

Software Engineering Researches in Osaka University

Katsuro Inoue



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Talk Structure

- Overview of Osaka University and Software Engineering Lab
- Code Clone Analysis and Application
- Component Ranking based on Use Relation
- Empirical Approach to Software Engineering



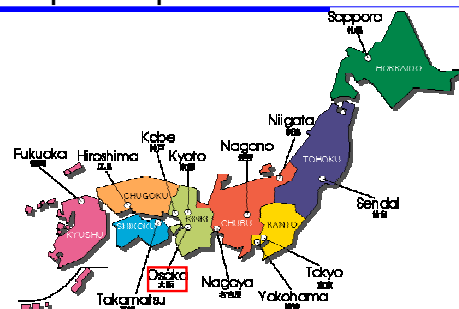
Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Overview of Osaka University and Software Engineering Lab



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Map of Japan



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Osaka University



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Osaka University

- 75 years history
- 14 schools
- 1000 million \$ budget
- 2500 faculty members
- 13,000 undergraduate students
- 7,500 graduate students
- Computer related school
 - Graduate School of Information Science and Technology



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Graduate Courses

Graduate School of Information Science and Technology

- *Pure and Applied Mathematics*
 - *Information and Physical Sciences*
 - *Computer Science*
 - *Information Systems Engineering*
 - *Information Networking*
 - *Multimedia Engineering*
 - *Bioinformatic Engineering*
- about 250 master students, 100 PhD students



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Undergraduate Courses

- School of Engineering Science
 - Department of Informatics and Mathematical Scienceabout 350 students
- School of Engineering
 - Department of Information Systems Engineeringabout 250 students



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Software Engineering Related Lab's

- 3 research groups, 8 faculty members
 - Software design and verification in formal approach
 - Dependable computing and reliability engineering
 - *Software engineering in empirical approach (SE lab.)*
- 2 faculty members in SE lab.
 - Inoue, K. Software Reuse, Program Analysis
 - Matsushita, M. Open Source, Software Environment
- Students
 - 1 post doc
 - 9 Ph.D. candidates (1 foreign student, Italy)
 - 13 master students (2 foreign students, Malaysia)
 - 4 undergraduate students



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Our Research Focus (1)

- Software analysis
 - Large scale and practical targets
 - *Code-clone detection*
 - *Component Ranking for software reuse*
 - Alias analysis for object oriented language



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Our Research Focus (2)

- Empirical Software Engineering
 - Large scale measurement
 - *Real-time data collection and analysis*
 - Function point analysis
 - Object oriented program measurement



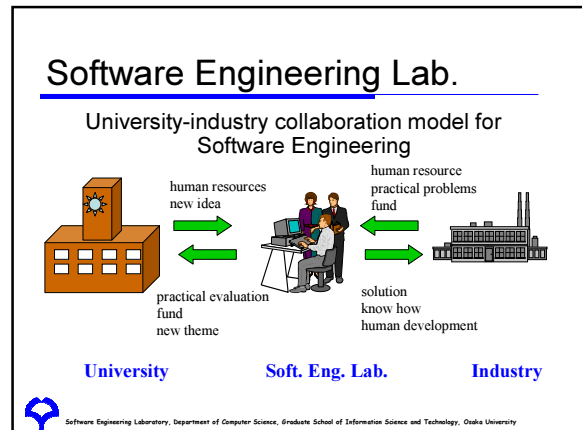
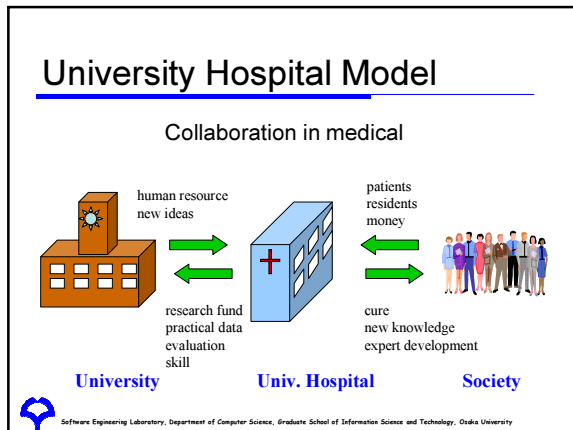
Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Our Research Focus (3)

