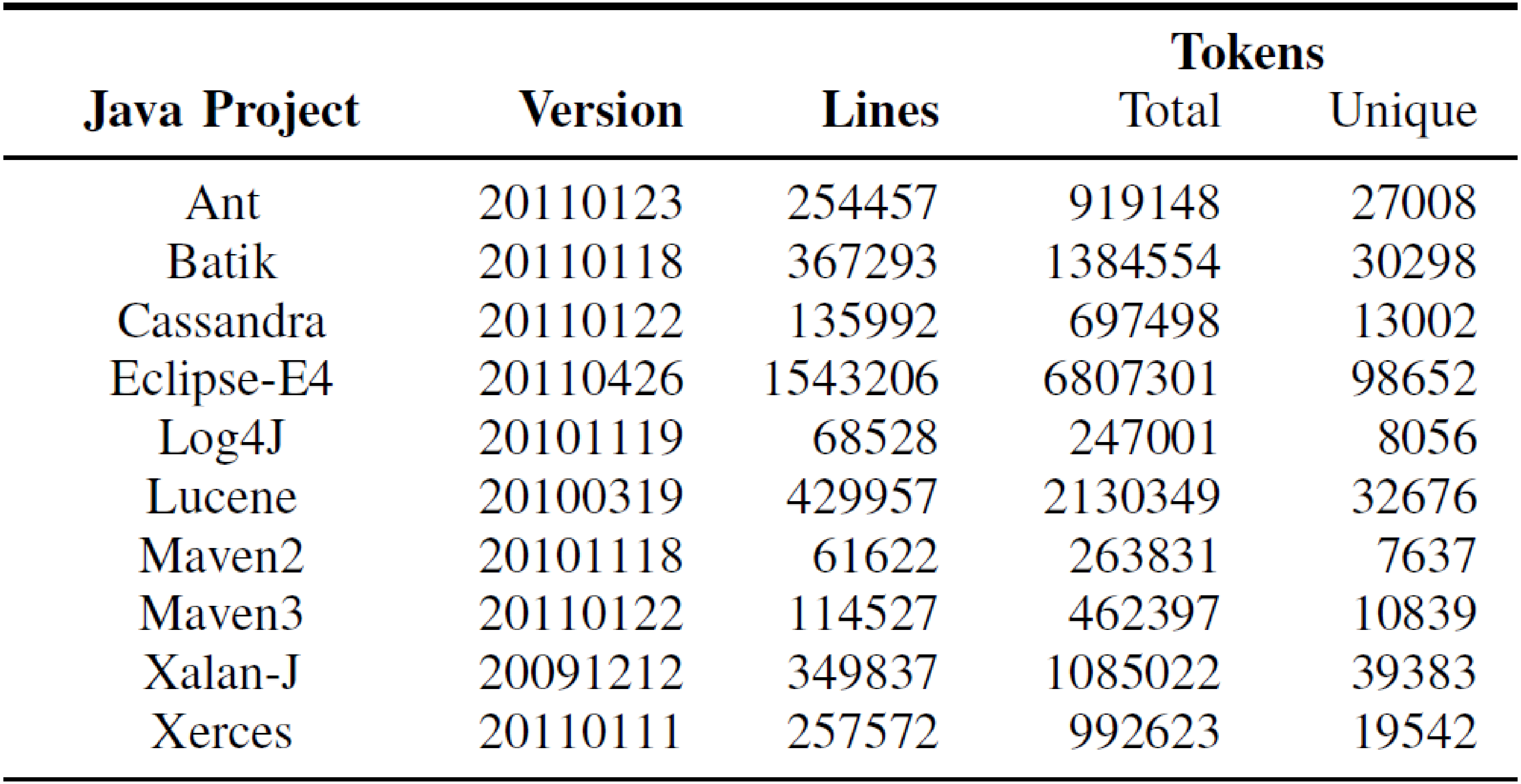
- For a 4-gram language model:

$$p(a_i|a_1 \ldots a_{i-1}) \simeq p(a_i \mid a_{i-3}a_{i-2}a_{i-1})$$

- Maximum Likelihood Estimate (MLE):

