Life beyond CMMI Level 5: The Empirical Approach to Software Engineering

Dr. Katsuro Inoue
Prof. Michael Barker
9/5/2006 (v9)
Introductions

- Who are we?
- Who are you?
- What is EASE?
- Plans for the day
Who are we?

- Dr. Katsuro Inoue, Osaka University
  co-leader of the EASE project
- Professor Mike Barker, Nara Institute of Science and Technology
  Researcher, EASE project
Some Rules for the Day

- First, informal meeting. Please ask questions!
- Second, each of you should make a “take aways” page. This is just a single page of paper with whatever key points you would like to remember. NASSCOM has suggested that if you will provide your “take aways” page, they will collate and then provide a copy of the consolidated page to everyone.
Who are you?

- Please pair up
- Introduce yourself to your partner.
  - What is your name?
  - Where are you from?
  - Why are you interested in this tutorial?
- Now, introduce the rest of us to your partner.
What kind of work do you do?
- Quality/Process improvement
- Software engineering
- Project management
- Upper management
- Consultants
- Other?

Are you a software consumer? Producer?

Industry? Academic? Government?
Who are you? (specialization)

Categories
- Quality/PI
- Software Engineers
- Project Managers
- Upper Management
- Consultants
- Others
Who are you?

- Consumer
- Producer
Who are you?
Who are you?

Which of these are you interested in?

- CMMI
- Empirical Methods
- Data collection and measurement
- Data analysis
- Data feedback
- Interaction of quality and empirical methods
- Japan
Who are you? (Interests)

Interest Problems

- CMMI
- Empirical Methods
- Data collection and measurement
- Data analysis
- Data feedback
- Interaction of quality and empirical methods
- Japan
What is EASE?

Sit back and relax a moment, this is our advertisement 😊
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: $2 million US / year.
- Project leader: Koji Torii, NAIST
  
  Sub-leader: Katsuro Inoue, Osaka University
  Kenichi Matsumoto, NAIST

http://www.empirical.jp/English/
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 (e.g. Toshiba, Unisys Japan, Fuji Film)
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
The EASE roadmap

Social impact Effectiveness

Data collection
- Upstream Product Data
- Production Data

Data analysis
- Estimation
- Suggestion

Feedback
- Alternatives
- Related Cases

Sharing
- Industry Level Sharing
- Sharing Between Organizations
- Sharing Between Projects
- Sharing Between Developers

Data Sharing

Communication Data
- Quality Data
- Downstream Product Data

Characterization
- Exception Analysis
- Analysis Results

Collected data

Project Plan Data


Software Development With High Reliability and Productivity

Software Development With High Reliability and Productivity

Quality Development With High Reliability and Productivity

Software Development With High Reliability and Productivity

Software Development With High Reliability and Productivity

Software Development With High Reliability and Productivity

Software Development With High Reliability and Productivity

Software Development With High Reliability and Productivity

Software Development With High Reliability and Productivity
Software Engineering Center (SEC) Japan is a project under METI

Strong cooperation

SEC (IPA) has issued an RFP to develop a commercial quality toolset using the existing EASE EPM and collaborative filtering. This three year project has the following stages:

✪ Develop easy-to-use distribution kit of measurement tools
✪ Practical usage in ten trial projects
✪ Propose service business using measurement database
The EASE Empirical Tool

- EPMplus measures
- CF Project Selection
- CF Estimation

SEC Benchmark Database (aka 1000 projects DB)

CF Selection

Selected Similar Projects

Current Project EPMplus

Current Project Estimated by CF

<table>
<thead>
<tr>
<th>Dev. Type</th>
<th>Language</th>
<th>Function Point</th>
<th># of Staffs</th>
<th>Dev. Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Java</td>
<td>20000</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>(MV)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>Java</td>
<td>25000</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>(MV)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>(MV)</td>
<td>5000</td>
<td>20</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Prediction: 59.99
A new research framework for applying empirical software engineering methods in industrial practice and accomplishments in using it.

The selected target: a governmentally funded software development project involving multiple vendors.

In-process project data measurement in real time.

Data sharing with industry and academia (I & A).

Data analysis, and feedback to the project members.

*) EPM: Empirical Project monitor
<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00-10:00</td>
<td>Introductions</td>
</tr>
<tr>
<td>10:15-11:15</td>
<td>Collecting empirical data</td>
</tr>
<tr>
<td>11:30-12:30</td>
<td>Analyzing empirical data</td>
</tr>
<tr>
<td>12:30-1:30</td>
<td>Lunch</td>
</tr>
<tr>
<td>1:30-2:30</td>
<td>Distributing the results</td>
</tr>
<tr>
<td>2:45-3:45</td>
<td>Using empirical methods for quality improvement</td>
</tr>
<tr>
<td>4:00-5:00</td>
<td>Next steps, summary and conclusions</td>
</tr>
</tbody>
</table>
BREAK!
Collecting Empirical Data
Collecting Empirical Data

1. Why collect data? The Empirical answer
2. GQM Planning
3. An Exercise
4. Group Data
Why do we collect data?

- How are you going to use it?
- Empirical approach use experimentation or intervention model to measure effects.
Empirical Methodologies

- Why do we need empirical evidence?
- How can we get it?
Empirical Methodologies

- Why Empirical Approach?
- Software engineering practical
- Criteria for success include quality such as:
  - Accuracy
  - Appropriateness
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability
  - Timeliness
  - Cost effectiveness
  - Customer satisfaction
- Human variation of production
- Environment variation
- Wide variety of products
- Hard to establish guiding principles
- Tendency to base practice on experience, hearsay, and general folklore and myth

Empirical Methodologies

To be engineering, we need:
- Observations of working practices
- Theories and hypotheses
- Testing to validate

Two Views of The World

- Facts and laws must be universally true
- Largely rely on controlled experiments, isolating independent and dependent variables to establish cause and effect
- Replication

Positivism

- It depends on the context
- Understand phenomena through meanings and values that people assign
- Explore and explain how all the factors in study are related and interdependent
- Unique cases

Interpretivism

Instant Poll!

Which View?