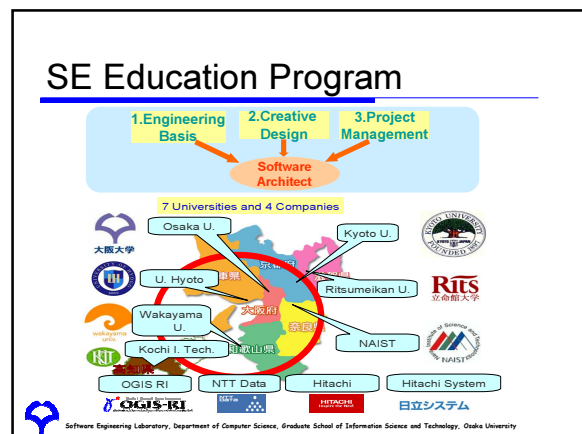
- Software Development Environment
 - Practically used tools
 - Open source
 - Versioning system



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University



- ### Collaboration and Funding
- EASE (Empirical Approach to Software Engineering)
 - MEXT (Ministry of Education, Culture, Sports, Science and Technology)
 - Code-Clone Analysis
 - MEXT
 - Software Engineering Center, Japan
 - Mega Software Engineering
 - MEXT
 - Software Maintenance Effort Estimation
 - Fujitsu Lab.
 - Model-Driven Architecture
 - NTT Data
 - XML-Based Domain Modeling
 - Hitachi
- Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University




- ### Talk Structure
- Overview of Osaka University and Software Engineering Lab
 - Code Clone Analysis and Application
 - Component Ranking based on Use Relation
 - Empirical Approach to Software Engineering
- Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Code Clone Analysis and Application

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

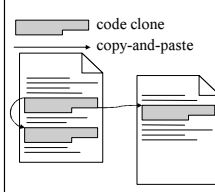

Clone Detection



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

What is Code Clone?


- A code fragment which has identical or similar code fragments in source code
- Introduced in source code because of various reasons
 - code reuse by 'copy-and-paste'
 - stereotyped function
 - ex. file open, DB connect, ...
 - intentional iteration
 - performance enhancement
- Makes software maintenance more difficult
 - If we modify a code clone with many similar code fragments, it is necessary to consider whether or not we have to modify each of them
 - It is likely to overlook some of them

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University


Definition of Code Clone

- No single or generic definition of code clone
 - So far, several methods of code clone detection have been proposed, and each of them has its own definition about code clone
- Various detection methods
 1. Line-based comparison
 2. AST (Abstract Syntax Tree) based comparison
 3. PDG(Program Dependency Graph) based comparison
 4. Metrics comparison
 5. Token-based comparison



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

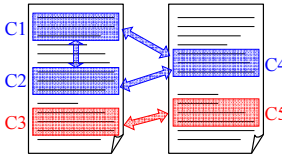
CCFinder and Associate Tools




Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Clone Pair and Clone Set

- Clone Pair
 - a pair of identical or similar code fragments
- Clone Set
 - a set of identical or similar fragments




Clone Pair	Clone Set
(C1, C2)	{C1, C2, C4}
(C1, C4)	{C3, C5}
(C2, C4)	
(C3, C5)	



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Our Code Clone Research

- Develop tools
 - Detection tool: CCFinder
 - Visualization tool: Gemini
 - Refactoring support tool: Aries
 - Debug support tool: Libra
- Deliver our tools to domestic or overseas organizations/individuals
 - More than 100 companies uses our tools!
- Promote academic-industrial collaboration
 - Organize code clone seminars
 - Manage mailing-lists




Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Detection tool:

Development of CCFinder

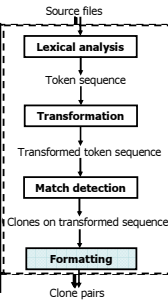
- Developed by industry requirement
 - Maintenance of a huge system
 - More than 10M LOC, more than 20 years old
 - Maintenance of code clones by hand had been performed, but ...
- Token-base clone detection tool CCFinder
 - Normalization of name space
 - Parameterization of user-defined names
 - Removal of table initialization
 - Identification of module delimiter
 - Suffix-tree algorithm
- CCFinder can analyze the system of millions line scale in 5-30 min.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University


Detection tool:

CCFinder Detection Process



```

1. static void foo() throws RESyntaxException {
2.   String a[] = new String [] { "123,400", "abc", "orange 100" };
3.   org.apache.regexp.RE pat = new org.apache.regexp.RE("(0-9,]+");
4.   int sum = 0;
5.   for (int i = 0; i < a.length; ++i)
6.     if (pat.match(a[i]))
7.       sum += Sample.parseNumber(pat.getParent());
8.   System.out.println("sum = " + sum);
9. }
10. static void goo(String [] a) throws RESyntaxException {
11.   RE exp = new RE("(0-9,]+");
12.   int sum = 0;
13.   for (int i = 0; i < a.length; ++i)
14.     if (exp.match(a[i]))
15.       sum += parseNumber(exp.getParent());
16.   System.out.println("sum = " + sum);
17. }
    
```