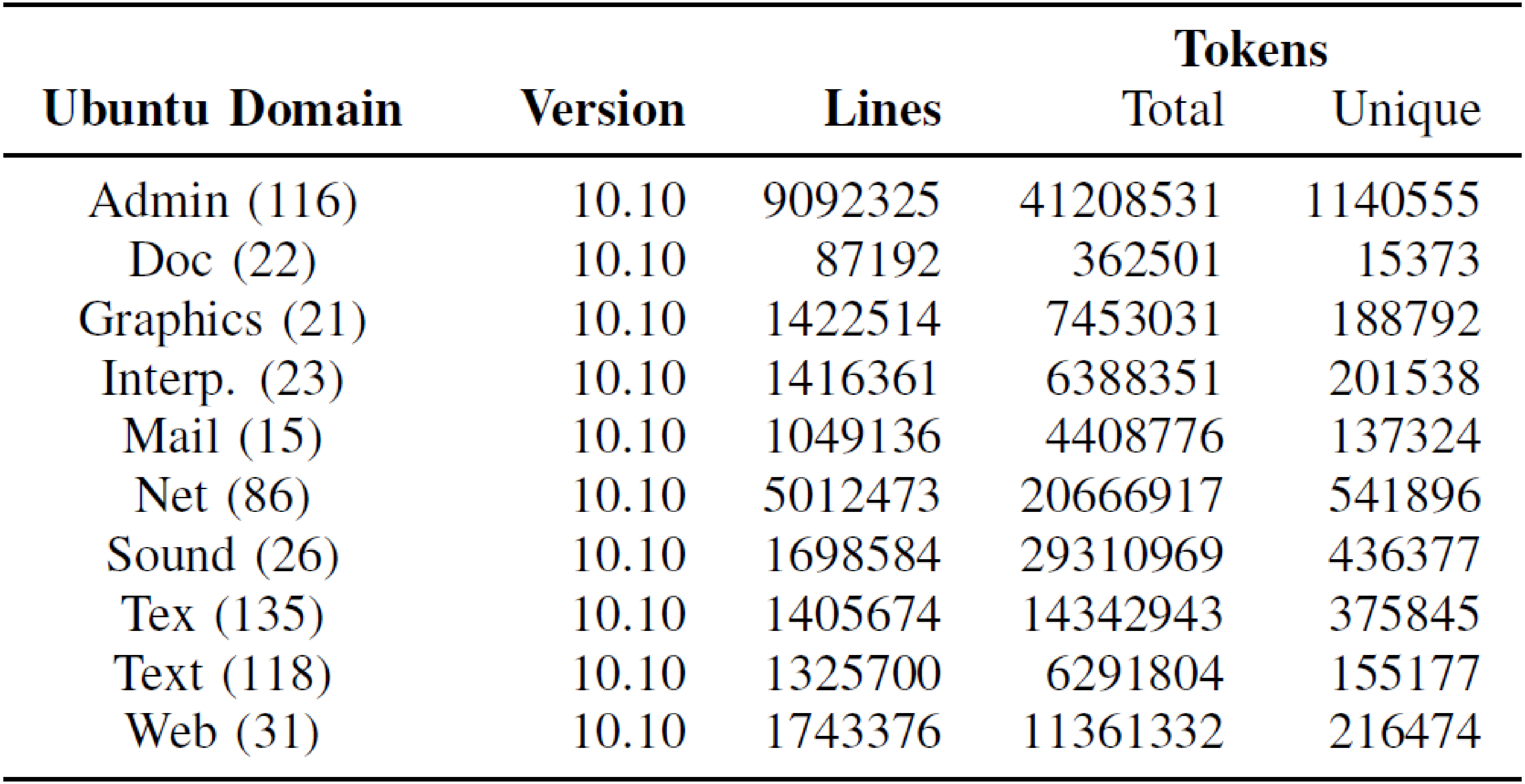
$$p(a_4|a_1a_2a_3) = \frac{count(a_1a_2a_3a_4)}{count(a_1a_2a_3*)}$$

Naturalness of Software: Dataset



			Tokens	
Java Project	**Version**	**Lines**	Total	Unique
Ant	20110123	254457	919148	27008
Batik	20110118	367293	1384554	30298
Cassandra	20110122	135992	697498	13002
Eclipse-E4	20110426	1543206	6807301	98652
Log4J	20101119	68528	247001	8056
Lucene	20100319	429957	2130349	32676
Maven2	20101118	61622	263831	7637
Maven3	20110122	114527	462397	10839
Xalan-J	20091212	349837	1085022	39383
Xerces	20110111	257572	992623	19542

Naturalness of Software: Dataset



			Tokens	
Ubuntu Domain	**Version**	**Lines**	Total	Unique
Admin (116)	10.10	9092325	41208531	1140555
Doc (22)	10.10	87192	362501	15373
Graphics (21)	10.10	1422514	7453031	188792
Interp. (23)	10.10	1416361	6388351	201538
Mail (15)	10.10	1049136	4408776	137324
Net (86)	10.10	5012473	20666917	541896
Sound (26)	10.10	1698584	29310969	436377
Tex (135)	10.10	1405674	14342943	375845
Text (118)	10.10	1325700	6291804	155177
Web (31)	10.10	1743376	11361332	216474