- Universal Laws?
- Context Rules?
A Pyramid of Empirical Research Types

![Diagram of research types]

- **Positivism**
  - Controlled Experiments
  - Experiments with students
- **Interpretivism**
  - Surveys, multiple case studies
  - Case studies, action research, ethnography
  - Anecdotes, story telling, diaries

## Framework For Mixed Method Research

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Sequential</th>
<th>Parallel</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangulation (corroboration)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complementarity (elaboration)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initiation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expansion</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## A Typology of Research Methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects Study of Objects</td>
<td>Unique Objects</td>
<td>History</td>
<td>Case Study Post Hoc Analysis</td>
</tr>
<tr>
<td>Class or Set of Objects</td>
<td>Survey</td>
<td></td>
<td>Experiment Meta-analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Researcher's Assumption About Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contextual (Kairotic)</td>
</tr>
<tr>
<td>Universal (Chronotic)</td>
</tr>
</tbody>
</table>

Counelis (2000).
<table>
<thead>
<tr>
<th>Aim</th>
<th>Evidence</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop understanding of nature of SE practice</td>
<td>Natural data</td>
<td>Field or case study</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interviews, think-aloud</td>
</tr>
<tr>
<td>Exploring how practice might be improved</td>
<td>Management evidence</td>
<td>Quantitative experiments</td>
</tr>
<tr>
<td></td>
<td>Practitioner evidence</td>
<td>Case studies</td>
</tr>
<tr>
<td>Evaluation of effects of introducing improvement into practice</td>
<td>Baseline intervention measure qualitative</td>
<td>Field or case studies</td>
</tr>
</tbody>
</table>

Segal (2004).
Four General Categories

- Scientific method: theory, hypothesis and experimentation
- Engineering method: develop and test a solution to a hypothesis, test and improve
- Empirical method: statistical method used to validate hypothesis
- Analytic method: formal theory, deductions compared with empirical observations

Zelkowitz and Wallace (1998)
Aspects of data collection

- Replication: can we do it again?
- Local control: can we control the treatment?
- Influence: does the experimental design assumed passive objects or active objects
- Temporal properties: is the data collection historical or current?

Zelkowitz and Wallace (1998)
## SE Validation Methods

<table>
<thead>
<tr>
<th>Category</th>
<th>Validation method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observational</td>
<td>Project monitoring</td>
</tr>
<tr>
<td></td>
<td>Case study</td>
</tr>
<tr>
<td></td>
<td>Assertion</td>
</tr>
<tr>
<td></td>
<td>Field study</td>
</tr>
<tr>
<td>Historical</td>
<td>Literature search</td>
</tr>
<tr>
<td></td>
<td>Legacy</td>
</tr>
<tr>
<td></td>
<td>Lessons learned</td>
</tr>
<tr>
<td></td>
<td>Static analysis</td>
</tr>
<tr>
<td>Controlled</td>
<td>Replicated</td>
</tr>
<tr>
<td></td>
<td>Synthetic</td>
</tr>
<tr>
<td></td>
<td>Dynamic analysis</td>
</tr>
<tr>
<td></td>
<td>Simulation</td>
</tr>
</tbody>
</table>

Zelkowitz and Wallace (1998)
Six Topic Areas

1. Experimental context
2. Experimental design
3. Conduct of the experiment in data collection
4. Analysis
5. Presentation of results
6. Interpretation of results

1. Be sure to specify as much of the industrial context as possible. In particular, clearly define the entities, attributes, and measures that are capturing the contextual information.

2. If a specific hypothesis is being tested, state it clearly prior to performing the study and discuss the theory from which it is derived, so that its implications are apparent.

3. If the research is exploratory, state clearly prior to data analysis what questions the investigation is intended to address and how it will address them.

4. Describe research that is similar to, or has a bearing on, the current research and how current work relates to it.

Experimental Design (1)

1. Identify the population from which the subjects and objects are drawn.
2. Define the process by which the subjects and objects were selected.
3. Define the process by which subjects and objects are assigned to treatments.
4. Restrict yourself to simple study designs or at least two designs that are fully analyzed in the statistical literature.
5. Define the experimental unit.
6. For formal experiments, perform a pre-experiment or precalculation to identify or estimate the minimum required sample size.

Experimental Design (2)

7. Use appropriate levels of blinding.
8. If you cannot avoid evaluating your own work, and make explicit any vested interests (including your sources of support) and report what you have done to minimize bias.
9. Avoid the use of controls unless you are sure the control situation can be unambiguously defined.
10. Fully define all treatments (interventions).
11. Justify the choice of outcome measures in terms of their relevance to the objectives of the empirical study.

Conducting the Experiment and Data Collection

1. Define all software measures fully, including the entity, attribute, unit and counting rules.
2. For subjective measures, present a measure of interrater agreement, such as the kappa statistic or the intraclass correlation coefficient for continuous measures.
3. Describe any quality control method used to ensure completeness and accuracy of data collection.
4. For surveys, monitor and report the response rate and discuss their representativeness of the responses and the impact of nonresponses.
5. For observational studies and experiments, record data about subjects who drop out from the studies.
6. For observational studies and experiments, record data about other performance measures that may be affected by the treatment, even if they are not the main focus of the study.

Analysis

1. Specify any procedures used to control for multiple testing.
2. Consider using blind analysis.
3. Perform sensitivity analysis.
4. Ensure that the data do not violate the assumptions of the tests used on them.
5. Apply appropriate quality control procedures to verify your results.

Presentation of Results

1. Describe or cite a reference for all statistical procedures used.
2. Report the statistical package used.
3. Present quantitative results as well as significance levels. Quantitative results should show the magnitude of the effects and the confidence limits.
4. Present the raw data whenever possible. Otherwise, confirm that they are available for confidential review by the reviewers and independent auditors.
5. Provide appropriate descriptive statistics.
6. Make appropriate use of graphics.

Interpretation of Results

1. Define the population to which inferential statistics and predictive models apply.
2. Differentiate between statistical significance and practical importance.
3. Define the type of study.
4. Specify any limitations of the study.

On-line Surveys
Types

- Survey types
  - Mail surveys
  - Street surveys
  - Telephone surveys
  - Electronic surveys

- Descriptive or retrospective survey: state-of-the-art overview. E.g., which tools, which reasons, what satisfaction?

- Explorative claims: to discover opinions or relationships in new areas. E.g. evaluate demand for a product or service.

On-Line Surveys: Validity

Compared to experiments and case studies, little control over variables. Low internal validity.

However, a large number of people makes results easier to generalize. High external validity.