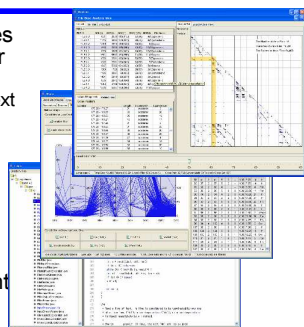



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Visualization Tool:

Gemini Outline

- Visualizes code clones detected by CCFinder
 - CCFinder outputs the detection result to a text file
- Provides interactive analyses of code clones
 - Scatter Plot
 - Clone metrics
 - File metrics
- Filters out unimportant code clones

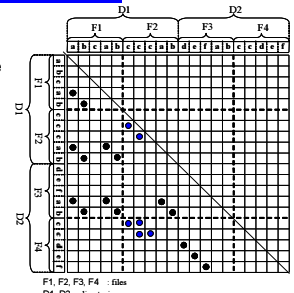




Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Visualization tool:

Gemini Scatter Plot

- Visually shows where code clones are
- Both the vertical and horizontal axes represent the token sequence of source code
 - The original point is the upper left corner
- means that corresponding two tokens on the two axes are the same
 - Symmetric to diagonal (show only low below)
 - is an element of practical code clone
 - is an element of non-interesting code clone





Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Visualization tool:


Gemini Clone Metrics, File Metrics

- Metrics are used to quantitatively characterize entities
- Clone metrics
 - LEN(S)**: the average length of code fragments (the number of tokens) in clone set **S**
 - POP(S)**: the number of code fragments in **S**
 - NIF(S)**: the number of source files including any fragments of **S**
 - RNR(S)**: the ratio of non-repeated code sequence in **S**
- File metrics
 - ROC(F)**: the ratio of duplication of file **F**
 - if completely duplicated, the value is 1.0
 - if not duplicated at all, the value is 0.0
 - NOC(F)**: the number of code fragments of any clone set in file **F**
 - NOF(F)**: the number of files sharing any code clones with file **F**



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University


Applications



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Academic-industrial collaboration:
Code Clone Seminar

- We have organized code seminars 6 times
 - From Dec 2002
- Seminar is the place where we exchange views with industrial people
- Contents of Seminar
 - Tools demonstration
 - Lecture of how to use code clone information
 - Case report of companies using our tools



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

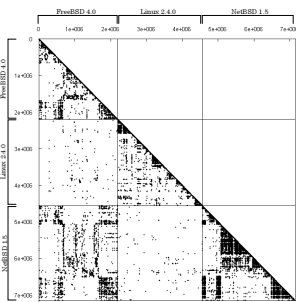
Case Studies

- Open source software
 - FreeBSD, NetBSD, Linux(C, 7MLLOC)
 - JDK Libraries(Java 1.8MLLOC)
 - Qt(C++, 240KLLOC)
- Commercial software (more than 100 companies)
 - IPA/SEC, NTT Data Corp., Hitachi Ltd., Hitachi GP, Hitachi SAS, NEC soft Ltd., ASTEC Inc., SRA Inc., JAXA, Daiwa Computer, etc...
- Students excise of Osaka University
- Filed in a court as an evidence for software copyright suit

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Case study 1:
Similarity between FreeBSD, NetBSD, Linux

- Result
 - There are many code clones between FreeBSD and NetBSD
 - There are little code clones between Linux and FreeBSD/NetBSD
- Their Histories can explain the result
 - The ancestors of FreeBSD and NetBSD are the same
 - Linux was made from scratch



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

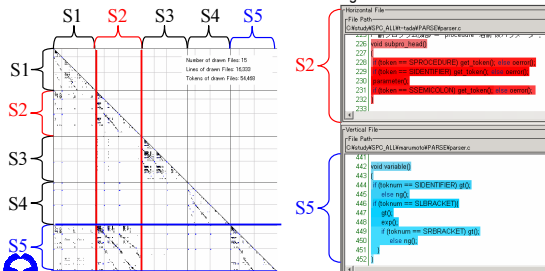
Case study 2:
Students Excise

- Target
 - Programs developed in a programming exercise of Osaka Univ.
 - Simple compiler for Pascal written in C language
 - Programs of 5 students
 - This exercise consists of 3 steps
 - STEP1: makes a **syntax checker**
 - STEP2: makes a **semantics checker** by extending his/her syntax checker
 - STEP3: makes a **total compiler** by extending his/her semantic checker
- Purpose
 - Similarity among students
 - In programming exercise, plagiarisms sometimes happen