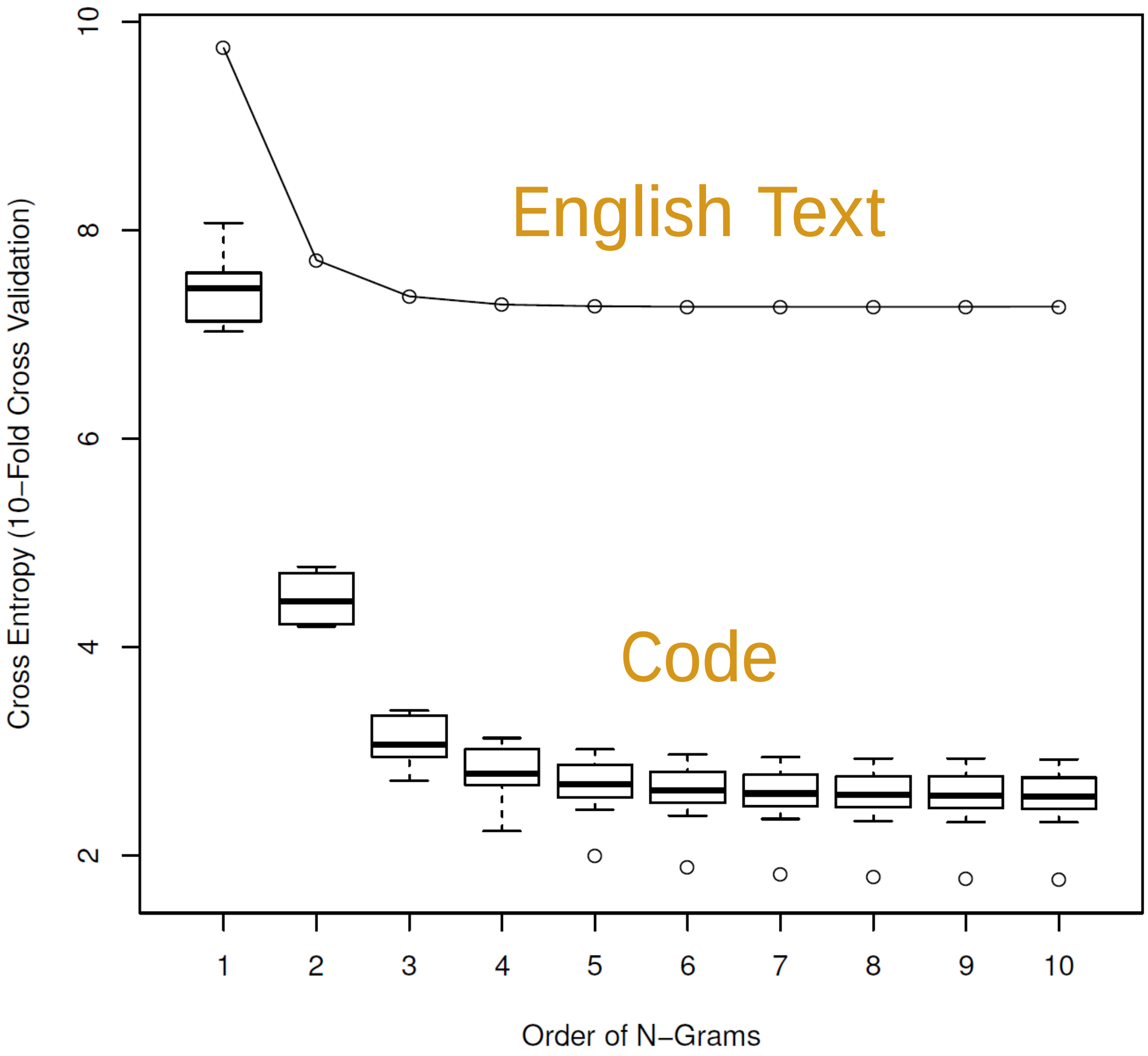
Naturalness of Software: Experiments

- Cross Entropy
  - Captures how bad a language model in modeling a new document.
  - Considering a document s (i.e., $a_1 \ldots a_n$) and a model M, the cross entropy of s wrt. model M:

$$H_{\mathcal{M}}(s) = -\frac{1}{n} \sum_{i=1}^{n} \log \ p_{\mathcal{M}}(a_i \mid a_1 \ldots a_{i-1})$$