On-line Surveys: Advantages and Problems

- Easy for participants: follow link, fill-in form, simple adjustments
- Easy for researcher: data already electronic, response rate easy to manage the
- Problem: lack of response from individuals who are not comfortable with technology

## On-Line Surveys: Process

<table>
<thead>
<tr>
<th>Activity</th>
<th>Purpose</th>
<th>Survey issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Study definition</td>
<td>Determine goal</td>
<td></td>
</tr>
<tr>
<td>Study design</td>
<td>Turn goals into questions and select respondents</td>
<td>Questionnaire design, define target population and sampling procedure, address validity</td>
</tr>
<tr>
<td>Implementation</td>
<td>Make design executable</td>
<td>Check completeness and understandability, define distribution</td>
</tr>
<tr>
<td>Execution</td>
<td>Collect and process data</td>
<td>Monitor responses</td>
</tr>
<tr>
<td>Analysis</td>
<td>Interpret the data</td>
<td>Verify data entry</td>
</tr>
<tr>
<td>Packaging</td>
<td>Report results</td>
<td>Statistics and graphics</td>
</tr>
</tbody>
</table>

On-Line Surveys: Sampling

- Sampling: probability sampling or convenience sampling
- Sampling: personalized or self-recruited

On-Line Surveys: Development Guidelines

- Motivation: why should respondents participate? Access to analysis report
- Number of questions: restrict questions to the topic!
- Type of questions: closed questions are easy to analyze, open questions with short answers provide important background information
- Layout: easy to read, minimize scrolling, provide indication of progress
- Order of questions: maintain motivation. Interesting questions first. Avoid personal questions.
- Provide specific instructions. Avoid mandatory questions. Provide ways for user to quit and restore.
- Ensure anonymity.

Case Studies

- A powerful and flexible empirical method
- Primarily exploratory investigations that attempt to understand and explain a phenomenon or construct a theory
- Generally observational or descriptive

- A popular way to understand, explain, or demonstrate capabilities of a new technique, method, tool, process, technology or organizational structure

Perry, Sim, and Easterbrook (2004).
Case Studies

- Defined method for
  - Posing research questions
  - Collecting data
  - Analyzing the data
  - Presenting results

- NOT
  - Example or case history
  - Experience report
  - Quasi-experimental design with n=1

Perry, Sim, and Easterbrook (2004).
Validity

- External validity: how well do the conclusions apply to other people, in other places, at other times?
- Internal validity: are there other possible causes or explanations?
- Construct validity: how well do the measurements reflect the theories, ideas, or models?

Validity: Did something really happen?

- **Internal Validity**: Did something happen?
  - History: did anything else happen between first and second measurement?
  - Maturation: did the object of study change?
  - Testing: did the first measurement change the second one? (Learning, sensitivity)
  - Instrumentation: did the measurement instrument or observers change?
  - Statistical Regression: if groups were selected based on extreme scores, they will tend to move to mean without treatment
  - Selection: were the treatment and control groups formed in non-random ways?
  - Experimental mortality: were dropped cases connected to the study?
  - Interaction Effects: are there interaction effects that hide the treatment?

- **External Validity**: Can we generalize to similar situations?
  - Reactive or interaction effects: does the treatment or testing hide real effects (aka Hawthorne effect or placebo effect)
  - Interaction of selection and treatment: are the effects due to selection?
  - Reactive effects of experimental arrangement: is the experimental environment so different from real world that generalization is not possible?
  - Multiple treatment interference: multiple treatments of same cases is not the same.

Campbell and Stanley research design summary, found at [http://www.csupomona.edu/~tom/lib/senior_projects/research_design_summary.pdf](http://www.csupomona.edu/~tom/lib/senior_projects/research_design_summary.pdf)
How do you collect data?

- What kind of data?
- Surveys, questionnaires, interviews, etc.
- EPM, Hackystat, PSP/TSP
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
Implementation of the Empirical Project Monitor (EPM)

Co-existing tools
- Code clone detection
- Component search
- Metrics measurement
- Logical Coupling
- Collaborative filtering

Plug-in
- Format Translator
- Format Translator
- Format Translator

Product data archive (CVS format)
Process data archive (XML format)

Source Share GUI

Other tool data

Managers
Developers

Core EPM
Versioning (CVS)
Mailing (Mailman)
Issue tracking (GNATS)

Project x
Project y
Project z
...
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
EPM can put data collected by CVS, Mailman, and GNATS together into one graph.
How do you decide what to measure?
GQM Approach

- Goal
- Questions/Model
- Metrics
The GQM Paradigm: An Example

Goal

Analyze CVS and GNATS data for file change patterns, for the purpose of evaluation, with respect to requirements instability, poor design, or poor product quality, from the point of view of the project manager, in the context of the particular project in the company.

Question

File change level (FCM)?

The number of lines changed (LCC)?

The number of developers making changes to files (ONR)?

The reason for the change?

Change frequency of each file (CVS)

Size of file (CVS)

Number of developers making changes to files (CVS)

Metric

The changed number of lines in each file (CVS)

Change motive: Bug or specification change (GNATS)
The GQM Template

- **object of study:** a process, product or any other experience model
- **purpose:** to characterize (what is it?), evaluate (is it good?), predict (can I estimate something in the future?), control (can I manipulate events?), improve (can I improve events?)
- **Focus:** what aspects of the object of study are of interest
- **point of view:** from what perspective; who needs this information
- **context:** what is the environment where the measurement will be taken
<table>
<thead>
<tr>
<th><strong>The GQM Template</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What</strong> do we want to measure?</td>
</tr>
<tr>
<td><strong>Why</strong> do we want to measure it?</td>
</tr>
<tr>
<td><strong>What aspect</strong> are we interested in?</td>
</tr>
<tr>
<td><strong>Who</strong> wants to know?</td>
</tr>
<tr>
<td><strong>Where</strong> are we going to work?</td>
</tr>
</tbody>
</table>
Team Exercise

- Pick one objective or goal
- Write the GQM template statement
- Develop some questions or model to meet that goal
- Design metrics
- Describe activities to be measured
- Consider how to integrate measurements into activities – what measurement tools could you use?
CMMI Levels

1. Initial: Competent people and heroics
2. Repeatable: basic project management
3. Defined: process standardization
4. Managed: quantitative management
5. Optimizing: continuous process improvement
CMMI Levels

- Initial
- Repeatable
- Defined
- Managed
- Optimizing
Data Collections

Levels

- Individual
- Team
- Project
- Department
- Company
- Consortium
- Industry
Break!
Analyzing Empirical Data
Analysis targets

