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.

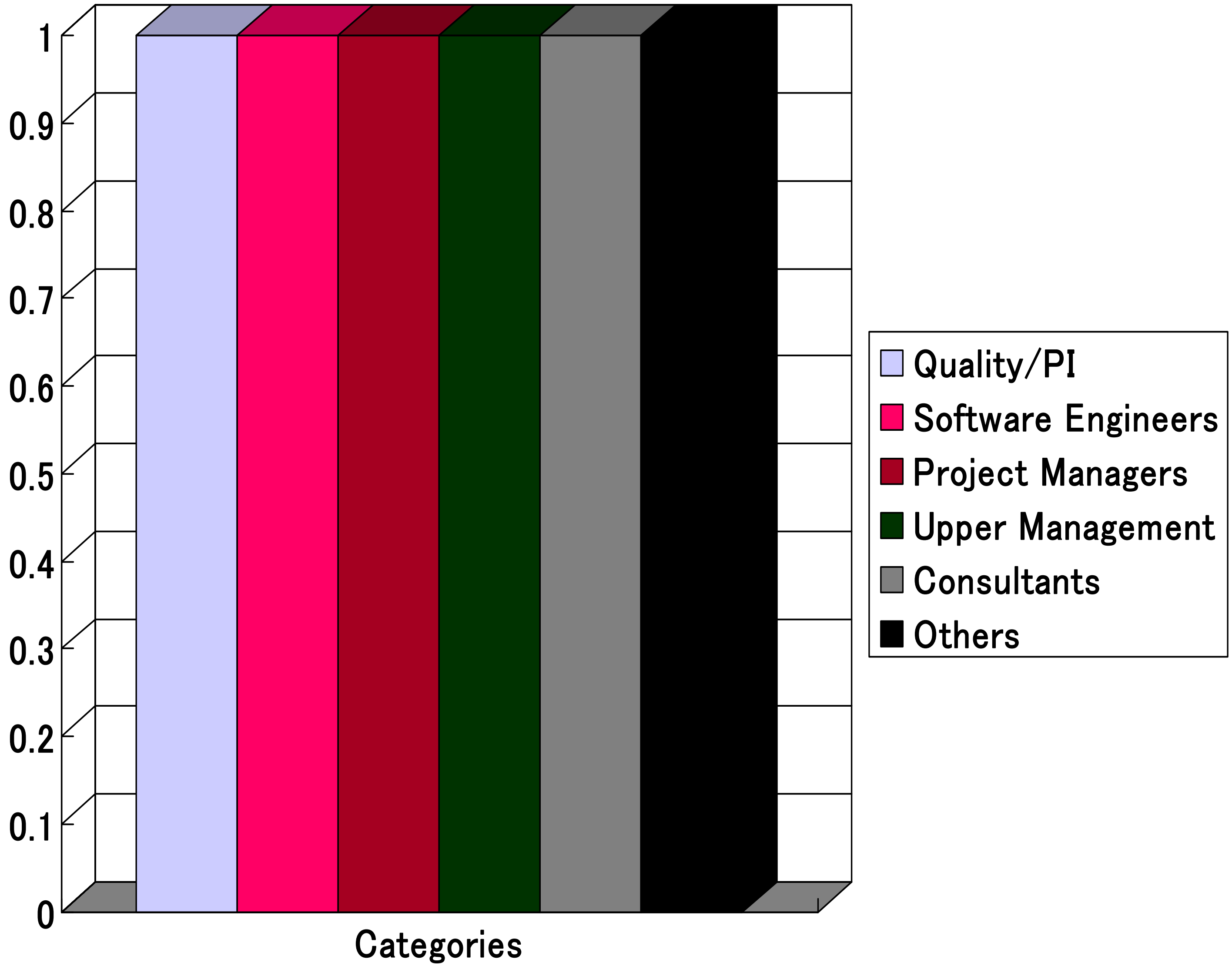
Who are you?

- 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.

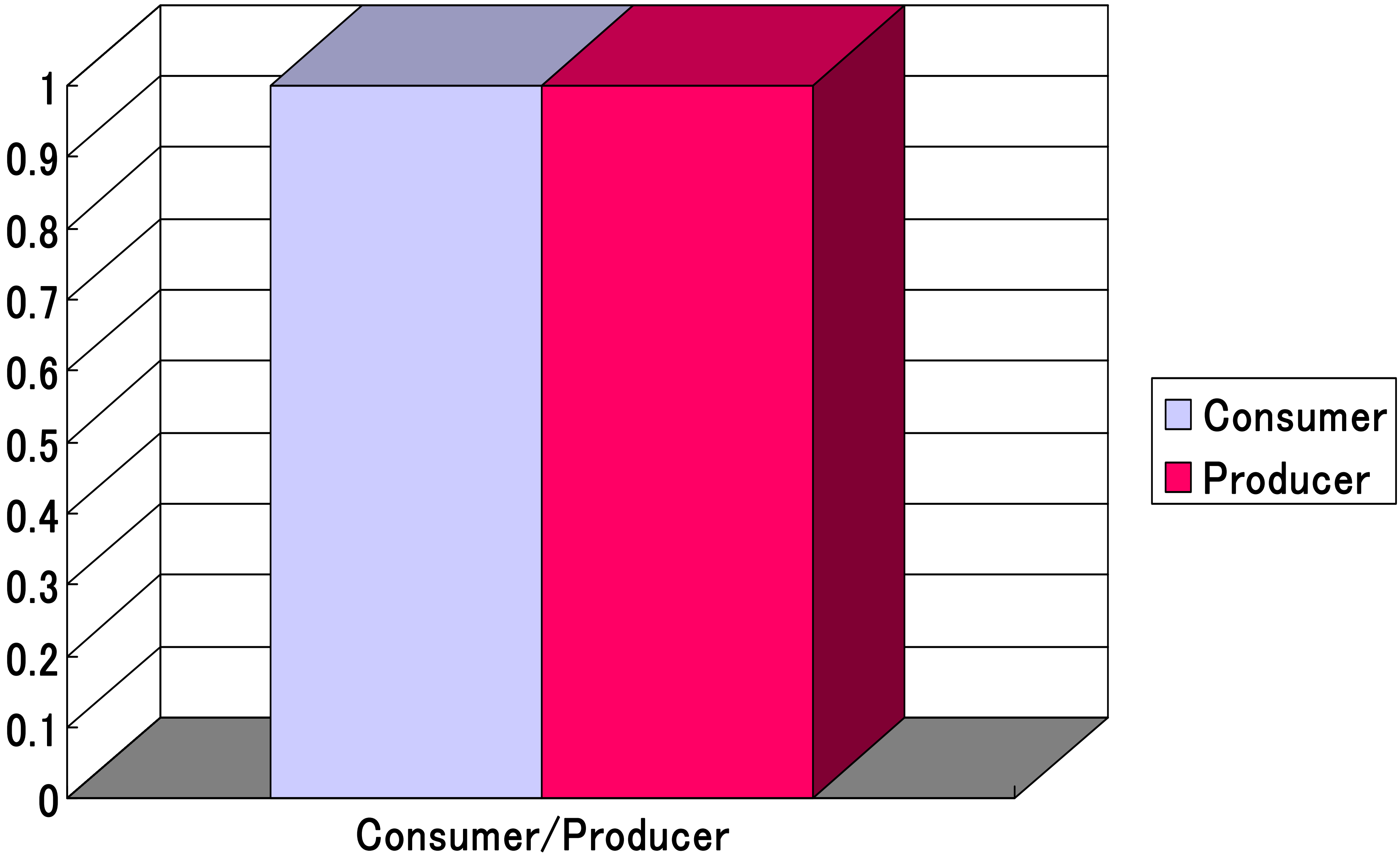
Who are you?

- 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?

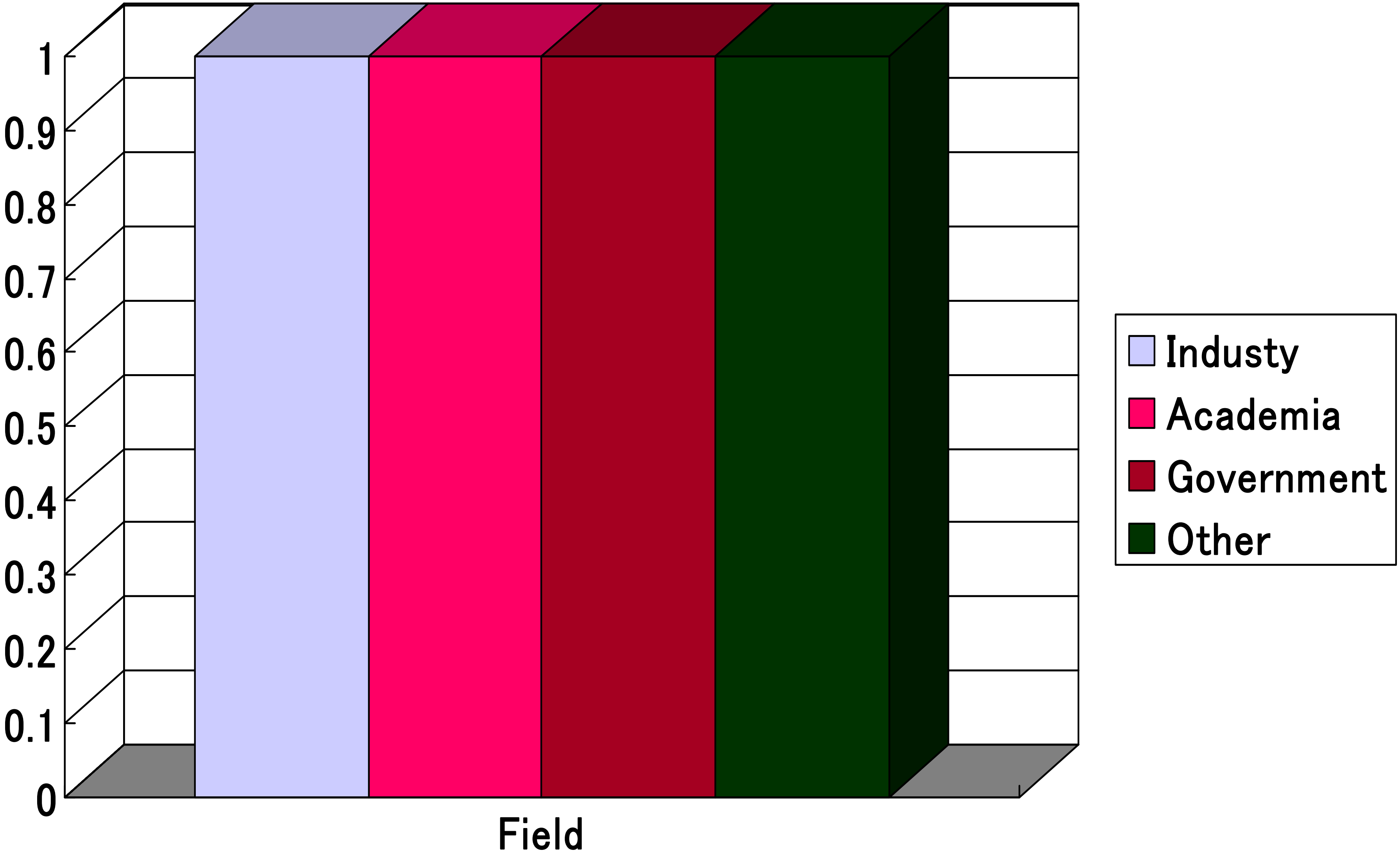
Who are you? (specialization)



Who are you?