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Result

- There were a lot of code clones between S2 and S5
- We did not use the detection result for evaluating their excises



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Case study 3:
IPA/SEC Advanced Project

- Target
 - A probe information system developed by 5 Japanese companies
 - The project manager didn't know states of the source code because the companies individually developed the components
- Purpose
 - Grasp features of black-boxed source code
- Others
 - Applied twice times, after unit test (280,000LOC), after combined test(300,000LOC)
 - The minimum size of detected code clone is 30 tokens

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

IPA/SEC Advanced Project:

Duplicated Ratio

- The below graph illustrates the transition of duplicated ratio of the sub-system developed by a company

Duplicated ratio	After unit test	After combined test
0% - 10%	10	10
11% - 20%	15	15
21% - 30%	20	20
31% - 40%	25	25
41% - 50%	30	30
51% - 60%	35	35
61% - 70%	40	40
71% - 80%	45	45
81% - 90%	50	50
91%+ 100%	55	55

- We interviewed developers of the sub-system
 - They added library code managed in their company to the system to add new functions right before combined test
 - Reliability of the library code is high because it has been maintained for a long time

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

IPA/SEC Advanced Project:

Scatter Plot Analysis

- Scatter Plot of company X
- In part A, there are many non-interesting code clones
 - output code for debug (consecutive printf-statements)
 - check data validity
 - consecutive if-statements
- In part B, there are many code clones across directories
 - This part treats vehicle position information
 - Each directory include a single kind of vehicles, e.g., taxi, bus, or truck
 - Logics are mostly the same

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Summary of Code Clone Analysis and Application

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Conclusion

- We have developed Code clone analysis tools
 - Detection tool: CCFinder
 - Visualization tool: Gemini
 - Refactoring support tool: Aries
 - Debug support tool: Libra
- We have promoted academic-industrial collaboration
 - organize code clone seminars
 - manage mailing lists
- We have applied our tools to various software

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Future Direction

- CCFinderX by Kamiya at AIST
 - Token analyzer is definable
- Architecture evolution by the view of code clones
- System analysis via code clones associated with other metrics

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Resources

- Papers
 - T. Kamiya, S. Kusumoto, and K. Inoue, CCFinder: A multi-linguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 654-670, Jul. 2002.
 - Many Others ... See our home page
- Web
 - CCFinder: <http://sel.ist.osaka-u.ac.jp/cdtools/index-e.html>
 - CCFinderX: <http://www.ccfinder.net/ccfinderx.html>
- Tools
 - See home pages

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Talk Structure

- Overview of Osaka University and Software Engineering Lab
- Code Clone Analysis and Application
- **Component Ranking based on Use Relation**
- Empirical Approach to Software Engineering



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Component Rank Based on Use Relation



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

SourceForge

- Large open source software development web site
- Version control, communication support, ...

Hosted Projects: **121,208**
Registered Users: **1,322,774**



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Motivation

- Numerous software systems are being developed day by day
- Similar components (libraries, portions of codes, or abstracted algorithms, ...) might be independently developed in different projects
- Reuse:
 - Key factor for high productivity and reliability in today's software development
- Large software libraries:
 - Little support to search components effectively
 - Managing structure and consistency is very difficult and impractical



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

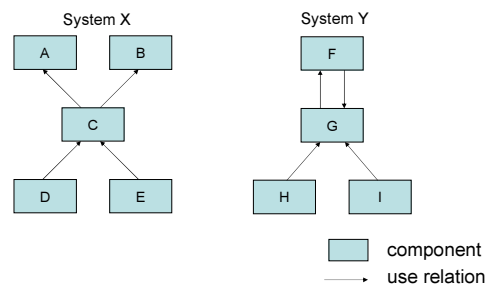
Automated Component Library

- Collect software components eagerly without preserving their inherent structures
- Analyze relations among components by using various analysis techniques
- Rank the components based on their significance [Component Rank Model](#)
- Answer user's queries according to the rank



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Component Graph



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Weight of Nodes

$0 \leq w(x) \leq 1$
 sum of all node weights = 1 ... (1)
 weight of node represents significance of node