Manager support

- Estimation of requirements for redesign caused by connections in the system structure
- Making similarity visible
- Class and utility evaluation of information related to use of methods

Developer support

- Evaluation of clone distribution
- Evaluation of component ranking
- Expert identification
- Distinction of modules (file) with high defect rates

The project delay risk detection model

The project management models for trouble evaluation

Estimate of effort to correct based on file scale, number of accumulated defects, and defect type

ID candidates for refactoring

Grasp of the situation Abnormal detection Forecast Advice

Collection, analysis, and search engine of software products (SPARS)

Code clone analysis

Collaborative filtering

Logical coupling

GQM (Goal/Question/Metric) Model

NASSCOM Quality Summit

~73~
Analyzing Empirical Data

- Analysis of potential project delay risks
- Logical coupling
- Defect correction analysis
- Collaborative filtering
- Extracting patterns through association rule mining
- Analysis in your hands?
Detect risks of project delay (e.g. Unstable Requirements, Incomplete Designs, Low Quality Program or Inappropriate Resource Planning) by monitoring the program change history.
Risk Detection Preventing Project Delay(2)

- Give the alert for project managers depending on a certain threshold
  - E.g. change frequency weekly for each module
  - A: Number of updates weekly

Approximately 30% of changes include a certain amount of line deletion normally -> “more than 30%” implies that some problems happened.
Logical Coupling Analysis

- Discover implicit knowledge for system maintenance to reduce mistakes or lack of needed changes
  - Relationships among files (modules) frequently changed at the same time
Complication analysis by logical coupling

- Logical Coupling
  - A logical relationship between software modules where changes in a part of one module require changes in a part of another module.

- By knowing logical coupling, the cause of a module’s complexity becomes clear, providing hints about module structure and where restructuring may be needed.

![Diagram showing logical coupling and its impact on module complexity and file changes.](image-url)
**Logical Coupling Analysis(1)**

- Analyze change history from Configuration Management System using association rule mining


### Logical Coupling Analysis (2)

- **Analyze each pattern by confidence**
  - **Association Rule**: \( X \) (Antecedent) \( \Rightarrow \) \( Y \) (Consequent)
  - **Confidence** (\( X \Rightarrow Y \)) : \( \# \text{ of Update} (X \cup Y) / \# \text{ of Update} (X) \)

<table>
<thead>
<tr>
<th>Confidence</th>
<th># of Update (Antecedent)</th>
<th># of Update (Consequent)</th>
<th>Antecedent</th>
<th>Consequent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>5</td>
<td>4</td>
<td>FileA.cpp, FileB.h, FileC.cpp</td>
<td>FileD.h</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>FileA.cpp, FileB.h, FileD.h</td>
<td>FileC.cpp</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>FileA.cpp, FileC.cpp, FileD.h</td>
<td>FileB.h</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>FileA.cpp, FileC.cpp, FileD.h</td>
<td>FileA.cpp</td>
</tr>
</tbody>
</table>

**Has “FileD.h” missed a change?**

**Was “FileY.cpp” copied from “FileX.cpp”? Or does “FileY.cpp” strongly depend on the “FileX.cpp”?**

<table>
<thead>
<tr>
<th>Confidence</th>
<th># of Update (Antecedent)</th>
<th># of Update (Consequent)</th>
<th>Antecedent</th>
<th>Consequent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>8</td>
<td>4</td>
<td>FileX.cpp</td>
<td>FileY.cpp</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>FileY.cpp</td>
<td>FileX.cpp</td>
</tr>
</tbody>
</table>
Factor Analysis of Defect Correction Effort

- Improve software process taking into account return on investment
E.g.) Defect Correction Effort classified by “Reason why the defect was not detected in the preferable phase”

These factors seem to influence the correction effort:
- Lack of Test Cases
- Misjudged test results
- Defects depending on the environment

Do any relations to another factors (e.g. detected phase) exist?
Factor Analysis of Defect Correction Effort (2)

- E.g.) Relation Analysis between “Reason why the defect was not detected in the preferable phase” and “Defect detected phase” using Parallel Coordinate Plot

- Lack of Test Cases
- Misjudged test results
- Defects depending on the environment
Estimation Using Collaborative Filtering: Robust for Missing Values
Collaborative Filtering

- Robust estimation method with missing data
- Applicable to estimating various attributes of project/system from similar project/system profiles

<table>
<thead>
<tr>
<th></th>
<th>Focused</th>
<th>Representative</th>
<th>Q &amp; M Resources</th>
<th>Collaborative</th>
<th>Outcome Adopted</th>
</tr>
</thead>
<tbody>
<tr>
<td>App. A</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>7</td>
<td>7.5 (target)</td>
</tr>
<tr>
<td>App. B</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>? (missing)</td>
<td>8</td>
</tr>
<tr>
<td>App. C</td>
<td>? (missing)</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>App. D</td>
<td>7</td>
<td>6</td>
<td>? (missing)</td>
<td>9</td>
<td>6</td>
</tr>
</tbody>
</table>
For development planning, practitioners can use estimation methods with historical projects’ data, but...

- **Historical data usually contain many *Missing Values.***

- **Missing values reduce the accuracy of estimation.**
Goal and Approach

Goal: to establish an estimation method which treats historical projects’ data containing many missing values.

Approach: use Collaborative Filtering (CF).

◆ A technique for estimating users’ favorite items with data containing many missing values (e.g. Amazon.com)
Estimation of Customer X’s favorite (rating on book 5).

- **Step 1: Similarity Computation**
  - Calculating the similarity between *Customer X* (current estimation target) and other customers.
  - Selecting top-\(k\) similar customers (e.g. \(k = 2\)).

- **Step 2: Prediction**
  - Calculating a prediction of customer X’s rating on book 5 using the similar customers’ rating on book 5.

<table>
<thead>
<tr>
<th></th>
<th>Book 1</th>
<th>Book 2</th>
<th>Book 3</th>
<th>Book 4</th>
<th>Book 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer X</td>
<td>1 (hate)</td>
<td>2 (unlike)</td>
<td>4 (like)</td>
<td>5 (love)</td>
<td></td>
</tr>
<tr>
<td>Similarity: +1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer A</td>
<td>1 (hate)</td>
<td>2 (unlike)</td>
<td></td>
<td>5 (love)</td>
<td>5 (love)</td>
</tr>
<tr>
<td>Similarity: +0.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer B</td>
<td>2 (unlike)</td>
<td></td>
<td>? (unread)</td>
<td>4 (like)</td>
<td>5 (love)</td>
</tr>
<tr>
<td>Similarity: -0.97</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer C</td>
<td>? (unread)</td>
<td>4 (like)</td>
<td>2 (unlike)</td>
<td>1 (hate)</td>
<td>1 (hate)</td>
</tr>
</tbody>
</table>

**Prediction: 4.51**
Step 1: Similarity Computation
- Calculating the similarity between Project X (current estimation target) and other past projects.
- Selecting top-\(k\) similar projects (e.g. \(k = 2\)).