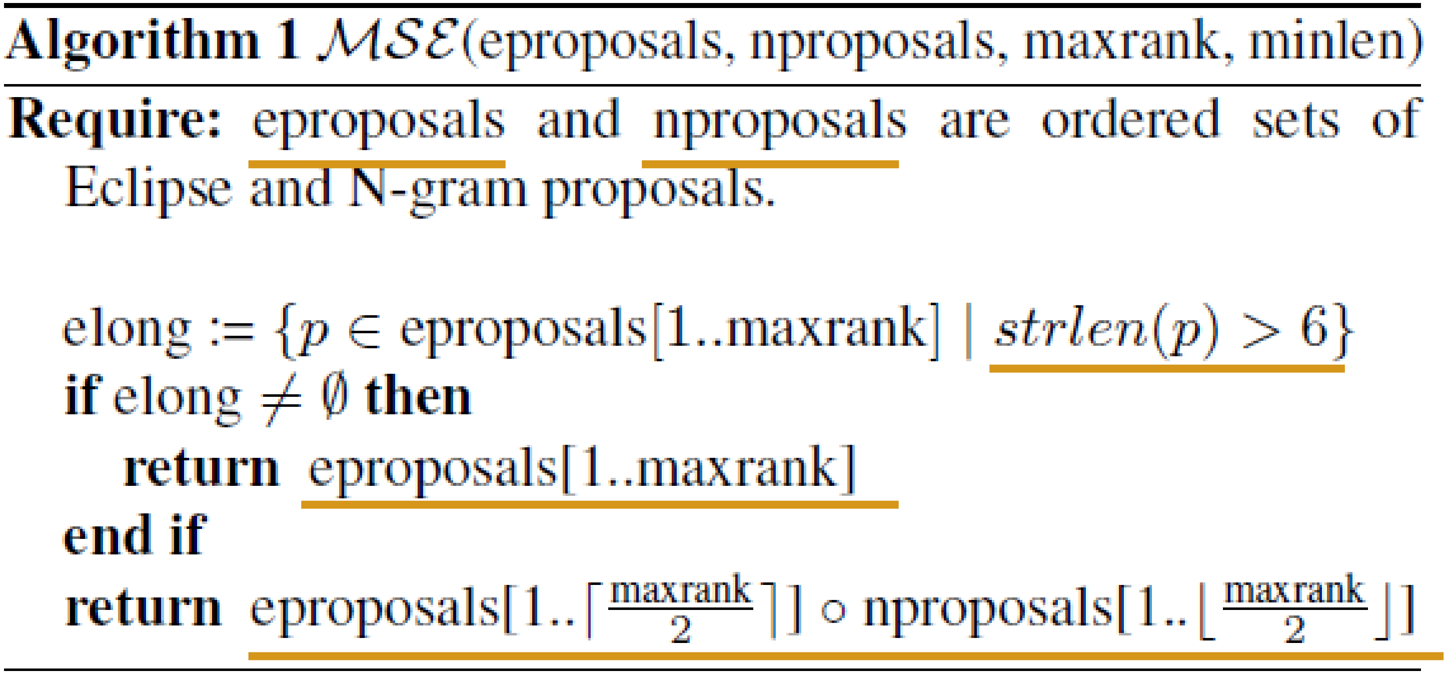
    - $PM(a_i|a_1 \ldots a_{i-1})$ = probability of a_i happening considering model M
  - The lower the cross entropy score, the better a language model is.

Is Software Natural ?



Could it be used for auto-completion?

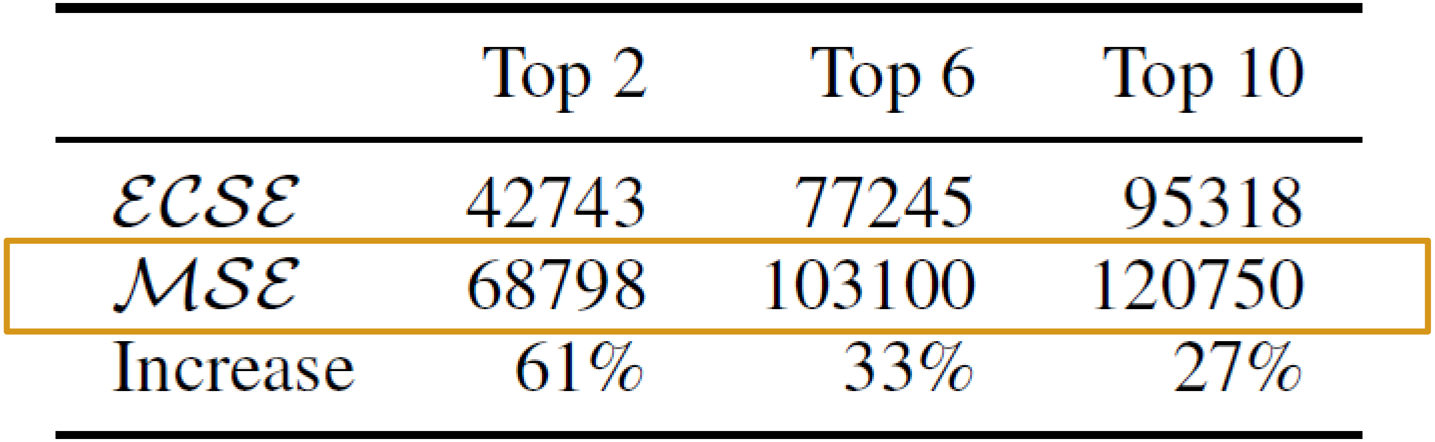
- Extend Eclipse IDE auto-completion function
  - Use Eclipse if at least 1 recommended tokens is long
  - Otherwise use both Eclipse and Language Model
- Uses a trigram model



```
Algorithm 1 $\mathcal{MSE}$(eproposals, nproposals, maxrank, minlen)
Require: eproposals and nproposals are ordered sets of
    Eclipse and N-gram proposals.

    elong := $\{p \in$ eproposals[1..maxrank] $\mid strlen(p) > 6\}$
    if elong $\neq \emptyset$ then
        return eproposals[1..maxrank]
    end if
    return eproposals[1..$\lceil \frac{maxrank}{2} \rceil$] $\circ$ nproposals[1..$\lfloor \frac{maxrank}{2} \rfloor$]
```

Could it be used for auto-completion?