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Consumer/Producer
Consumer
Producer

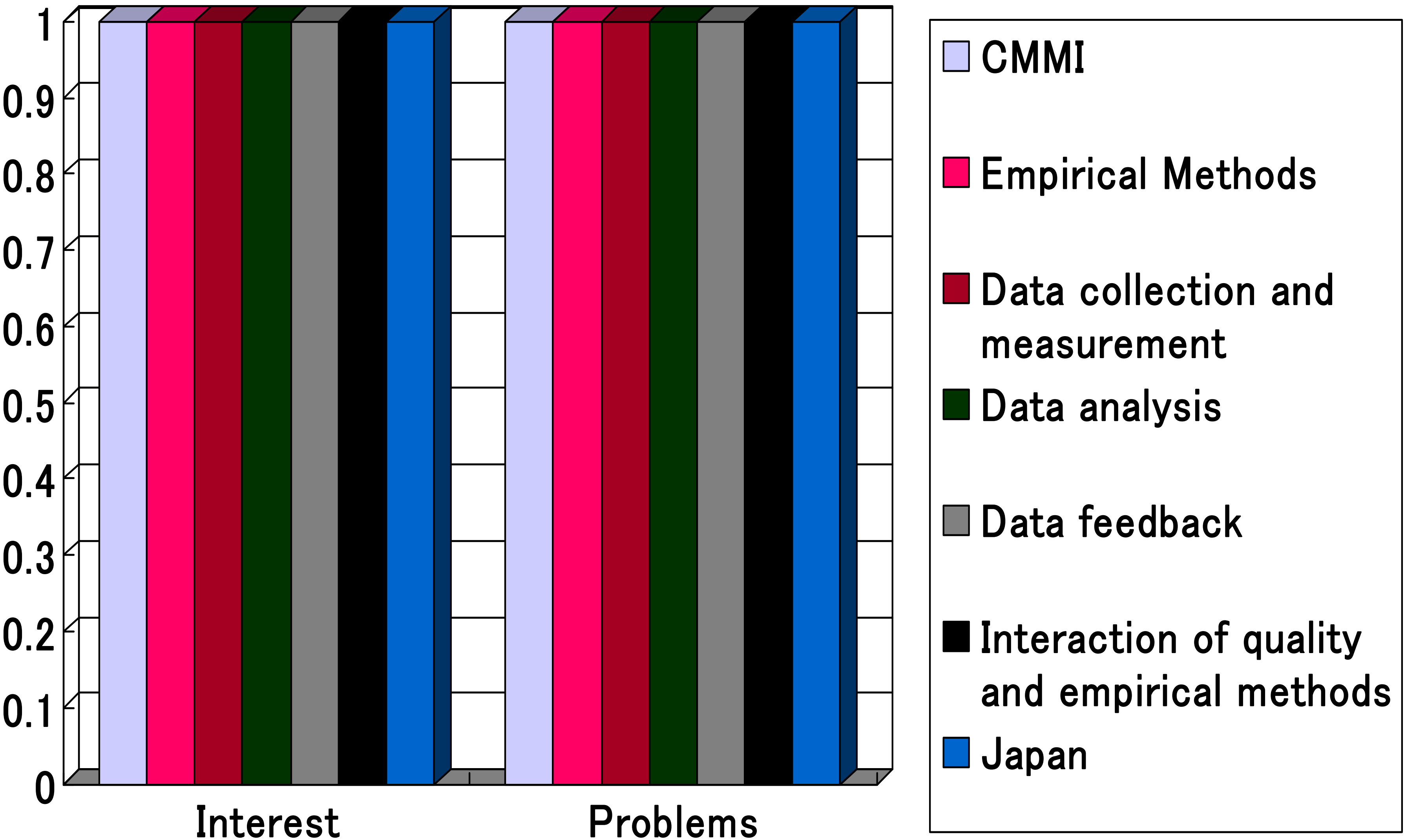
Who are you?


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Field
Industy
Academia
Government
Other

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)



What is EASE?

- Sit back and relax a moment, this is our advertisement ☺

The EASE Project

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/

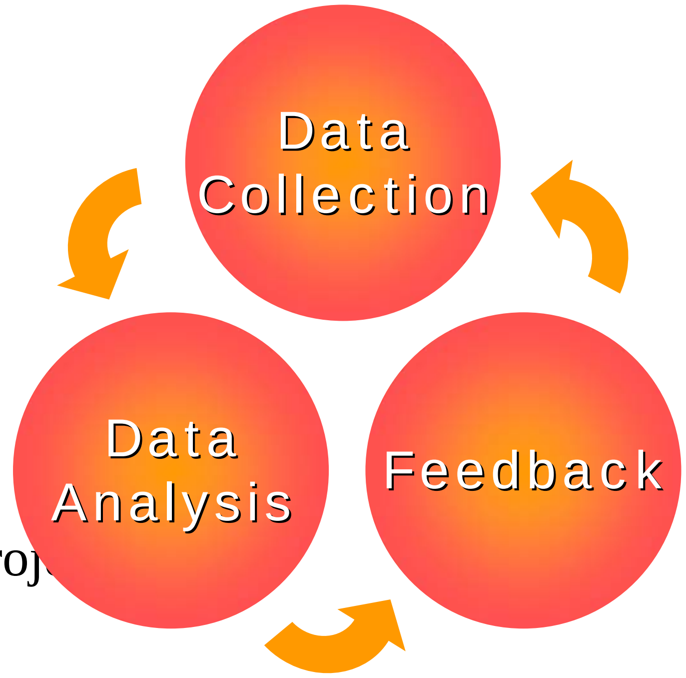
ease
Nara Institute of Science and Technology
EASE Project
Empirical Approach to Software Engineering

The purpose of the EASE project

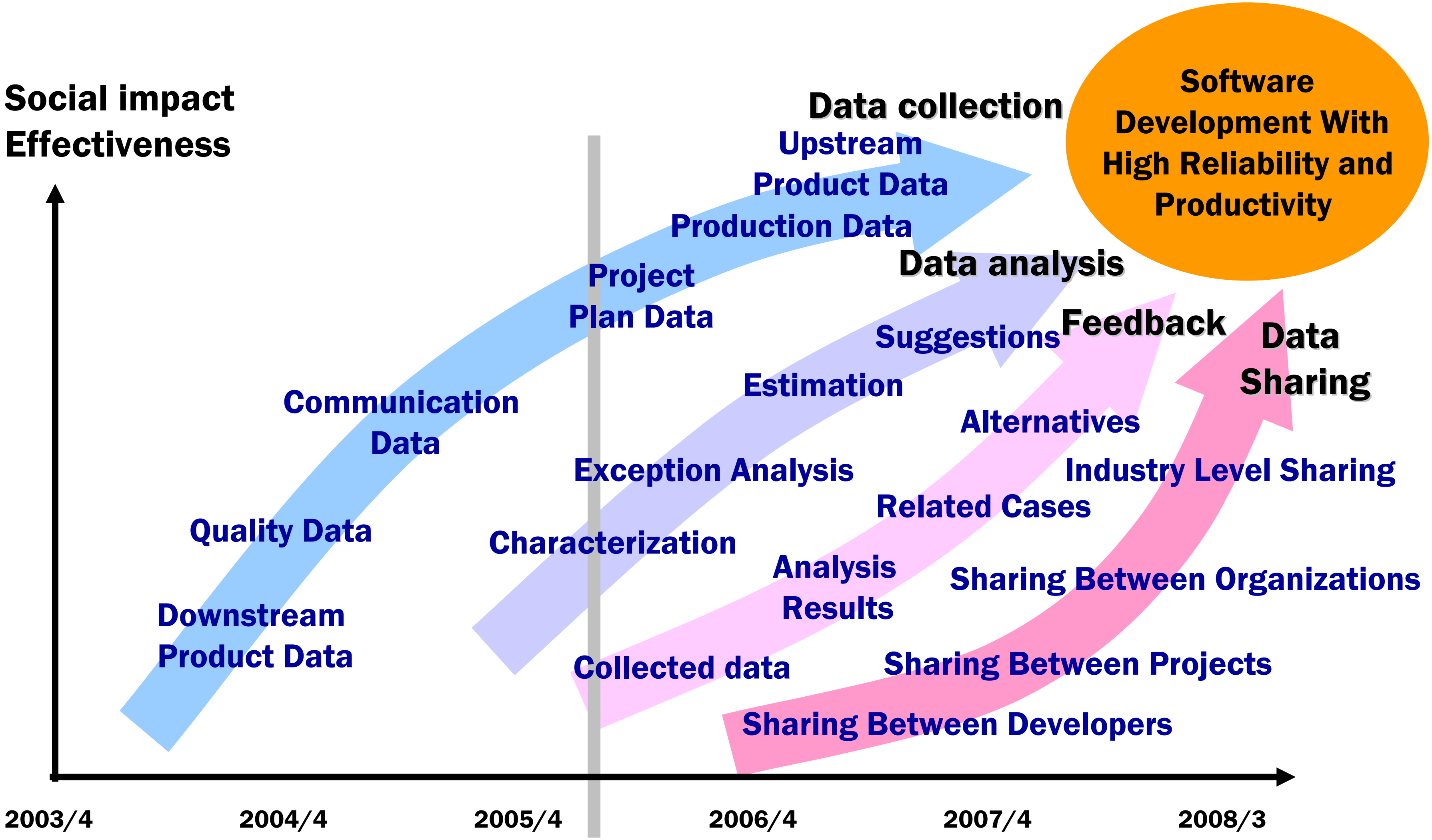
- Achievement of software development technology based on quantitative data
  - Construction of a quantitative data collection system
    - Result 1: Making of EPM open source
  - Construction of a system that supports development based on analyzed data
    - Result 2: EPM application experience
    - 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 proj



The EASE roadmap



EASE and SEC

- 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

SEC Benchmark Database
(aka 1000 projects DB)

CF Selection

Selected Similar Projects

- EPMplus measures
- CF Project Selection
- CF Estimation

	anguage	ev. Typ	Function Points	of Staff	ev. Co t
Project X	Java	New	2000	7	Prediction: 59.99
Project A	Java	New	? (MV)	8	50
Project B	Java	? (MV)	2500	10	70
Project C	? (MV)	intenan	5000	20	250

Current Project EPMplus

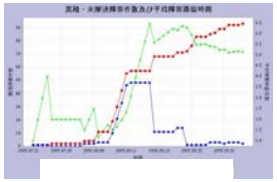
Current Project Estimated by CF

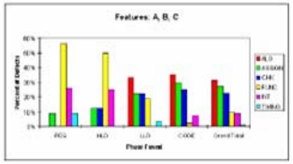
In-process Project Measurement and Feedback

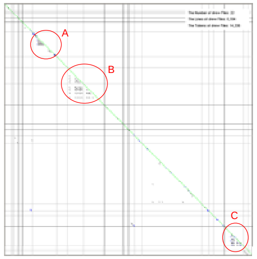
A new research framework for applying empirical software engineering methods in industrial practice and accomplishments in using it.