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Weights of Edges

$w(A) = \text{sum of all outgoing edge weights} \dots (2)$
 $\text{sum of all incoming edge weights} = w(B) \dots (3)$

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Definition of Weights

- Under constraints (1)~(3), we have a simultaneous equation

$$\begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix} = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{pmatrix}^t \cdot \begin{pmatrix} w(v_1) \\ w(v_2) \\ \vdots \\ w(v_n) \end{pmatrix}$$

W : node weight vector D : transposed matrix of distribution ratio

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Propagating Weights

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Propagating Weights

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Propagating Weights

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Propagating Weights

Stable weight assignment (eigenvector computation)
Component Rank : order of nodes sorted by the weight

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Markov Model

- Component rank model can be considered as a Markov Chain of user's focus
- User's focus moves from one component to another along a use relation at a fixed time period
- Node weight represents the existence probability of the user's focus

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Clustering Components

$A \cong D, B \cong F$
component graph

clustered component graph

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

SPARS-J: Component Rank System

input: .java file = component

similarity measure by SMMT

use relation extraction

similarity criterion f : sharing 80% statements

clustering

clustered graph construction

weight ratio p between real and pseudo edges : 0.85

node weight computation

de-clustering to original graph

output: component-rank pairs

inheritance
method call
attribute access
abstract class impl

equal distribution ratio d' to outgoing edges

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Experiment 1

JDK1.3.0

575,000 lines, 1877 components
7 minutes on PC (Pentium IV, 2GHz, 2GB)

rank	class name	weight
1	java.lang.Object	0.16126
2	java.lang.Class	0.08712
3	java.lang.Throwable	0.05510
4	java.lang.Exception	0.03103
5	java.io.IOException	0.01343
6	java.lang.StringBuffer	0.01214
7	java.lang.SecurityManager	0.01169
8	java.io.InputStream	0.01027
9	java.lang.reflect.Field	0.00948
10	java.lang.reflect.Constructor	0.00936
...
1256	sunw.util.EventListener	0.00011
...
1256

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Experiment 2:

Collection of SE Tools and Libraries

- CK metrics measurement tools, component rank system
- ANTLR, JAMA, Caffe Cappuccino
- 582 components

rank	class name	weight
1	antlr.Token	0.10727
2	antlr.debug.Event	0.06189
2	antlr.debug.NewLineEvent	0.06189
4	antlr.collections.impl.Vector	0.05434
5	jp.gr.java_conf.keisuken.text.html.HtmlParameter	0.05246
6	jp.gr.java_conf.keisuken.net.server.ServerProperties	0.03699
7	Jama.Matrix	0.01564
8	jp.gr.java_conf.keisuken.util.IntegerArray	0.01390
8	jp.gr.java_conf.keisuken.util.LongArray	0.01390
10	jp.ac.osaka_u.es.ics.iip_lab.metrics.parser.IdentifierInfo	0.01365
...
418	cktool_new.examples.Main	0.00050

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Experiment 3:

Application to Industry

- Daiwa computer: a middle size software company in Osaka
- A shared Java application framework for web-based data management
- 5 applications + framework
 - 1538 components, 339 clustered nodes
- Classes in the framework and definitions of data structure are highly ranked



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Related Works

- Markov models of documentation traversal
 - *Influence Weight*: impact of publication thought references
 - *Page Rank*: weight of HTML in the Internet through web links
 Explicit use relation
 No clustering (important for software products)
- Reusability measurement
 - Various characteristic metrics of components or interfaces
 Indirect inference of reusability (our approach directly reflects usage of components)



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

The screenshot shows the SPARS-J search interface. At the top, the logo 'SPARS J' is displayed in a stylized blue font. Below the logo is a search bar with the keyword 'bubblesort' entered and a 'Search' button. Underneath the search bar are links for 'Preferences' and 'Advanced Search'. At the bottom of the page, it says 'Searching 171,160 classes.' and '© 2003, The SPARS Project & Osaka University.'

The screenshot shows search results for 'bubblesort'. It indicates '17 groups (26 classes) found'. Two results are visible:

- Group 1: **BubbleSort**. Description: java.about.com. Last modified: Sat Feb 9 23:14:52 2002. File name: BubbleSort.java (LOC: 22, # of Methods: 1). Used by 2 classes.
- Group 2: **SortItem**. Description: www.tolstoy.com. Last modified: Mon Feb 3 11:32:07 2003. File name: SortItem.java (LOC: 202, # of Methods: 11).