- Use a test set of 200 files to see how good is the auto-complete.
- Keystrokes saved:



	Top 2	Top 6	Top 10
$\mathcal{ECSE}$	42743	77245	95318
$\mathcal{MSE}$	68798	103100	120750
Increase	61%	33%	27%

Other Applications

- Code auto-completion
  - Tung Thanh Nguyen, Anh Tuan Nguyen, Hoan Anh Nguyen, Tien N. Nguyen: A statistical semantic language model for source code. ESEC/SIGSOFT FSE 2013: 532-542

- Finding buggy files from bug descriptions
  - Shivani Rao, Avinash C. Kak: Retrieval from software libraries for bug localization: a comparative study of generic and composite text models. MSR 2011: 43-52

Part IV: Topic Model

- Model a group of words as a topic
  - Typically in a probabilistic sense
- Many recent SE papers use topic models

Structure

- Techniques
  - Topic Modeling: Black-Box View
  - Using Topic Modeling for IR
  - Algorithms
- Applications
  - Bug Localization
  - Other Applications

IV(A): Techniques

Topic Modeling: A Black-Box View
Using Topic Modeling for IR Algorithms

Topic Modeling: Black-Box View

- Model a document as a probability distribution of topics
  - A topic is a probability distribution of words
- Dimensionality reduction: words -> topics
- Benefit: Able to link a document and a query
  - Do not share any words
  - Share related words of the same topics

IR using Topic Model (VSM Like)

- Create topic model for a training set of documents
  - Infer topic distributions of all documents in the training set
- Infer topic distributions of new, unseen document (query)
- Compute similarity between two distributions
  - Kullback Leibner (KL) divergence
  - Jensen Shannon (JS) divergence

IR using Topic Model (Language Model Like)

- We can use the query likelihood model
- Training a topic model computes:

$$P(t \mid topic)$$

$$P(topic \mid d)$$

- With the above we can compute:

$$P(t \mid d) = \sum_{k=1}^{K} P(t \mid topic_k) P(topic_k \mid doc)$$

- Extending to query level, we can compute:

$$P(q \mid d) = \prod_{t \in q} \{\sum_{k=1}^{K} P(t \mid topic_k) P(topic_k \mid d)\}^{tf_{t,d}}$$

Algorithms

- Probabilistic Latent Semantic Analysis (pLSA)
- Latent Dirichlet Allocation (LDA)
- Many more

IV(B): Applications

A Topic-Based Approach for Narrowing the Search Space of Buggy Files from a Bug Report

Bug Localization

- Anh Tuan Nguyen, Tung Thanh Nguyen, Jafar M. Al-Kofahi, Hung Viet Nguyen, Tien N. Nguyen, Iowa State University, USA
- Published in IEEE/ACM International Conference on Automated Software Engineering (ASE), 2011

Bug Localization: Introduction

- Program is often large with hundreds/thousands of files.
- Given a bug report, how to locate files responsible for the bug?
- A (semi) automated solution is needed.

Bug Localization: Technique

- Model the similarity of bug reports and files
  - At topic level
- Model the bug proneness of files
  - Number of bugs in a file (based on its history)
  - Size of the file

Bug Localization: Technique

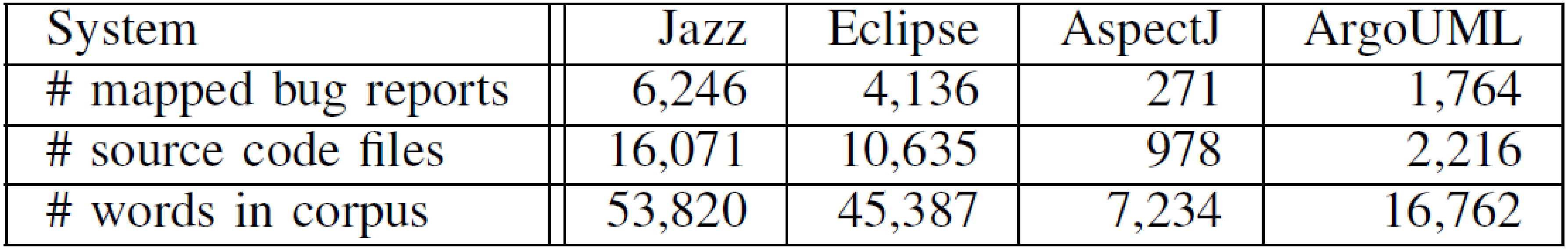
- Computing topic similarity:
  - Learn a topic model
  - Find the topic distribution of a bug report
  - Find the topic distribution of a source code file
  - Compute the similarity using cosine similarity
- Combine topic similarity and bug proneness:

$$P(s|b) = P(s) * sim(s, b)$$

  - P(s) = bug proneness score of file s
  - sim(s,b) = similarity between file s and bug report b