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

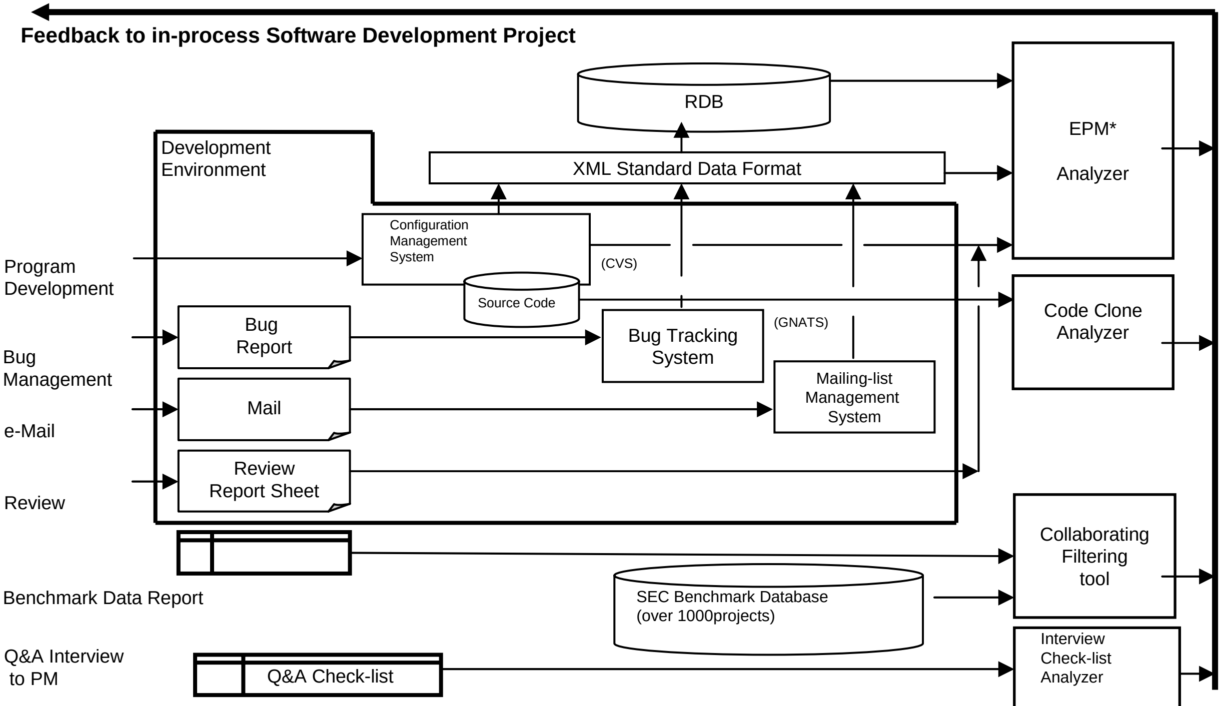
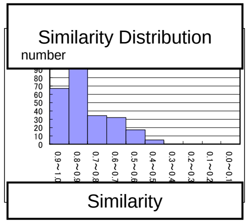
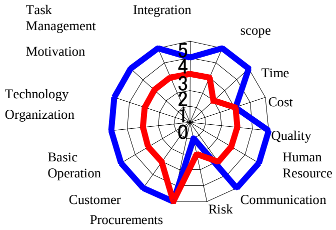
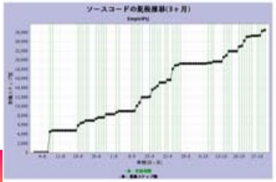
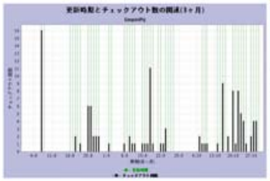


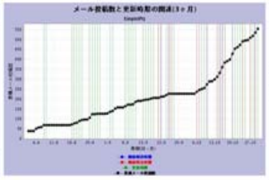
Fig. In-process Project Measurement and Feedback Structure

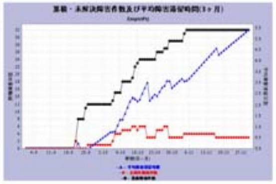












Today's Agenda

Time	*Topic*
9:00-10:00	Introductions
10:15-11:15	Collecting empirical data
11:30-12:30	Analyzing empirical data
12:30-1:30	Lunch
1:30-2:30	Distributing the results
2:45-3:45	Using empirical methods for quality improvement
4:00-5:00	Next steps, summary and conclusions

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

Dawson, Bones, Oates, Brereton, Azuma, and Jackson (2004).

Empirical Methodologies

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

Dawson, Bones, Oates, Brereton, Azuma, and Jackson (2004).