The screenshot shows the 'Class View' for 'BubbleSort'. It includes a navigation bar with links for 'Source Code', 'Group', 'Using BubbleSort', 'Used by BubbleSort', 'Metrics', and 'Download'. The source code is displayed as follows:

```

/*
 * Copyright (c) 1999-2002, Xiaoping Jia.
 * All Rights Reserved.
 */

/**
 * This program implements the bubble sort algorithm to sort an array of integers.
 */
public class BubbleSort {

    public static void main(String[] args) {
        int a[] = {21, 9, 45, 17, 33, 72, 50, 12, 41, 39};

        for (int i = a.length - 1; i >= 0; ) {
            for (int j = 0; j < i; j++) {
                if (a[j] > a[j+1]) {
                    int temp = a[j];
                    a[j] = a[j+1];
                    a[j+1] = temp;
                }
            }
        }
    }
}
    
```

Conclusion & Future Work

- Component Rank: a novel model for software component
- SPARS-J for Java
- Application to various collections of Java programs
- Application to Companies
- Other model (weight distribution, similarity, ...)



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Resources

- Papers
 - Component Rank: Relative Significance Rank for Software Component Search
Proceedings of the 25th International Conference on Software Engineering (ICSE2003), pp14-24, Portland, Oregon, U.S.A., May 6-8, 2003.
 - Ranking Significance of Software Components Based on Use Relations
IEEE Transactions on Software Engineering, Vol.31, No.3, pp.213-225
- SPARS-J Demo WEB site (about 200,000 Java classes)
<http://www.spars.info>



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Talk Structure

- Overview of Osaka University and Software Engineering Lab
- Code Clone Analysis and Application
- Component Ranking based on Use Relation
- Empirical Approach to Software Engineering



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Empirical Approach to Software Engineering



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Software Development with Scientific Background

- Many other science and technology fields use an approach:
 - Measure -> Quantify
 - Evaluation based on the quantification
 - Feedback for improvement
- How about software engineering field?



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Current Situation on Software Engineering

- Many many software engineering methods, tools and techniques had been proposed for 30 years.
- Are they really useful?
- No real evaluation for them
 - Evaluation is generally very expensive
 - Evaluation by history (ICSE n-10)



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Empirical Software Engineering

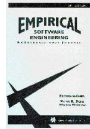
- Quantitative evaluation of various methods, tools, and techniques on software engineering
- Limitation of university
- Data collection from industry is essential
 - Collaboration with industry



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

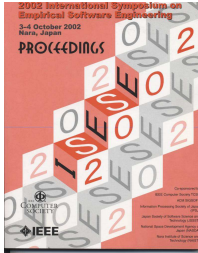
Journal by Kluwer
Empirical Software Engineering

- **Scope**
 - Cost estimation techniques
 - Analysis of the effects of design methods and characteristics
 - Evaluation of testing methodologies
 - Development of predictive models of defect rates and reliability from real data
 - Infrastructure issues, such as measurement theory, experimental design, qualitative modeling and analysis approaches.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

International Symposium on Empirical Software Engineering
ISESE



- ISESE2002 @ Nara, Japan
- ISESE2003 @ Roma, Italy
- ISESE2004 @ California, USA
- ISESE2005 @ Noosa Heads, AU
- ISESE2006 @ Rio de Janeiro, Brazil

Sep 21-22, 2006

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

International Software Engineering Research Network
ISERN

- ISERN is a community that believes software engineering research needs to be performed in an experimental context.
- ISERN was established in 1993 by researchers of software engineering from 12 countries, including USA, Germany, Australia, Italy, Finland and Japan.
- ISERN provides several means of communication between members;
 - Electronic Communication,
 - Annual meetings, and
 - Exchange of researchers.

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

The EASE Project

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

What Is the EASE Project?