Bug Localization: Experiments

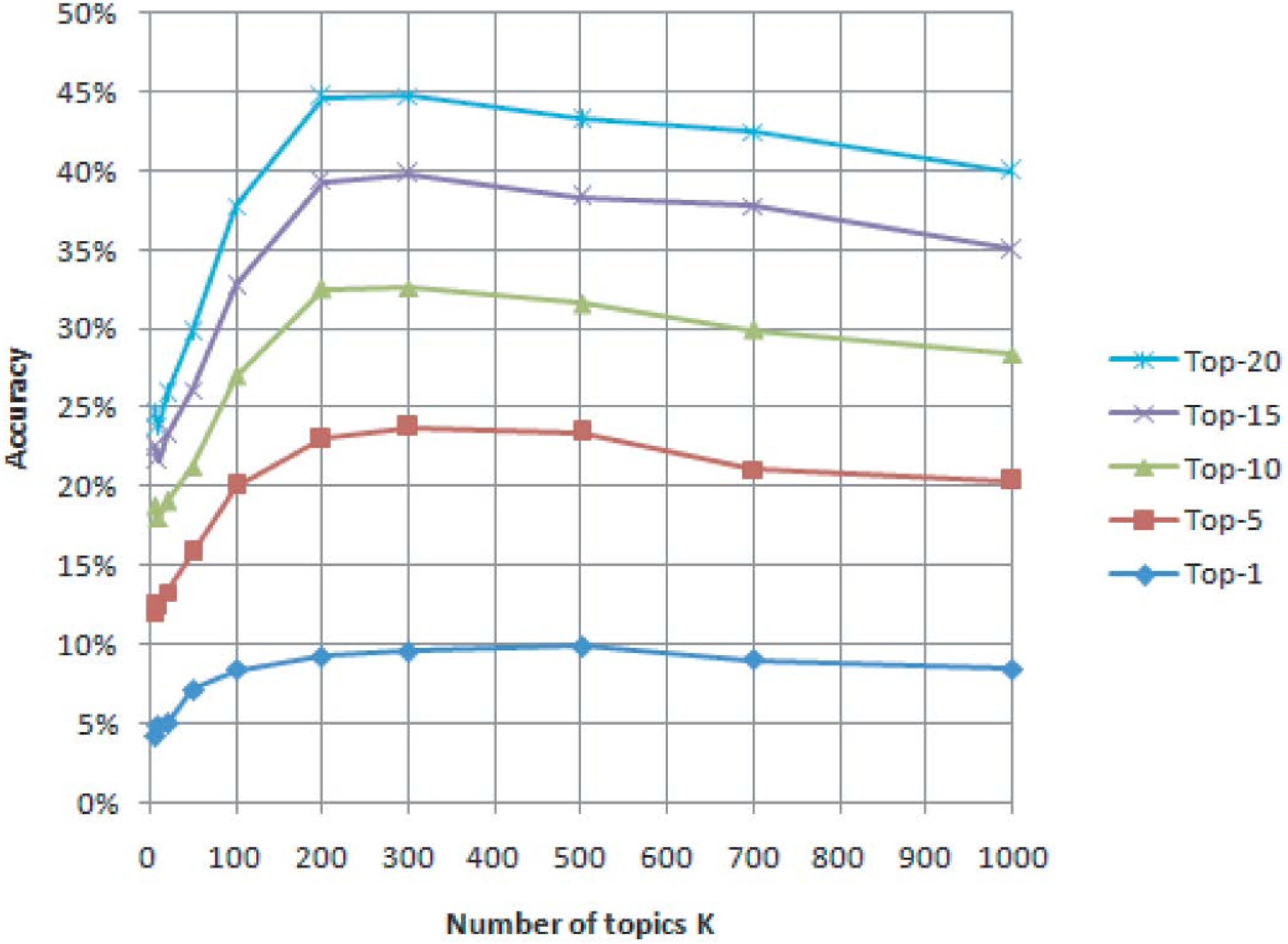
- Subjects



System	Jazz	Eclipse	AspectJ	ArgoUML
# mapped bug reports	6,246	4,136	271	1,764
# source code files	16,071	10,635	978	2,216
# words in corpus	53,820	45,387	7,234	16,762

- Accuracy
  - Return top-k most likely files
  - If at least one matches, then a recommendation is a hit
  - Accuracy = proportion of recommendations which are hits

Bug Localization: Experiments



Other Applications

- Recovering links from code to documentation
  - Andrian Marcus, Jonathan I. Maletic: Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing. ICSE 2003: 125-137

- Black-box test case prioritization
  - Stephen W. Thomas, Hadi Hemmati, Ahmed E. Hassan, Dorothea Blostein: Static test case prioritization using topic models. Empirical Software Engineering 19(1): 182-212 (2014)