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

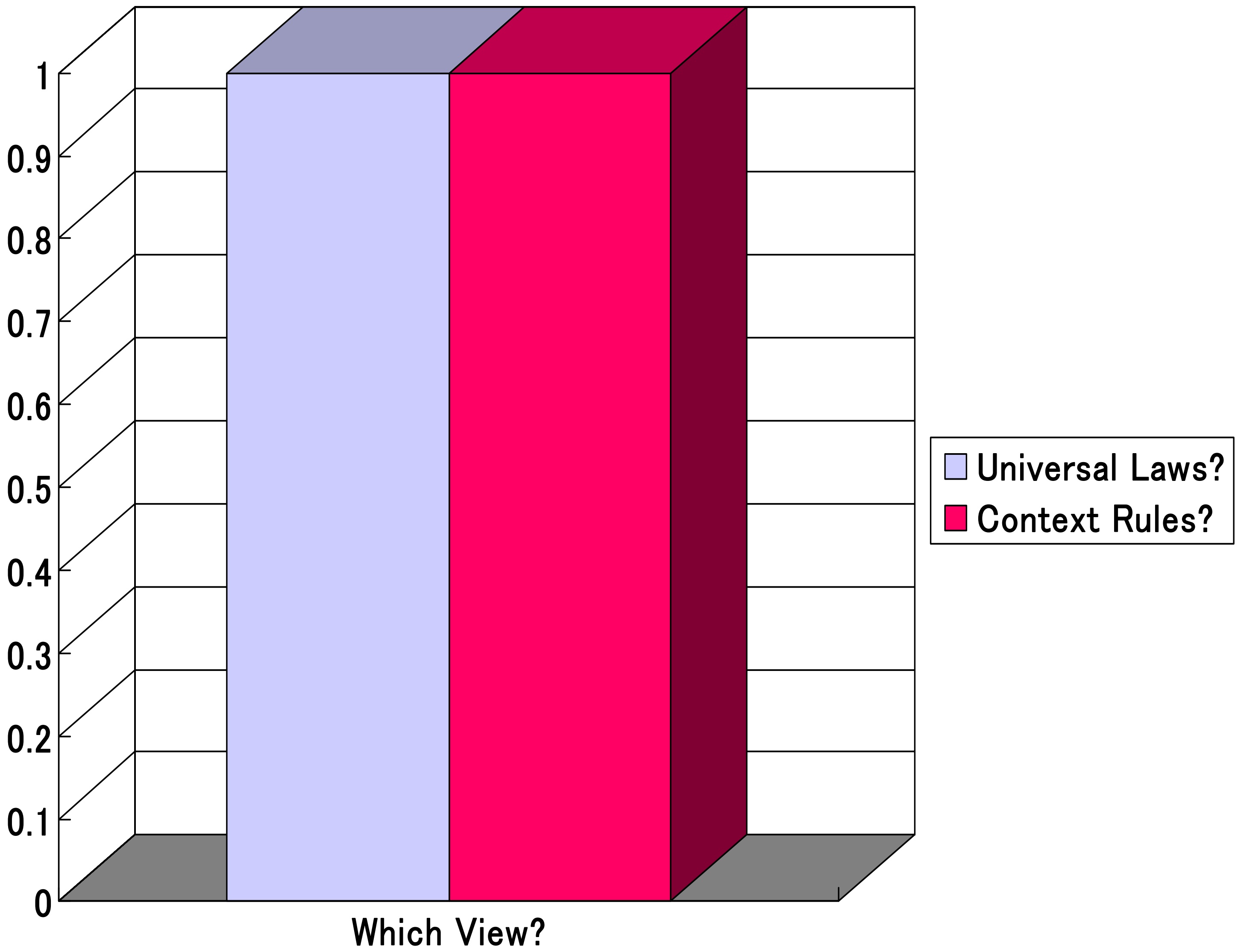
Positivism

- 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

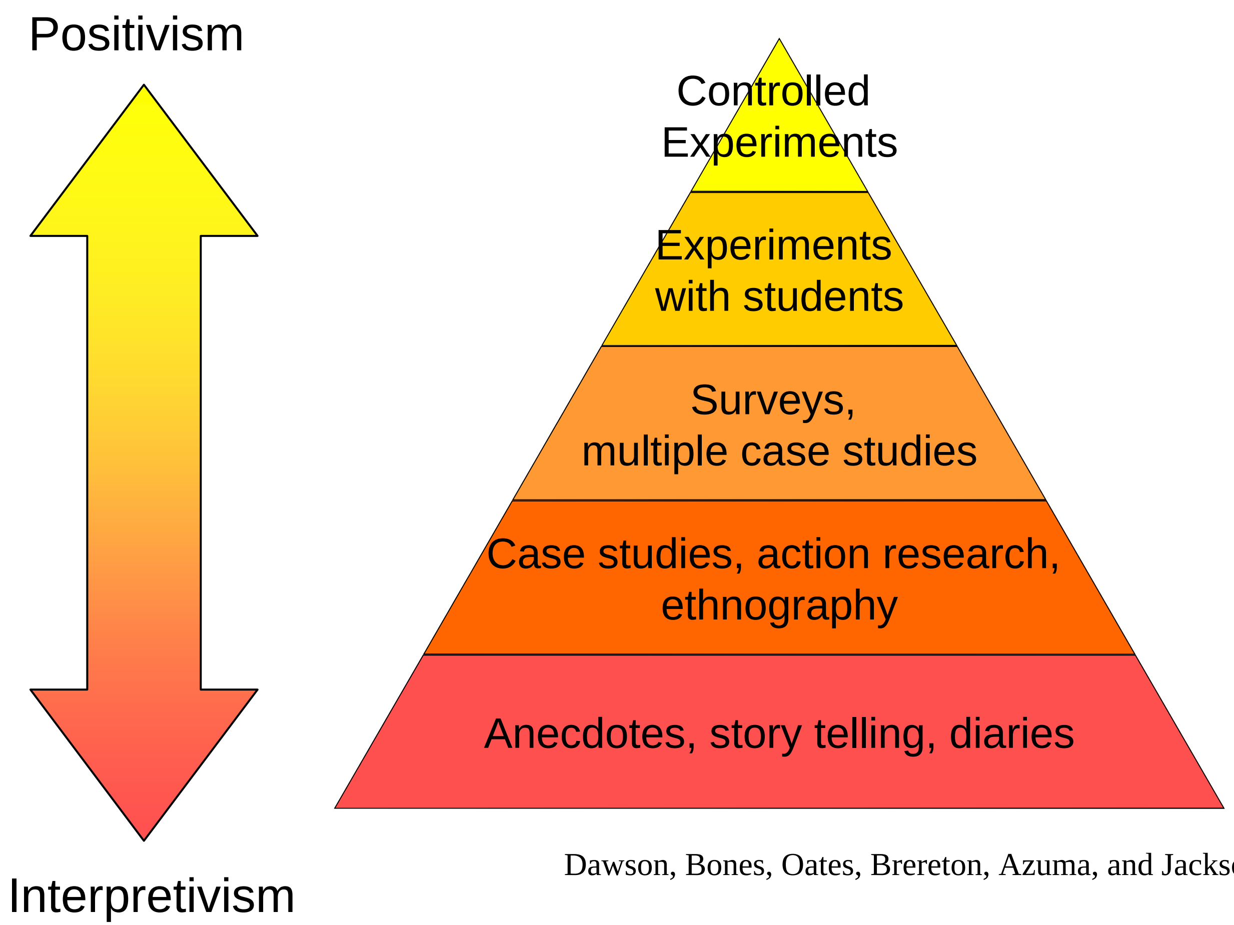
Interpretivism

Dawson, Bones, Oates, Brereton, Azuma, and Jackson (2004).

Instant Poll!



A Pyramid of Empirical Research Types



Dawson, Bones, Oates, Brereton, Azuma, and Jackson (2004).

Framework For Mixed Method Research

		Approach		
		Sequential	Parallel	Independent
Purpose	Triangulation (corroboration)			
	Complementarity (elaboration)			
	Development			
	Initiation			
	Expansion			

Petter and Gallivan. (2004).

A Typology of Research Methods

			Researcher's Assumption About Results	
			Contextual (Kairotic)	Universal (Chronotic)
Topic of Study	Concept (Idea) Study of Concepts		Research Review Conceptual Book Review	Philosophical Study Simulation
	Objects Study of Objects	Unique Objects	History	Case Study Post Hoc Analysis
		Class or Set of Objects	Survey	Experiment Meta-analysis

Counelis (2000).

What Evidence Do We Need?

Aim	Evidence	Studies
Develop understanding of nature of SE practice	Natural data	Field or case study Interviews, think-aloud
Exploring how practice might be improved	Management evidence Practitioner evidence	Quantitative experiments Case studies
Evaluation of effects of introducing improvement into practice	Baseline intervention measure qualitative	Field or case studies

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

Category	Validation method
Observational	Project monitoring
	Case study
	Assertion
	Field study
Historical	Literature search
	Legacy
	Lessons learned
	Static analysis
Controlled	Replicated
	Synthetic
	Dynamic analysis
	Simulation

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

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

Experimental Context

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Kitchenham, Pfleeger, Pickard, Jones, Hoaglin, El Emam, and Rosenberg, (2002).

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.

Punter, Ciolkowski, Freimut, and John. (2003).

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.

Punter, Ciolkowski, Freimut, and John. (2003).

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

Punter, Ciolkowski, Freimut, and John. (2003).

On-Line Surveys: Process

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

Punter, Ciolkowski, Freimut, and John. (2003).

On-Line Surveys: Sampling

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

Punter, Ciolkowski, Freimut, and John. (2003).

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.

Punter, Ciolkowski, Freimut, and John. (2003).

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?

O'Brien, Buckley, and Exton (2005).

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

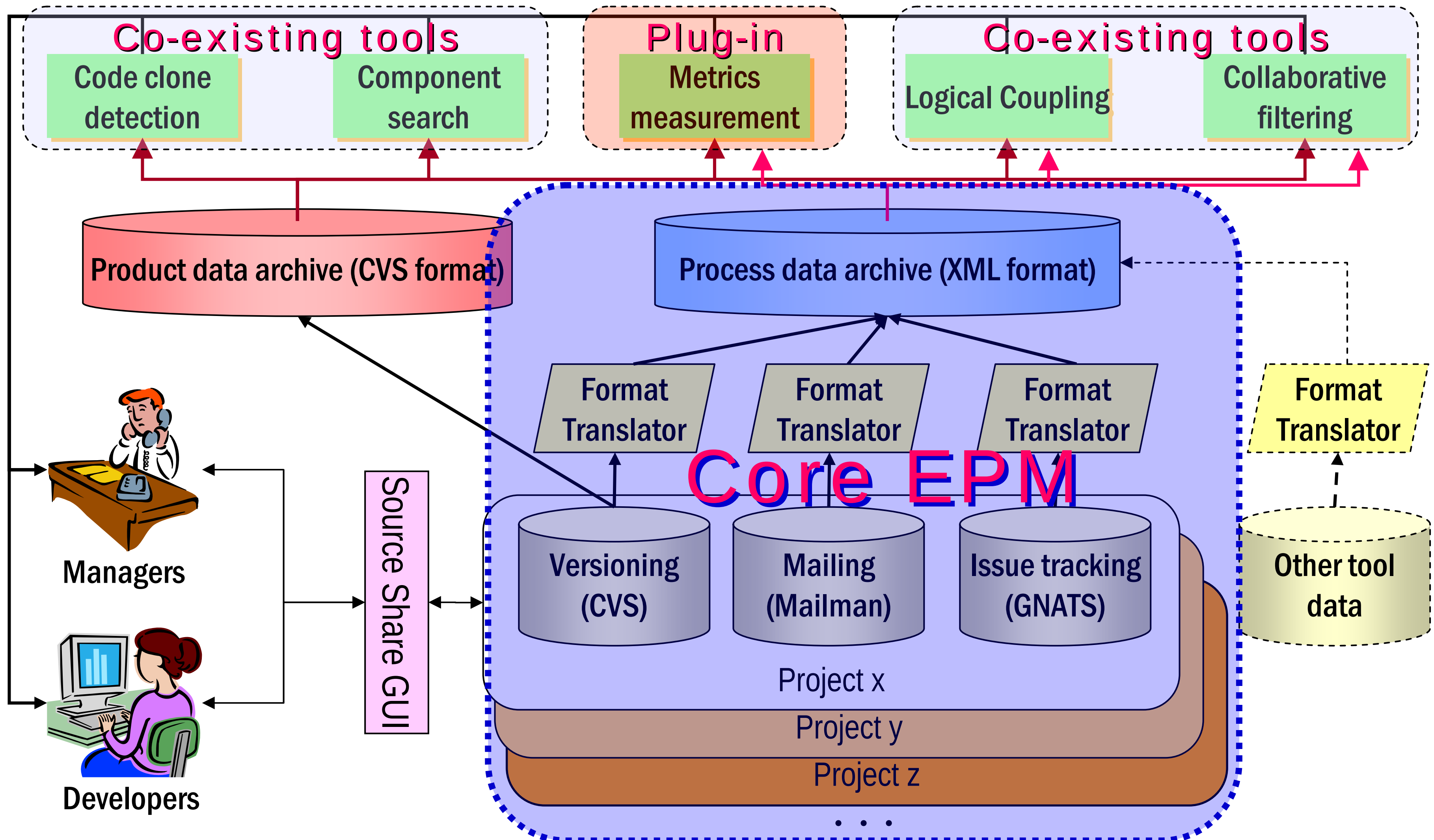
How do you collect data?