Step 2: Prediction
- Calculating a prediction of project X’s cost using the similar projects’ cost.

<table>
<thead>
<tr>
<th>Project</th>
<th>Language</th>
<th>Dev. Type</th>
<th>Function Points</th>
<th># of Staffs</th>
<th>Dev. Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project X</td>
<td>Java</td>
<td>New</td>
<td>2000</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Project A</td>
<td>Java</td>
<td>New</td>
<td>? (MV)</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>Project B</td>
<td>Java</td>
<td>? (MV)</td>
<td>2500</td>
<td>10</td>
<td>70</td>
</tr>
<tr>
<td>Project C</td>
<td>? (MV)</td>
<td>Maintenance</td>
<td>5000</td>
<td>20</td>
<td>250</td>
</tr>
</tbody>
</table>

Prediction: 59.99
Methodology: Cross-Validation of Cost Estimation
Data Size: 10 variables and about 140 projects
Missing Value Ratio: from 0 to 50% (by 10%)
Association rule mining for software project data
A retail sales analysis example:
“if A is bought from 12:00~14:00 then B is bought in 80% of the cases, too.” is obtained from POS histories.
From 12:00~14:00, retailer can locate goods B next to goods A.

A software project analysis example would be…
From project development data,
“if the development type is enhancement and code developers are outsourced, then the testing phase is longer in 70% of the cases.” is obtained.
When doing enhancement development and coding phase is outsourced, the testing phase should be estimated longer than usual.
**Project dataset (project descriptive data) has**
- rows correspond to individual project
- columns correspond to data field

<table>
<thead>
<tr>
<th>ID</th>
<th>Development Type</th>
<th>Organization Type</th>
<th>Resource Level</th>
<th>Effort Specify Ratio</th>
<th>Effort Coding Ratio</th>
<th>Effort Test Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>New Development</td>
<td>Banking</td>
<td>A</td>
<td>80</td>
<td>230</td>
<td>200</td>
</tr>
<tr>
<td>0002</td>
<td>Enhancement</td>
<td>Construction</td>
<td>B</td>
<td>120</td>
<td>200</td>
<td>360</td>
</tr>
<tr>
<td>0003</td>
<td>Enhancement</td>
<td>Public Administration</td>
<td>B</td>
<td>60</td>
<td>260</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extracted rule is in the form “if A then B”.
- A, B: categorical value
  - ex) if (development type = enhancement) and
    (effort specify ratio = small) then (effort test = large)

Rules can be unified using “or”.
- ex) if (organizational type = banking) then (effort test ratio
  = medium or large)
NEEDLE: Case Study

- Extracted rules from 37 real system integration projects data with 40 data fields.
- E.g. *if* known customer *and* known industry / business *and* without middleware for specific industry / business *then* ratio of staff-month in design phase is 1 *or* 2 *or* 3. (1: lowest, 9: highest). 37.8%(support)

known: having experience to develop with
Current EASE Research Topics

- Developer Role Categorization
- Correspondence Analysis
- Personal EPM
Analysis in your hands

- In teams
- Select a process or project that you could collect data from (or already do)
- How do you analyze this data? Which of the following terms help:
  - Compare
  - Contrast
  - Abstract
  - Model
  - Hypothesis
  - Context
  - Relationships
Which area needs analysis most?

- Requirements elicitation
- Design
- Development
- Integration
- Testing
- Operations
- Maintenance
Which area most needs analysis?

![Bar chart showing lifecycle stages: Requirements, Design, Development, Integration, Testing, Operations, Maintenance.](chart.png)
Which Analysis Method Is Most Useful for You?

- Potential Delay Risks
- Logical coupling
- Defect correction
- Collaborative filtering
- Association Rule Mining
Which Analysis Method is Most Useful for You?

- Potential Delay Risks
- Logical coupling
- Defect correction
- Collaborative filtering
- Association Rule Mining
Break!
Distributing the Results
Distributing the Results

- Feedback through reports, training, simulations, plans, etc.
- One approach: the Experience Factory
- Replayer/Simulator
- Workshops, tutorials, training
Simulations
Most documented software failures can be attributed to software engineering process breakdowns.

The root cause of this is lack of practice with issues surrounding the software engineering process.

Navarro and van der Hoek (2004).
Components

- Teams of people
- Large-scale projects
- Critical decision-making
- Personnel issues
- Multiple stakeholders
- Budgets
- Planning
- Random, unexpected events

Navarro and van der Hoek (2004).
SIMSE

- Tradeoffs: faithfulness to reality, level of detail, usability, teaching objectives, and fun factors
- Guidelines
  - SimSE should illustrate both specific lessons and overarching practices of the software process
  - SimSE should support the instructor in specifying the lessons he or she wishes to teach
  - SimSE should provide a student with clear feedback concerning their decisions
  - SimSE should be easy to learn, enjoyable, and comparatively quick

Navarro and van der Hoek (2004).
Why is simulation successful?

- Simulation allow students to gain variable hands-on experience of the process being simulated without monetary costs or harmful effects of real world experience
- Simulations can be repeated, allowing experimentation with different approaches
- Relative ease of configuration allows educator to introduce a wide variety of unknown situations
- Simulation can be run at faster pace, allowing students to practice process many more times than feasible in real world

Navarro and van der Hoek (2004).
Simulation Games

- Problems and programmers, an educational software engineering card game
  - Two-person game
  - Balance budget and reliability
- Concept, programmer, and problem cards
  - Concept: decision regarding approach
  - Programmer: skill, personality, salary
  - Problem: various project problems
- Pick a project card to start game: complexity, length, quality, budget

Models in Software Engineering

- Models have three properties:
  - Mapping: there is an original object or phenomenon represented by the model
  - Reduction: the model does not represent all properties, but does represent some
  - Pragmatic: for some purposes, the model can replace the original, so it is useful.