Other Applications

- Duplicate bug report detection
    - Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen, David Lo, Chengnian Sun: Duplicate bug report detection with a combination of information retrieval and topic modeling. ASE 2012: 70-79

- Predicting affected components from bug reports
    - Kalyanasundaram Somasundaram, Gail C. Murphy: Automatic categorization of bug reports using latent Dirichlet allocation. ISEC 2012: 125-130

Other Applications

- Recovering links from feature description to source code implementing it
  - Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, Andrea De Lucia: How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. ICSE 2013: 522-531

Part V - Text Classification

- Consider a set of textual documents that are assigned some class labels as a training dataset.
- Create a model that differentiates documents of one class from other class(es).
- Use this model to label textual documents with unknown labels.

Structure

- Techniques
  - Vector space representation
  - Vector space classification
  - Feature selection
- Applications
  - Defect Categorization
  - Other Applications

V(A): Techniques

Vector space representation
Vector space classification
Feature selection

Vector Space Representation

- Each document is a vector
    - One element for each term/word
    - Value of each element:
        - Number of times that word appear
- Normalize each vector (document) to unit length
- High dimensionality: 100,000s of dimensions
    - Terms/words are dimensions

Vector Space Classification

- The training set of documents with known class labels.
  - Labeled set of points in a high dimensional space
- We define lines, surfaces, hypersurfaces to divide regions.
- Use classification algorithms to divide the training sets into regions
  - E.g., SVM

Feature Selection

- Many dimensions correspond to rare words.
    - Rare words can mislead the classifier.
    - Rare misleading features are called noise features.
- Eliminating noise features from the representation
    - Increases efficiency and effectiveness
    - Called feature selection.

Example of a Noise Feature

- A rare term ARACHNOCENTRIC happens to occur in China documents in our training data.
  - Then we may learn a classifier that incorrectly interprets ARACHNOCENTRIC as evidence for the class China.
- Such an incorrect generalization from an accidental property of the training set is called overfitting.
- Feature selection reduces overfitting and improves the accuracy of the classifier.

V(B): Applications

AutoODC: Automated Generation of Orthogonal Defect Classifications

Defect Categorization

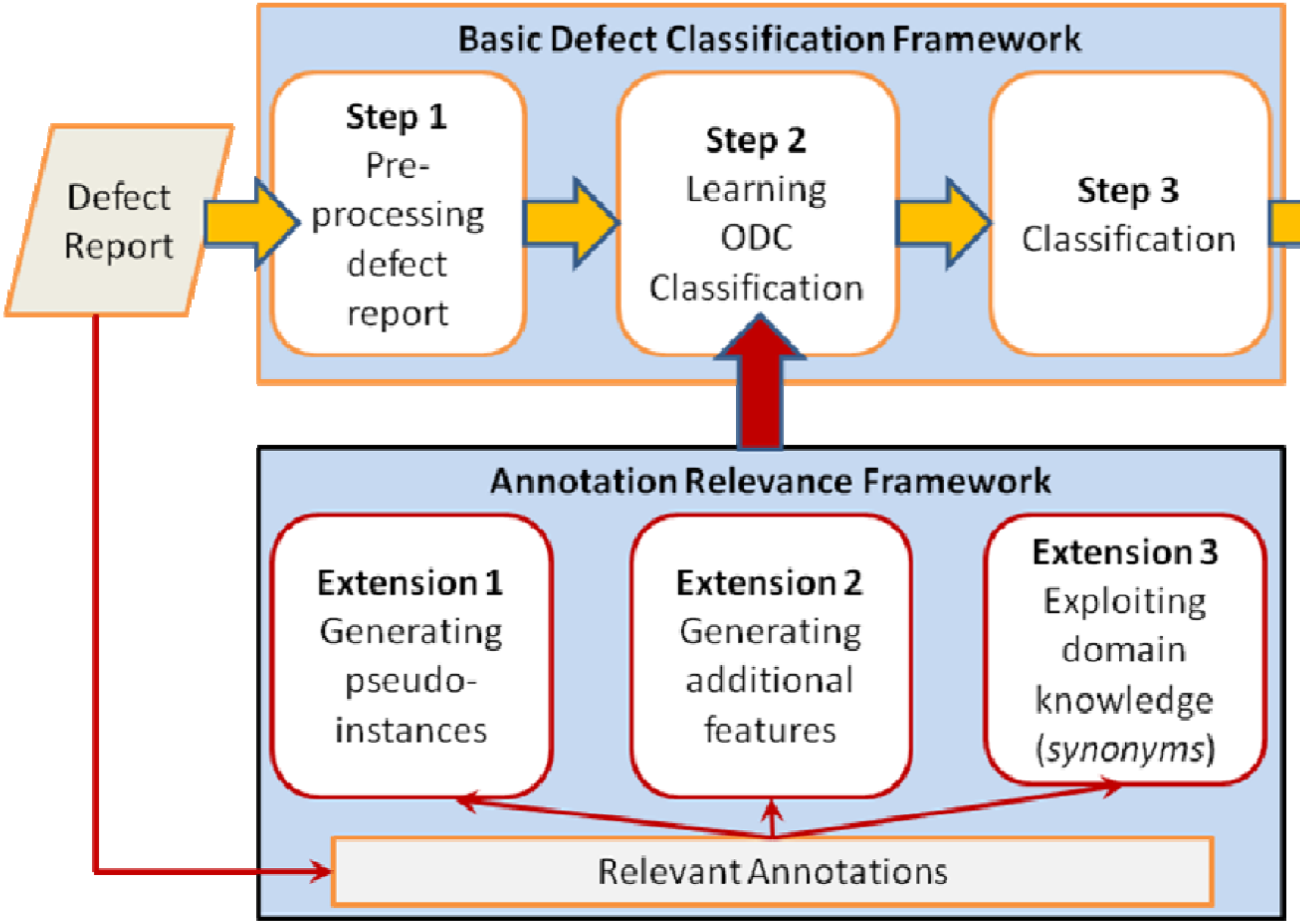
- LiGuo Huang, Ruili Geng, Xu Bai, Jeff Tian, Southern Methodist University, USA
- Vincent Ng, Isaac Persing, University of Texas at Dallas, USA