- 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

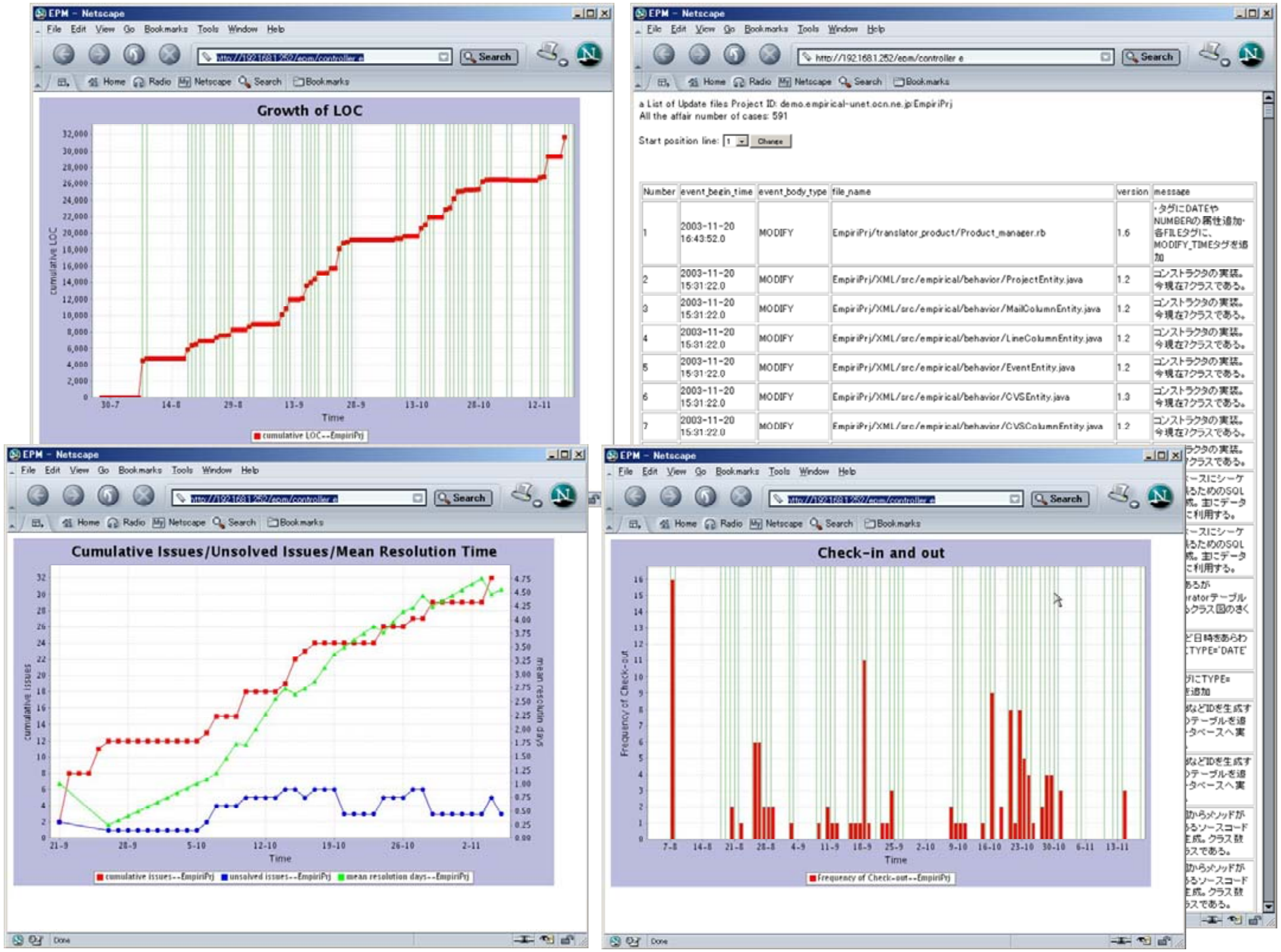
Implementation of the Empirical Project Monitor (EPM)



Automated data collection in EPM

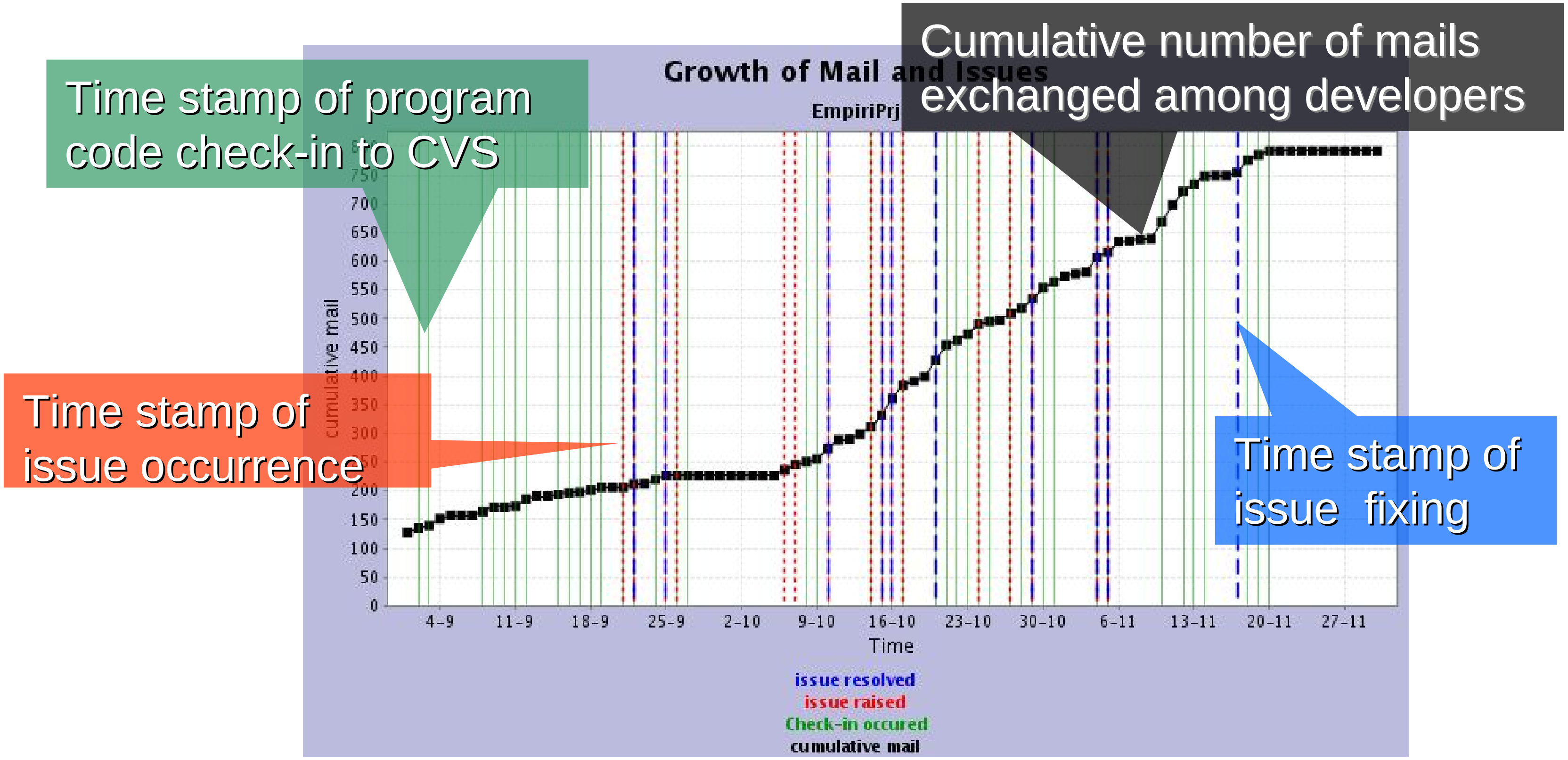
- Reduces the reporting burden on developers
  - without additional work for developers
- Reduces the project information delay
  - data available in real time
- Avoids mistakes and estimation errors
  - uses real (quantitative) data

EPM's GUI


EPM - Netscape
Growth of LOC
cumulative LOC
Time
cumulative LOC--EmpiriPrj
a List of Update files Project ID: demo.empirical-unet.ocn.ne.jp:EmpiriPrj
All the affair number of cases: 591
Start position line:
Number
event_begin_time
event_body_type
file_name
version
message
Cumulative Issues/Unsolved Issues/Mean Resolution Time
cumulative issues
mean resolution days
cumulative issues--EmpiriPrj
unsolved issues--EmpiriPrj
mean resolution days--EmpiriPrj
Check-in and out
Frequency of Check-out
Frequency of Check-out--EmpiriPrj

An example of output

- 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 reason for the change?
- The number of developers making changes to files (ONR)?

Metric

- Change frequency of each file (CVS)
- The changed number of lines in each file (CVS)
- Size of file (CVS)
- Change motive: Bug or specification change (GNATS)
- Number of developers making changes to files (CVS)

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

The GQM Template

What do we want to measure?	
Why do we want to measure it?	
What aspect are we interested in?	
Who wants to know?	
Where are we going to work?	