- **Empirical Approach to Software Engineering**
- One of the leading projects of the Ministry of Education, Culture, Sports, Science and Technology (MEXT).
- 5 year project starting in 2003.
- Budget: about \$1 million US / year.
- Project leader: Koji Torii, NAIST
 - Sub-leader: Katsuro Inoue, Osaka University
 - Kenichi Matsumoto, NAIST

<http://www.empirical.jp/English/>



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

The Purpose of the EASE Project

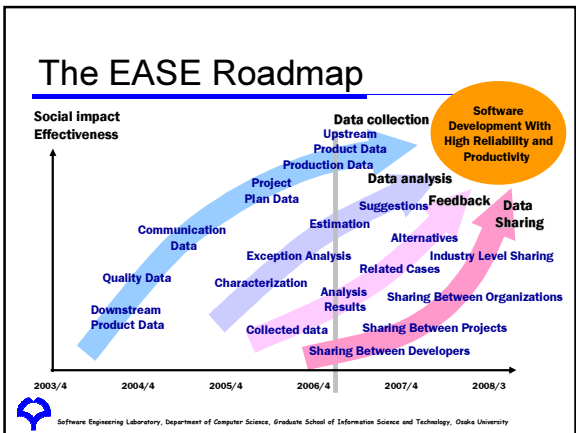
- Achievement of software development technology based on quantitative data
 - Construction of a quantitative data collection system
 - **Result 1: Making of EPM open source**
 - Construction of a system that supports development based on analyzed data
 - **Result 2: EPM application experience**
 - **Result 3: Coordinated cooperation with SEC**
- Spread and promotion of software development technology based on quantitative data to industry sites
 - **Result 4: Activation of the industrial world**

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Empirical Activities in EASE

- Data collection in real time, e.g.
 - configuration management history
 - issue tracking history
 - e-mail communication history
- Analysis with software tools, e.g.
 - metrics measurement
 - project categorization
 - collaborative filtering
 - software component retrieval
- Feedback to stakeholders for improvement, e.g.
 - observations and rules
 - experiences and instances in previous projects

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University



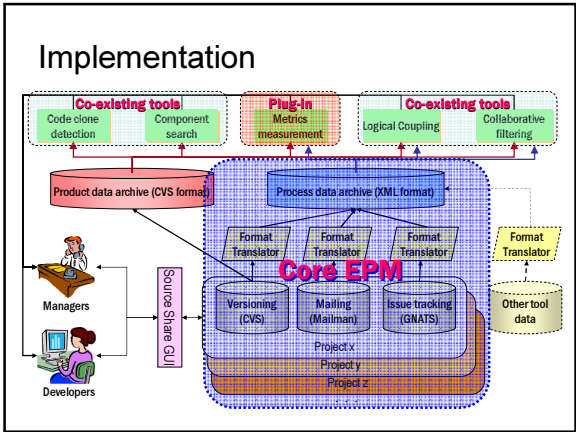
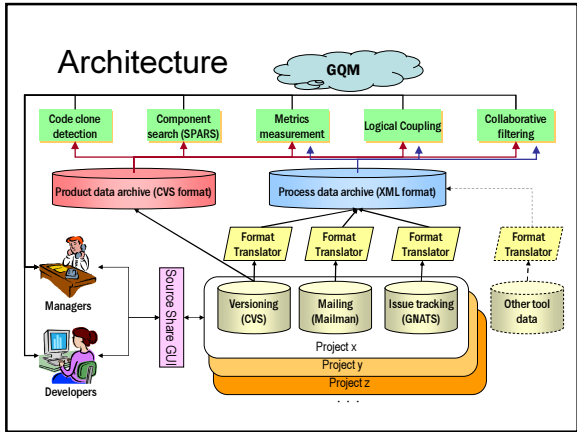
EPM: the Empirical Project Monitor

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

EPM The Empirical Project Monitor

- An application supporting empirical software engineering
- EPM **automatically collects** development data accumulated in development tools through everyday development activities
 - Configuration management system: CVS
 - Issue tracking systems: GNATS
 - Mailing list managers: Mailman, Majordomo, FML

Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University



Automated Data Collection in EPM

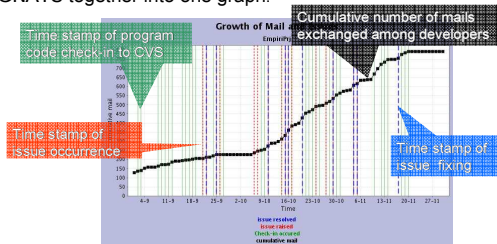
- Reduces the reporting burden on developers
 - without additional work for developers
- Reduces the project information delay
 - data available in real time
- Avoids mistakes and estimation errors
 - uses real (quantitative) data

EPM's GUI



An Example of Output

- EPM can put data collected by CVS, Mailman, and GNATS together into one graph.



Merits to Introducing EPM

- Easy monitoring of projects in cooperation with the existing development environment.
- Easy accumulation of the knowledge and experience of projects.
- Collection and sharing of uniform data for projects in real time.
- Sharing and reuse of information enabled through empirical data.