- Published in IEEE/ACM International Conference on Automated Software Engineering (ASE), 2011

AutoODC: Introduction

- Developers often analyze and categorize bugs for post-mortem investigation
- This process is often done manually
- One commonly used categorization is Orthogonal Defect Categorization (ODC)
    - Class Labels: Reliability, Capability, Security, Usability, Requirements.
- Huang et al. would like to automate the process.

AutoODC: Approach



AutoODC: Preprocessing

- Tokenization
- Stemming
- No removal of stop words
- Normalize each vector

AutoODC: Learning

- Use Support Vector Machine (SVM)
    - Train one SVM per class
    - One-versus-others training
    - Assign class of highest probability value
- Incorporation of user annotations
    - User highlights part of the defect report that are useful for classification
        - Used to generate more instances (pseudo +ve/-ve)
        - Used as “k-gram” like features (new features)
    - Use manually constructed dictionary that define synonymous phrases that are mapped to a common representation (new features)

AutoODC: Results

	Reliability	**Capability**	**Security**	**Usability**	**Require-ments**
F-Measure	22.2%	88.5%	70.0%	62.9%	39.3%

Other Applications

- Predicting severity of bug reports
  - Tim Menzies, Andrian Marcus: Automated severity assessment of software defect reports. ICSM 2008: 346-355

- Predicting priority of bug reports
  - Yuan Tian, David Lo, Chengnian Sun: DRONE: Predicting Priority of Reported Bugs by Multi-factor Analysis. ICSM 2013: 200-209

Other Applications

- Content categorization in software forums
  - Daqing Hou, Lingfeng Mo: Content Categorization of API Discussions. ICSM 2013: 60-69

- Filtering software microblogs
  - Philips Kokoh Prasetyo, David Lo, Palakorn Achananuparp, Yuan Tian, Ee-Peng Lim: Automatic classification of software related microblogs. ICSM 2012: 596-599

Other Applications

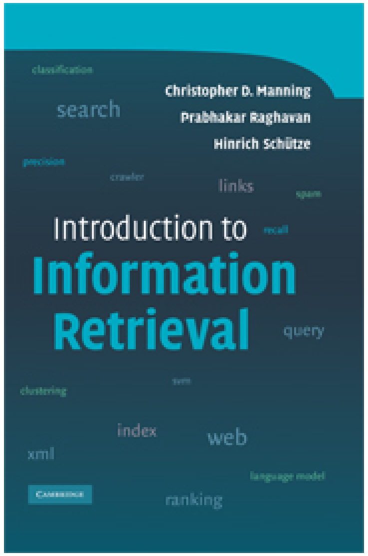
- Recommending a developer to fix a bug report
    - John Anvik, Gail C. Murphy: Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM Trans. Softw. Eng. Methodol. 20(3): 10 (2011)

Conclusion

- Part I: Preliminaries
  - Tokeniz., Stop Word Removal, Stemming, Indexing, etc.
- Part II: Vector Space Modeling
  - Model a document as a vector of term weights
- Part III: Language Model
  - Model a document as a probability distribution of terms
  - Query likelihood model
- Part IV: Topic Model
  - Model a document as a probability distribution of topics
  - Model a topic as a probability distribution of words
- Part V: Text Classification
  - Convert to VSM representation
  - Use standard classifiers (e.g., SVM)

Acknowledgements & Additional References

- Many slides and images are taken or adapted from:
  - Resource slides of: Introduction to Information Retrieval, by Manning et al., Cambridge Press, 2008
  - The research papers mentioned in the slides.



Thank you!

Questions? Comments?

davidlo@smu.edu.sg