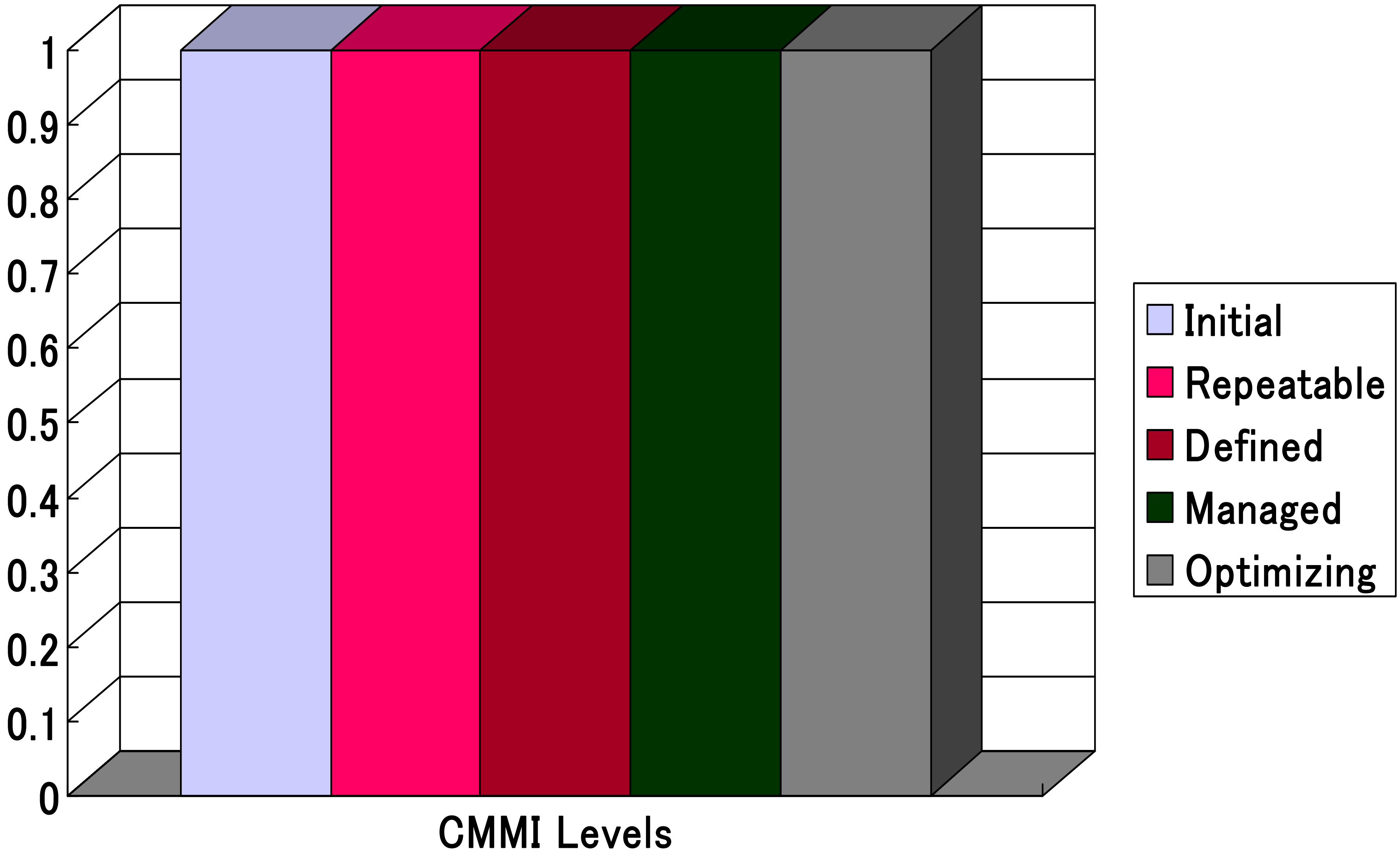
Team Exercise

- 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

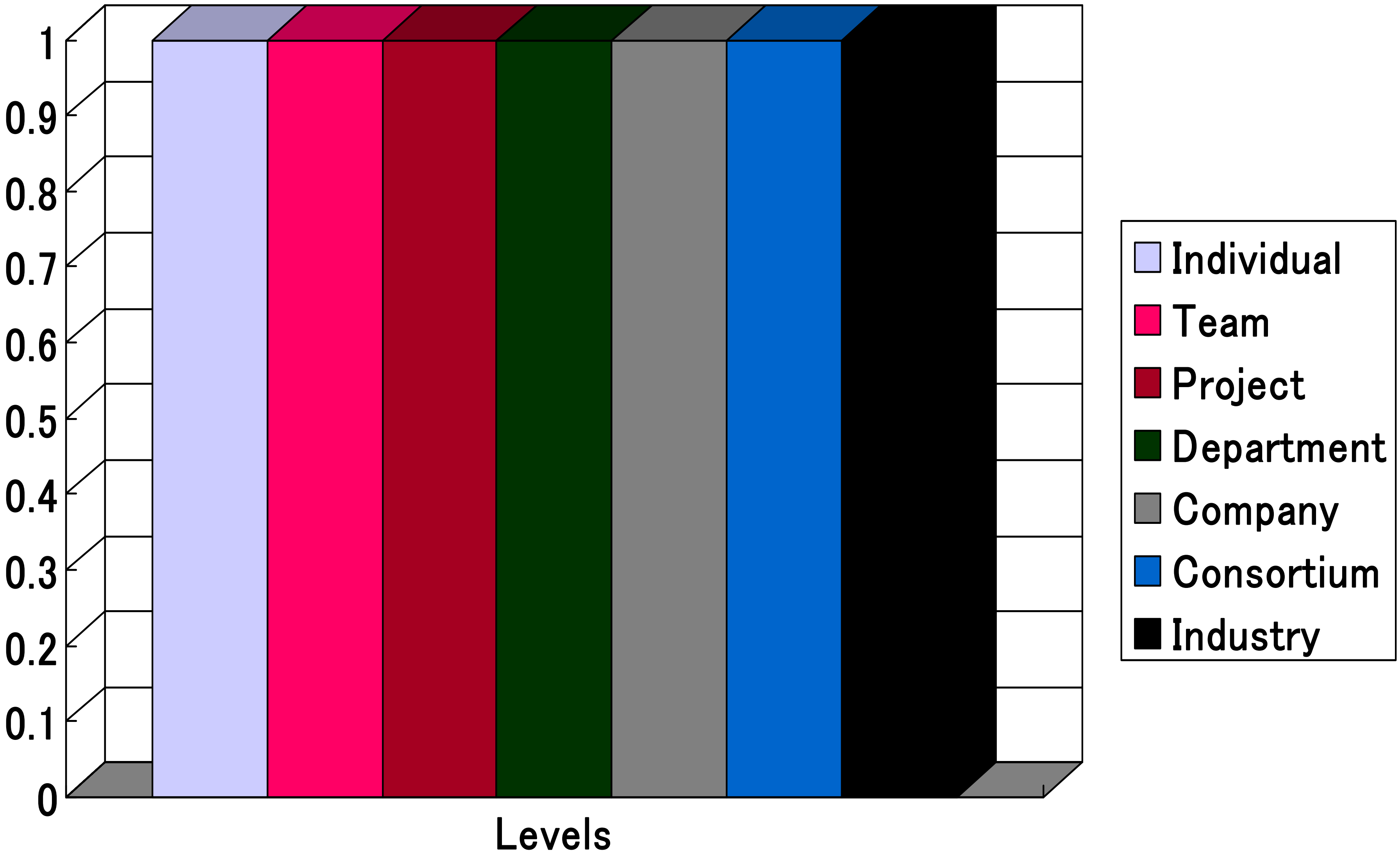
1. Initial: Competent people and heroics
2. Repeatable: basic project management
3. Defined: process standardization
4. Managed: quantitative management
5. Optimizing: continuous process improvement

CMMI Levels



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
CMMI Levels
Initial
Repeatable
Defined
Managed
Optimizing

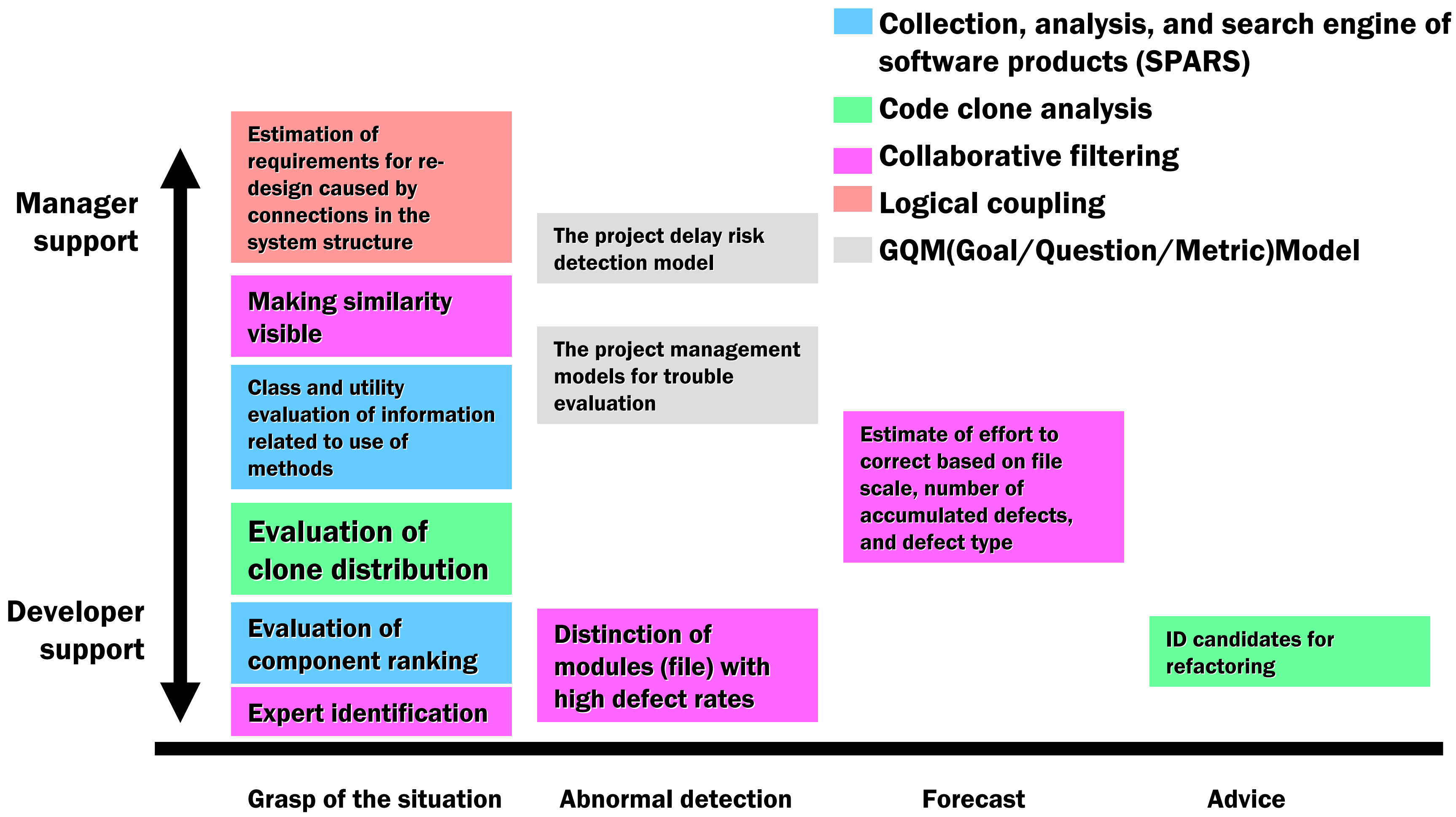
Data Collections



Break!

Analyzing Empirical Data

Analysis targets



Analyzing Empirical Data

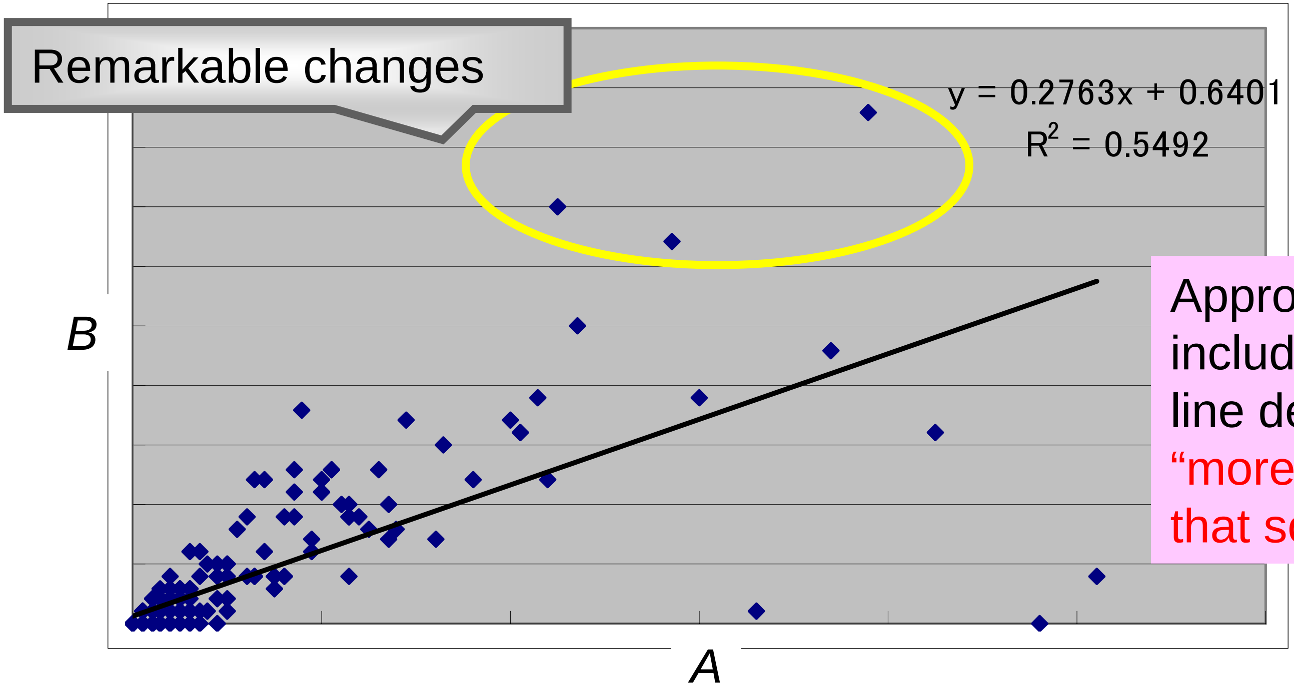
- Analysis of potential project delay risks
- Logical coupling
- Defect correction analysis
- Collaborative filtering
- Extracting patterns through association rule mining
- Analysis in your hands?