Data Analysis with EPM

How can we use so much data?



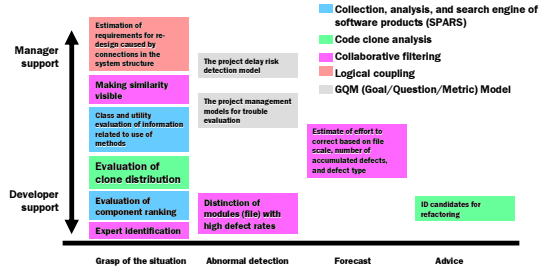
Analysis Technologies and Models

- Code clone analysis (CCfinder)
- Collection, analysis, and search engine of software products (SPARS: Software Product Archive, analysis, and Retrieval System)
- Collaborative filtering
- Logical coupling
- GQM (Goal/Question/Metric) model



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Analysis Targets



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

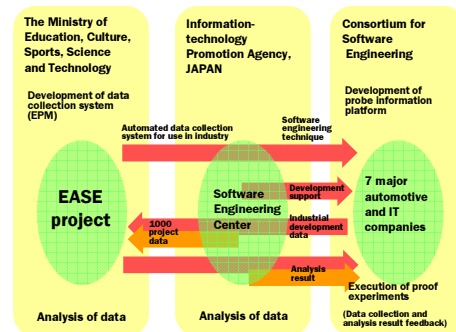
Applying EPM in Industries

- EPM is being applied to several real projects
 - Business systems (Hitachi GP, Ltd.)
 - Personal mobile applications (Mitsubishi Space Software Co., Ltd.)
 - Automobile information systems (SEC collaborative project)...
- Very low additional effort by developers for data collection
- Collected data is currently under analysis
 - Many findings only from collected data such as
 - Module refactoring candidates
 - Internal trouble detection



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Collaboration between EASE and SEC



EPM Application Experience

Collaborating Enterprises	Hitachi GP, Ltd.	Mitsubishi Space Software Co., Ltd.
Software for application	Package software for business in municipality agency	Software purchased by a certain enterprise
Development language	Java	Java, others
Development period	6 months	10 months
Development scale	130,000 Loc	250,000 Loc
EPM introduction and operation cost	25 man-days	11 man-days
EPM subjective evaluation by developers	The automated data collection is useful. The presentation method of the analysis result is a useful means to see software and the development process objectively.	The data collection doesn't disturb the development. The project transitions can be objectively understood from the collected data.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Making of EPM Open Source

- Japanese version in June, 2005
 - Downloaded about 300 so far
- English version in August, 2005
- License
 - Empirical Project Monitor License (EPML) that fixed the special contract articles to Common Public License (CPL) was made.
 - The user selects either CPL or EPML.

CPL
License regulations of U.S. IBM made based on IBM Public License. When software that combines the source code of CPL with the source code of original development is made, and the object code is distributed, **it has to open only the part of the source code of CPL** to the public.

EPML: Special contract of articles
The modification code need not be opened to the public, and be delivered to the project member the distribution ahead when the change to EPM and the source code (modification code) in an additional part are distributed or the individual who made the modification code distributes the modification code that hangs to the main employment corporation or individual moreover by the joint research project etc. to develop software by using EPM.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Lessons Learned in EASE

- The data collection cost is very low compared with the development cost.
 - It reached 10-20% of the development cost so far.
- Data analysis costs must be reduced.
 - As application experience increases, more systematic *reuse of the analysis work process* is expected.
- Clarification of concrete needs for analysis.
 - Example: "Analysis at the program module level and management of unexpected values are necessary."
- Grasp of the situation and detection of abnormalities can be done using only collected data and analysis results.
 - Even inexperienced students identified significant points and understood the software construction process more clearly using EPM data and analysis.



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Resources

<http://www.empirical.jp/English/>



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Overall Summary



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Talk Summary

- Overview of Osaka University and Software Engineering Lab
- Code Clone Analysis and Application
- Component Ranking based on Use Relation
- Empirical Approach to Software Engineering



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Collaboration

- Very good SE research needs collaboration between
 - Academia and industry
 - Academia and academia
- Always seeking various levels of collaboration
 - Joint research
 - Workshop
 - Student exchange
 - ...
- Contact
 - inoue@ist.osaka-u.ac.jp
 - <http://sel.ist.osaka-u.ac.jp/>



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

Thank you !



Software Engineering Laboratory, Department of Computer Science, Graduate School of Information Science and Technology, Osaka University