- Mapping: mapped, removed, and added attributes

Ludewig (2003).
Models in Software Engineering

- Descriptive: mirrors existing original
- Prescriptive: used to create original
- Transient: descriptive, but then modified to guide changes to original

Ludewig (2003).
Models in Software Engineering

Purpose:
- Documentation
  - Concise descriptions
  - Minutes, protocols, logs
  - Metrics
- Instructions
- Exploratory models
- Educational models and games
- Formal or mathematical models

Ludewig (2003).
Why a simulator? Knowledge and skills. Simulator provides experience in many different scenarios without high risk and expense. Provides hands-on, active experience to build effective skills.

Areas which can benefit from simulation:
- Cost assessment
- Practicing metric collection
- Building consensus and communication skills
- Requirements management
- Project management
- Training
- Process improvement
- Risk management
- Acquisition management

Collofello (2000).
Simulation Based Project Management Training

- The simulator allows various user inputs, both initially and during a run, such as:
  - Planned completion time
  - Project complexity
  - Time allocations
  - Communication overhead
  - Staffing, including various levels of expertise

- Various output monitors, such as
  - Current staff load
  - Elapsed man hours and days
  - Remaining hours
  - Schedule pressure gauge
  - Exhaustion rate gauge
  - Percent completion
  - Earned Value outputs

Collofello (2000).
Simulation Based Project Management Training

- Kinds of exercises
  - Life Cycle model comparison: simulate using waterfall and sequential incremental development, with varying inspection effectiveness, project complexity, and staffing levels
  - Risk management, with various contingency plans and personnel losses
  - Software inspections: effect of varying inspections on test time and project completion
  - Critical path scheduling: vary the skill levels of staff on the critical path and assess impact
  - Overall planning and tracking: Plan and execute a project from inception to completion, adjusting to events.

Collofello (2000).
Simulation in SE Training

- SESAM: Software Engineering Simulation by Animated Models
- Focus on motivation: it is hard for students to imagine project management failure because the projects they normally experience are small. SESAM provides an opportunity to experience large project without real risk or expense.

Drappa and Ludewig (2000).
Simulation in SE Training

- During game, restricted information like real project manager receives.
- After game, significant extra information available.

Drappa and Ludewig (2000).
Simulation in SE Training

Observations

- Students do not like planning.
- Students make the same mistakes again and again.
- Students do not reflect on why they fail.

Drappa and Ludewig (2000).
Project Replayer
An Investigation Tool to Revisit Processes of Past Project

Keita Goto*, Noriko Hanakawa**, and Hajimu Iida*

* Graduate School of Information Science, NAIST, Japan
** Faculty of Management Information, Hannan Univ., Japan

http://sdlab.naist.jp/
A Problem in Recent Software Development (1/2)

It is difficult to accumulate knowledge and experiences in organizations.

Because

- Software life cycle is getting shorter (e.g. software of mobile phone).
- Team members frequently change.
A Problem in Recent Software Development (2/2)

The same mistakes are repeated in an organization.

Past project
A mistake was made.

By a developer

Knowledge
Developer learned
- Why this mistake happened.
- How to avoid such mistakes.

New project
The same kind of the mistake was made again.

By a new developer

One year after

The knowledge was not transferred.

- Short development periods
- High fluidity of developers
Goal

- Establishing knowledge feedback cycle from past projects to new projects.

Approach

- We propose a Knowledge Feedback Cycle (KFC) framework.
- The following 3 tools play very important roles in the KFC framework.
  - **Empirical Project Monitor (EPM)**: automatically collects software project data. (EPM was made by EASE project Japan)
  - **Project Replayer**: replays history of a past project for researchers to help extracting knowledge.
  - **Project Simulator**: reuses the extracted knowledge to provide estimations for new projects.
Outline

- A Problem in Recent Software Development
- Goal and Approach

- Knowledge Feedback Cycle (KFC)
- Project Replayer (one of 3 key tools in the KFC)
  - Features of Project Replayer
  - Preliminary experiment
  - Results and discussion
- Conclusions and Future Work
Knowledge Feedback Cycle (KFC) Overview

Developers
Project Execution

Researchers
Project Analysis

Project Data
Tacit knowledge
Knowledge Feedback Cycle (KFC) Overview

Project Execution
- Developer
- Code management log
- Bug tracking log
- E-mailing log
- Tacit knowledge

EPM (Empirical Project Monitor)

Project Analysis
- Researcher
- Project Data Summary
Knowledge Feedback Cycle (KFC) Overview
Knowledge Feedback Cycle (KFC) Overview

Knowledge Feedback Cycle (KFC) Overview

Project Simulator

Explicit knowledge
Simulation Model

Project Replayer

Dynamic View

Developer

Project Execution

Code management log
Bug tracking log
E-mailing log
Tacit knowledge

EPM
(Empirical Project Monitor)

Project Data Summary
Static View

Researcher

Project Analysis & Building Simulation Model

Training/Planning

NAIST Ocase
Support two roles in KFC

- Developers can revisit their past projects for postmortem evaluations
- Researchers can deeply understand and analyze dynamic behavior of the projects
Project Replayer reorganizes collected data by EPM, sorted in time order.
Project Replayer reorganizes collected data by EPM, sorted in time order.
- Project Replayer shows events graphically and simultaneously.
Project Replayer shows events graphically and simultaneously.
Views provided by Replayer

Event List View shows all events listed in time order.

Graph View shows graph of project progress data.

File View shows current status of each source file in project.

Developer View shows current status of each member in project.

Time control slider
Aims of the experiment
1. Confirming benefits of developers and researchers with replaying.
2. Confirming whether tacit knowledge can be transformed to explicit knowledge.
Setting of preliminary experiment

■ Target project
  ◆ Application domain : Typing game
  ◆ Developers : 6 graduate school students
  ◆ Development period : 24 days
  ◆ Program code : 9,578 lines in C++

■ Subjects of experiment
  ◆ 3 subjects as developers (who were project members)
  ◆ 1 subject as researcher (was not a project member)
The experiment was carried out after the target project was finished. Project data was automatically collected using EPM.