Risk Detection Preventing Project Delay

- 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



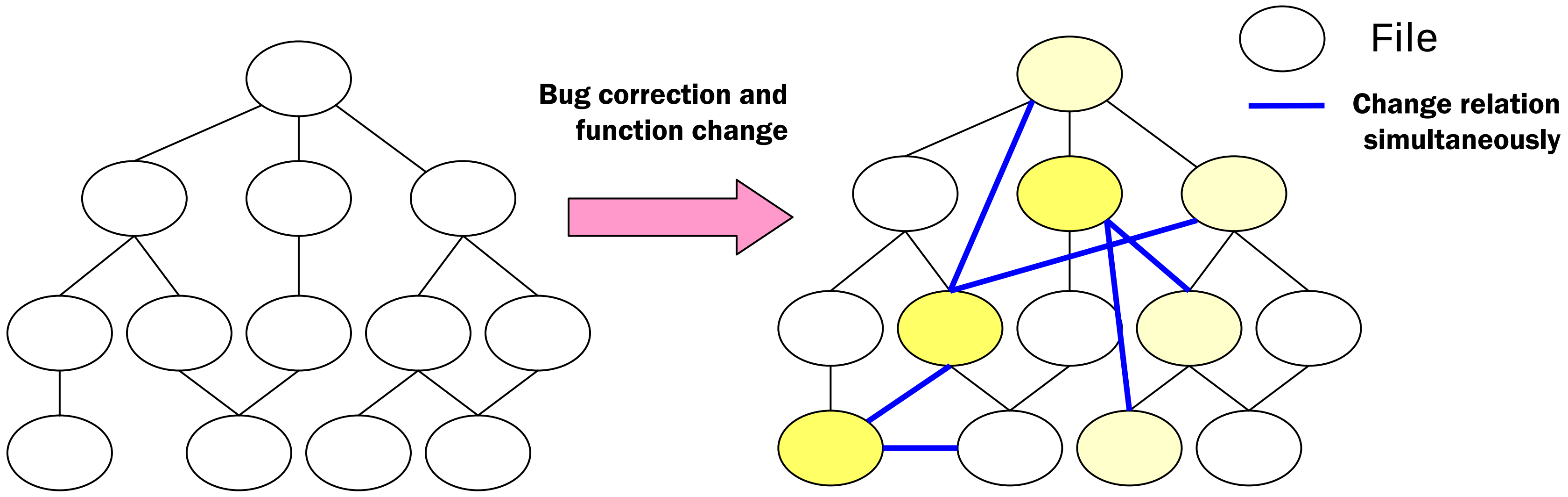
Approximately 30% of changes include a certain amount of line deletion normally ->
"more than 30%" implies that some problems happened

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

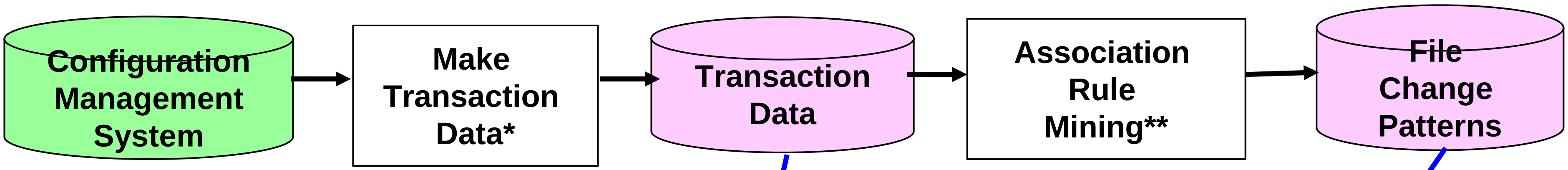
Complication analysis by logical coupling

- 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.



Logical Coupling Analysis(1)

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



Trans. No.	Date of Update	Author	Number of Files	File1	File2	File3	File4	File5	File6	…
1	2000/2/26 9:28:42	tomoko	21	FileA	FileB	FileC	FileD	FileE	FileF	
2	2000/2/27 9:42:12	tomoko	263	FileG	FileH	FileB	FileI	FileJ	FileK	
3	2000/2/27 11:03:35	noriko	4	FileK						
4	2000/2/28 1:30:28	kohei	5	FileF						
5	2000/2/28 1:36:06	noriko	2	FileP						
6	2000/2/28 1:37:14	tomoko	2	FileF						
7	2000/2/28 1:38:07	kohei	1	FileT						
…										

No. of Patterns	Count of changed together	Support	Number of files	File 1	File 2	File 3	File 4
1	9	0.00438	4	FileL	FileM	FileN	FileP
2	9	0.00438	4	FileF	FileG	FileH	FileI
3	9	0.00438	4	FileC	FileJ	FileE	FileK
4	9	0.00438	4	FileC	FileD	FileE	FileK
5	9	0.00438	4	FileA	FileB	FileC	FileK
6	10	0.00486	3	FileN	FileO	FileP	
7	9	0.00438	3	FileM	FileN	FileP	
…….							

*Thomas Zimmermann, Peter Weißgerber: "***Preprocessing CVS Data for Fine-Grained Analysis***", Proc. International Workshop on Mining Software Repositories (MSR), Edinburgh, Scotland, UK, May 2004

Apriori Algorithm：R. Agrawal, R. Srikant: "*Fast algorithm for mining association rules***", Proc. 20th Very Large Data Bases Conference(VLDB), pp.487-499. Morgan Kaufmann, 1994

Logical Coupling Analysis(2)

- Analyze each pattern by confidence
  - Association Rule：X (Antecedent) ⇒ Y (Consequent)
  - Confidence (X⇒Y) ： # of Update (X∪Y) / # of Update (X)

Confidence	# of Update (Antecedent)	# of Update (Consequent)	Antecedent		Consequent
0.8	5	4	FileA.cpp FileB.h FileC.cpp	->	FileD.h
1	4	4	FileA.cpp FileB.h FileD.h	->	FileC.cpp
1	4	4	FileA.cpp FileC.cpp FileD.h	->	FileB.h
1	4	4	FileB.h FileC.cpp FileD.h	->	FileA.cpp

Has “FileD.h” missed a change?

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

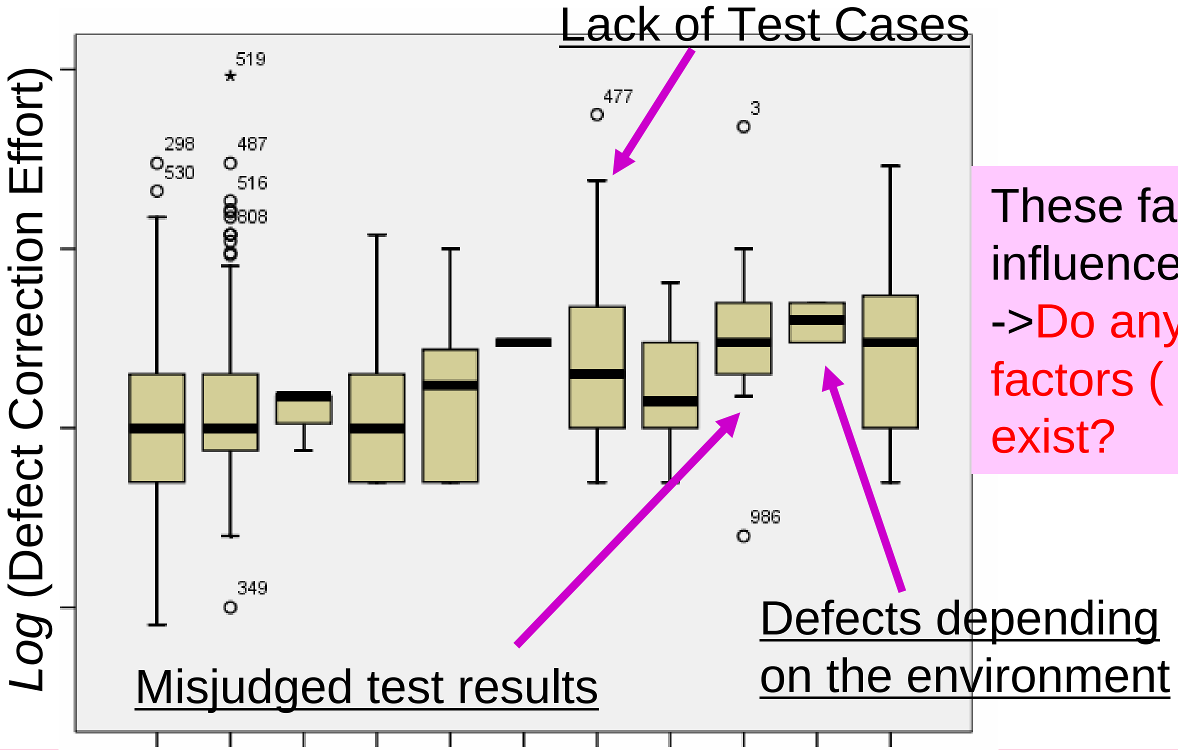
Confidence	# of Update (Antecedent)	# of Update (Consequent)	Antecedent		Consequent
0.5	8	4	FileX.cpp	->	FileY.cpp
1	4	4	FileY.cpp	->	FileX.cpp

Factor Analysis of Defect Correction Effort

- Improve software process taking into account return on investment

Factor Analysis of Defect Correction Effort(1)