- **Phase1: Making hypotheses and questions**
  - The researcher analyzes the project using Project Replayer. The researcher makes hypotheses and questions.
- **Phase2: Answering the questions**
  - The developers find the answers for the questions using Project Replayer.
- **Phase3: Modifying the hypotheses**
  - The researcher modifies the hypotheses in Phase1 according to the answers.
  - If required, the researcher performs additional analysis using Project Replayer.
At first, the researcher made hypotheses using Project Replayer

- (H1) “Modules developed at the final stage of the project have low quality.”
- (H2) “If CVS’s event behavior does not match to bug reports and e-mail data, the project is in confusion, and resulting software has low quality.”

H1: Developers did not start making some modules until a few days before deadline. The researcher expected that these modules have low quality. Therefore, the researcher established H1.

H2: For the last three days, total lines of code did not change. That is, the development has somehow stopped before the deadline. However, the bug reports and e-mail events were occurring during this period. It looked like inconsistency. Therefore, the researcher expected that the project fallen into confusion and the inconsistency was produced. So the researcher established H2.
The researcher made some questions about the development data.

- (Q3) “How was the quality of the module made at the final stage of a project?” (This question related hypothesis, H1)
- (Q4) “Why was not CVS renewed during the last three days?” (This question related hypothesis, H2)

We only show the question directly related to 2 hypothesis.
The developers answered to the questions.
- (Q3) “How was the quality of the module made at the final stage of a project.”
- (A3) “Most of them have good quality except one module.”
- (Q4) “Why was not CVS renewed during the last three days?”
- (A4) “Because the last three days were maintenance phase.”

Those answers were derived with help of Project Replayer.

A3: File view helped to know the name of files developed in the final stage of the project.

A4: Event list view helped developers to recall that the last three days were maintenance stage after the actual deadline.
H1 was not clearly supported by the developers’ answers

- (H1) “Modules developed at the end stage of the project have low quality.”
- (A3) “Most of them have good quality except one module.”

H2 was just withdrawn according to the answers.

Additional analysis was performed using Project Replayer to refine H1.
H1 was changed to the following

(H1’) “Modules developed at the final stage of the project have low quality if the developers have little experience of developing similar functions”

This fact was found by seeing Developer view and File view.
Final results of Preliminary Experiment

- Project Replayer was useful for researchers.
  - To initially build hypothesis.
  - To refine hypothesis.
- Project Replayer was also useful for developers.
  - To recall detail of past project in answering questions.
- H1’ may be regarded as an explicit knowledge derived from the past project.
Conclusions and Future work

- KFC framework was proposed
- Prototype of Project Replayer for KFC was developed
- Preliminary experiment with Project Replayer was carried out
  - Project Replayer was useful for both developers and researchers

Future work
- Further evaluation and validation of Project Replayer
- Development of Project Simulator (another tool in KFC)
Diffusion of Innovations

- Relative Advantage: How much benefit do we get?
- Compatibility: How well does it fit with what we already do?
- Complexity: How hard is it to do?
- Observability: Will anyone notice?
- Trailability: Can we try it out without risk?
Acceptance of New Methods

1. Usefulness: will it help me?
2. Voluntariness: do I have a choice?
3. Compatibility: how well does it fit with values, needs, and past experiences?
4. Subjective norm: do important others think I should use this?
5. Ease of use (or complexity): how hard is it to use?
6. Relative advantage: how much better is it?
7. Result demonstrability: can it show real advantages?
8. Image: Does it make me look better?
9. Visibility: can others see what I’m doing?
10. Career Consequences: what is the long-term payoff for using this?

Riemenschneider, Hardgrave, and Davis (2002).
An Exercise

- In teams
- Estimate the effectiveness of reports, simulations, training in your organization
- Estimate the effort needed (project size) to create reports, simulations, training in your organization
Which do you think would be most effective in your organization?

- Reports
- Simulations
- Training
- Policy Directive

Which do you actually use?
Which is most effective?

Approach

- Reports
- Simulations
- Training
- Policy
Which is most effective?

![Chart showing the comparison of effectiveness of different methods: Reports, Simulations, Training, and Policy. The chart indicates that Reports are the most effective, followed by Simulations, Training, and Policy.]
Which factor is most important for you?

- Relative Advantage: How much benefit do we get?
- Compatibility: How well does it fit with what we already do?
- Complexity: How hard is it to do?
- Observability: Will anyone notice?
- Trailability: Can we try it out without risk?
Which Is Most Important for You?

Factors

- Relative Advantage
- Compatibility
- Complexity
- Observability
- Tailability
Acceptance of New Methods

1. Usefulness: will it help me?
2. Voluntariness: do I have a choice?
3. Compatibility: how well does it fit with values, needs, and past experiences?
4. Subjective norm: do important others think I should use this?
5. Ease of use (or complexity): how hard is it to use?
6. Relative advantage: how much better is it?
7. Result demonstrability: can it show real advantages?
8. Image: Does it make me look better?
9. Visibility: can others see what I’m doing?
10. Career Consequences: what is the long-term payoff for using this?

Riemenschneider, Hardgrave, and Davis (2002).
Using Empirical Methods for quality Improvement
Putting It All Together

First, an example of an empirical tool called CCFinder that we have developed for gathering, analyzing and providing results.

Second, we will consider how you can use empirical methods in your quality improvement.
Code Clone Analysis and Application

Katsuro Inoue
Osaka University
Talk Structure

- Clone Detection
- CCFinder and Associate Tools
- Applications
- Summary of Code Clone Analysis and Application
Clone Detection
What is A Code Clone?

- A code fragment that has identical or similar code fragments in source code
- Code clones are introduced in source code for various reasons
  - code reuse by ‘copy-and-paste’
  - stereotyped function
    - ex. file open, DB connect, …
  - intentional iteration
    - performance enhancement
- It 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. We are likely to overlook some!
AFG::AFG(JaObject* obj) {
    objname = "afg";
    object = obj;
}

AFG::~AFG() {
    for(unsigned int i = 0; i < children.size(); i++)
        if(children[i] != NULL)
            delete children[i];

    ...

    for(unsigned int i = 0;
        i < nodes.size(); i++)
        if(nodes[i] != NULL)
            delete nodes[i];
}
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 of a 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
Clone Pair and Clone Set

- **Clone Pair**
  - a pair of identical or similar code fragments

- **Clone Set**
  - a set of identical or similar fragments
Our Code Clone Research

- Develop tools
  - Detection tool: CCFinder
  - Visualization tool: Gemini
  - Refactoring support tool: Aries
  - Change support tool: Libra

- Deliver our tools to domestic or overseas organizations/individuals
  - More than 100 companies use our tools!

- Promote academic-industrial collaboration
  - Organize code clone seminars
  - Manage mailing-lists
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 a system in the scale of millions of lines in 5-30 min.
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,]+"yna); 
4.   int sum = 0;
5.   for (int i = 0; i < a.length; ++i)
6.       if (pat.match(a[i]))
7.           sum += Sample.parseNumber(pat.getParen(0));
8.   System.out.println("sum = " + sum);
9. }

10. static void goo(String [] a) throws RESyntaxException {
11.   RE exp = new RE("[0-9,]+"yna); 
12.   int sum = 0;
13.   for (int i = 0; i < a.length; ++i)
14.       if (exp.match(a[i]))
15.           sum += parseNumber(exp.getParen(0));
16.   System.out.println("sum = " + sum);
17. }

\textbf{Detection tool:}
\textbf{CCFinder Detection Process}

\begin{itemize}
\item Source files
\item Lexical analysis
\item Transformation
\item Match detection
\item Formatting
\item Clone pairs
\end{itemize}
- Visualize code clones detected by CCFinder
  - CCFinder outputs the detection result to a text file
- Provide interactive analyses of code clones
  - Scatter Plot
  - Clone metrics
  - File metrics
- Filter out unimportant code clones
Visualization tool: 

Gemini Scatter Plot

- Visually show 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.
- Dot means corresponding two tokens on the two axes are the same.
  - Symmetric to main diagonal (show only lower left).

```
ac ac acbb bccabdef cd
F1 F2 F3
F1 F2 F3 F4
D1 D2
D1 D2

F1, F2, F3, F4: files
D1, D2: directories
```
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
- **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 \( F \)
Structural code clones are regarded as a target for refactoring
1. Detect clone pairs by CCFinder
2. Transform the detected clone pairs into clone sets
3. Extract structural parts as structural code clones from the detected clone sets

What is a structural code clone?
- example: Java language
  - Declaration: class declaration, interface declaration
  - Method: method body, constructor, static initializer
  - statement: do, for, if, switch, synchronized, try, while
reset();
grammar = g;
// Lookup make-switch threshold in the grammar generic options
if (grammar.hasOption("codeGenMakeSwitchThreshold")) {
    try {
        makeSwitchThreshold = grammar.getIntegerOption("codeGenMakeSwitchThreshold");
    } catch (NumberFormatException e) {
        tool.error("option 'codeGenMakeSwitchThreshold' must be an integer",
                   grammar.getClassName(),
                   grammar.getOption("codeGenMakeSwitchThreshold").getLine());
    }
}
// Lookup bitset-test threshold in the grammar generic options
if (grammar.hasOption("codeGenBitsetTestThreshold")) {
    try {
        bitsetTestThreshold = grammar.getIntegerOption("codeGenBitsetTestThreshold");
    } catch (NumberFormatException e) {
        tool.error("option 'codeGenBitsetTestThreshold' must be an integer",
                   grammar.getClassName(),
                   grammar.getOption("codeGenBitsetTestThreshold").getLine());
    }
}
// Lookup debug code-gen in the grammar
if (grammar.hasOption("codeGenDebug")) {
    try {
        debugCodeGen = grammar.getBooleanOption("codeGenDebug");
    } catch (NumberFormatException e) {
        tool.error("option 'codeGenDebug' must be a boolean",
                   grammar.getClassName(),
                   grammar.getOption("codeGenDebug").getLine());
    }
}
The following refactoring patterns[1][2] can be used to remove code sets including structural code clones

- Extract Class,
- Extract Method,
- Extract Super Class,
- Form Template Method,
- Move Method,
- Parameterize Method,
- Pull Up Constructor,
- Pull Up Method,

For each clone set, Aries suggests which refactoring pattern is applicable by using metrics.

NASSCOM Quality Summit

~175~
Input a code fragment

```java
if (ret == null) {
    ServerSocketFactory factory = getFactory();
    if (factory instanceof CoyoteServerSocketFactory) {
        return ((CoyoteServerSocketFactory)factory).getKeystorePass();
    }
}
```
Find clones between the input and target
We have periodically organized code clone seminars from Dec 2002.

The seminar is a good place to exchange views with industrial people.

Seminar overview:
- Tool demonstration
- Lecture of how to use code clone information
- Case study of companies using our tools
Summary of Code Clone Analysis and Application
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 projects
What is the lifecycle of quality improvement?

What are the areas of quality improvement?

What are specific tools or interventions that quality improvement recommends?
How do empirical methods relate to quality improvement?

- Relative Advantage: How much benefit do we get?
- Compatibility: How well does it fit with what we already do?
- Complexity: How hard is it to do?
- Observability: Will anyone notice?
- Trailability: Can we try it out without risk?
As a team, consider those questions:
• How would you describe quality improvement?
• How could you use empirical methods for quality improvement?
• What kind of plan would show what you intend to do with quality improvement and identify where you will use empirical methods in that process?
Plan-Do-Check-Act is often given as an approach to quality improvement.

Do you use empirical methods in this cycle? Do you collect data? Do you analyze it? Do you provide feedback?
Collect Data  Analyze  Feedback

Plan

Empirical
Do

Collect Data
Analyze
Feedback

Empirical
Check

![Graph showing Empirical analysis with categories: Collect Data, Analyze, Feedback.](image)
Act

- Collect Data
- Analyze
- Feedback

Empirical
Break!
Review the day

- Introductions
- Data collection: GQM
- Data analysis: collaborative filtering
- Feedback: simulations and training
- Quality Improvement and Empirical Methods: Measure, Analyze, and Feedback
Team Exercise

As a team, list and rank top five points from the day for you.
Quick Collection

Please tell me your top point that has not been mentioned yet!
Large group

- Let them think individually
- Do quick collect as sample
Next Steps?

- These are your ideas. I’d suggest repeating the process we just used to collect key points.

- One possibility: we would like to create and host a conference to foster India-Japanese research exchange. This would be intended for short papers and demos. We are considering hosting the initial meeting in Japan, perhaps with a technology tour.
If you want to send us questions later, send them to mbarker@computer.org
NASSCOM Quality Summit

Instant Vote!

Did you learn something? Useful?

Data Collection
Data Analysis
Feedback
Quality and Empirical
Another Instant Vote!

Were you a good audience?

- Data Collection
- Data Analysis
- Feedback
- Quality and Empirical
- Overall
Thank you all!
どうもありがとうございました

(Remember to turn in your takeaway sheets!)