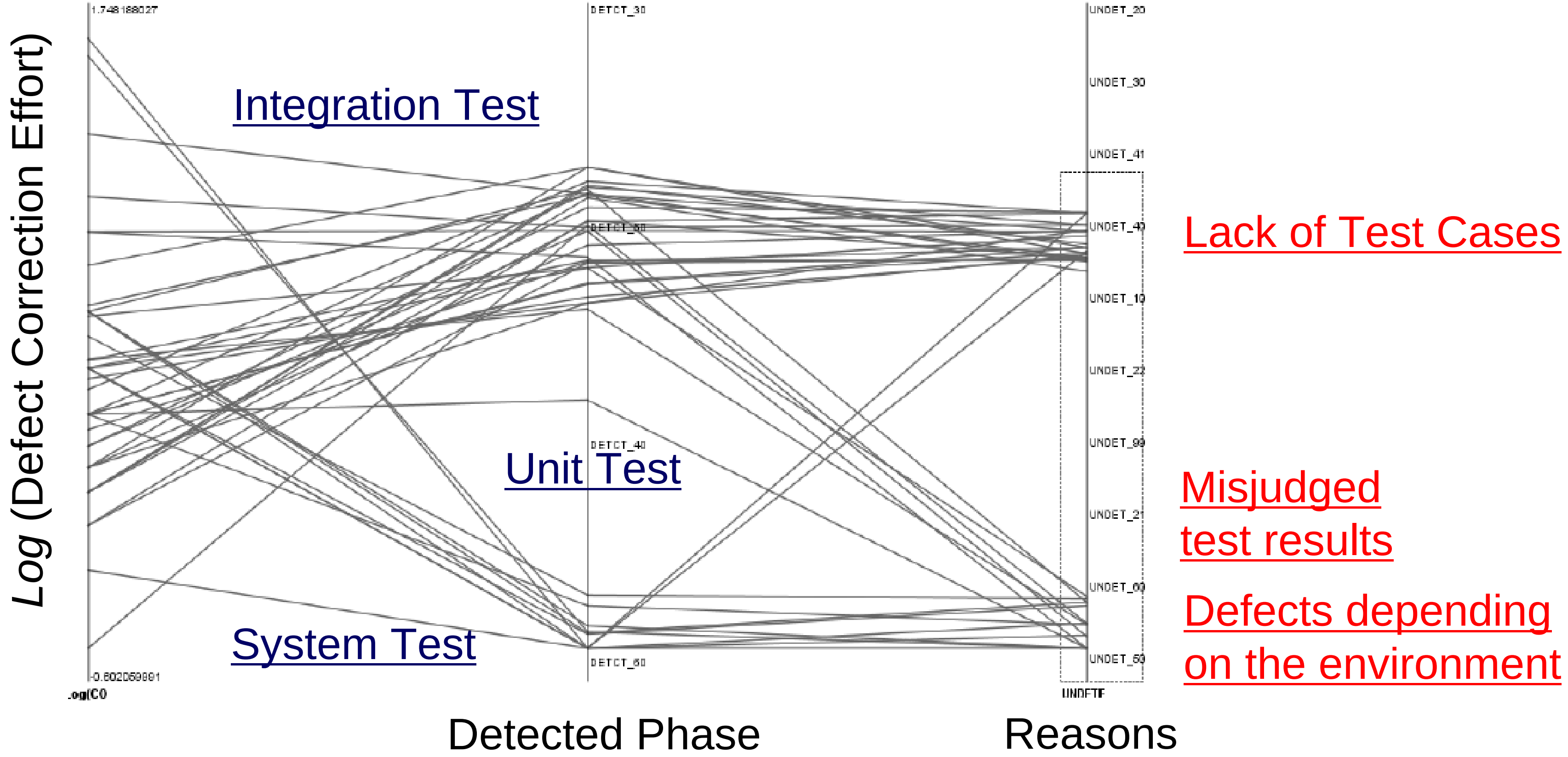
- E.g.) Defect Correction Effort classified by “Reason why the defect was not detected in the preferable phase”



These factors seem to influence the correction effort ->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



Estimation Using Collaborative Filtering: Robust for Missing Values

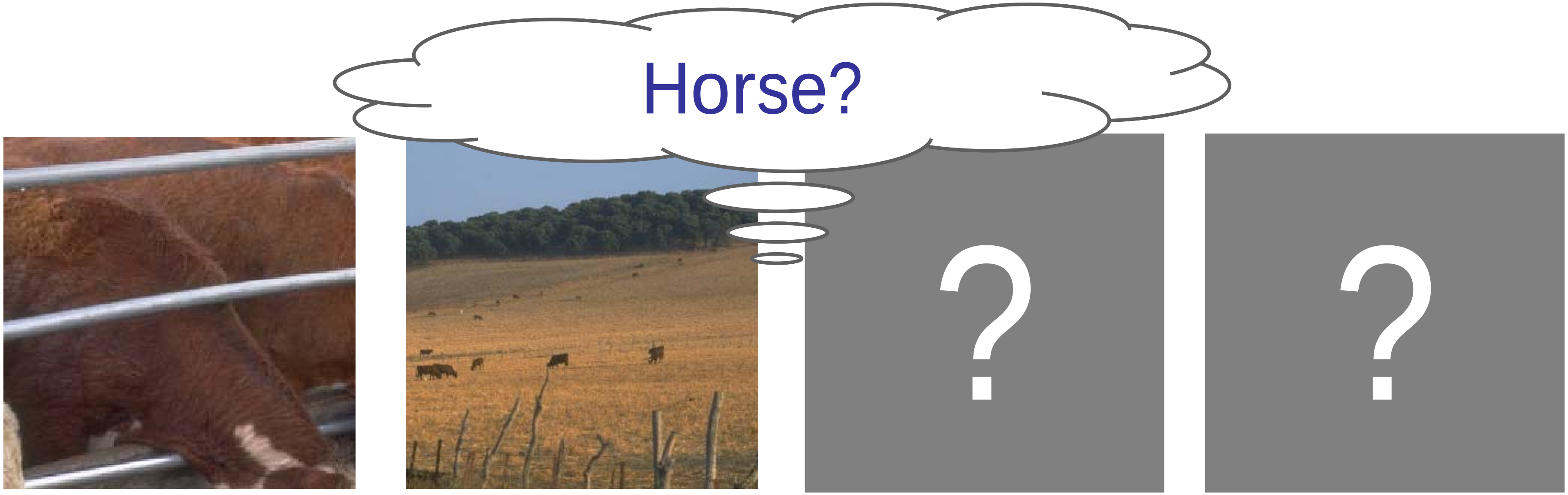
Collaborative Filtering

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

	Focused	Represen-tative	Q & M Resources	Collaborative	Outcome Adopted
App. A	9	9	9	7	7.5 (target)
App. B	8	7	8	? (missing)	8
App. C	? (missing)	8	8	8	7
App. D	7	6	? (missing)	9	6

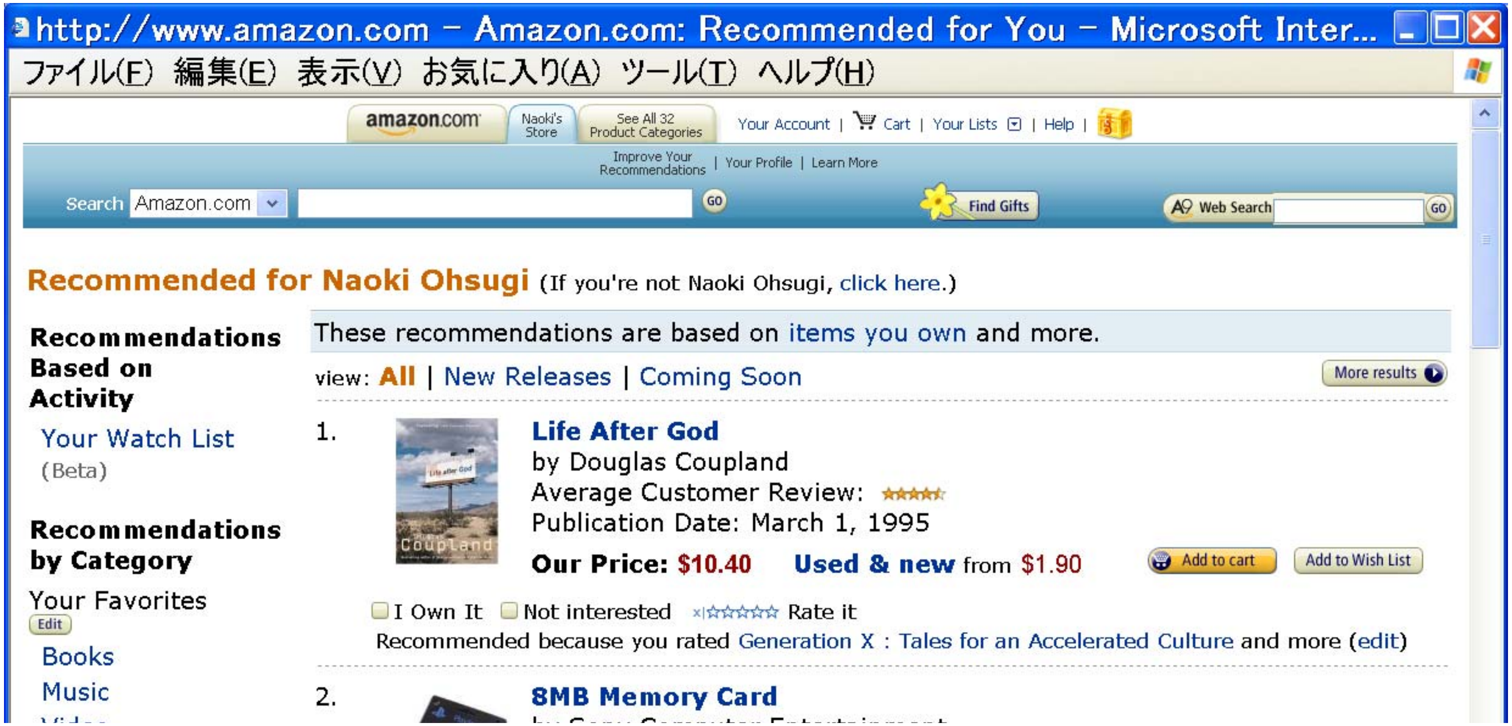
Background

- For development planning, practitioners can use estimation methods with historical projects' data, but…
  - Historical data usually contain many *Missing Values*.
    - Briand, L., Basili, V., and Thomas, W. "A Pattern Recognition Approach for Software Engineering Data Analysis," IEEE Trans. on Software Eng., vol.18, no.11, pp.931-942 (1992)
  - Missing values reduce the accuracy of estimation.
    - Kromrey, J., and Hines, C., "Nonrandomly Missing Data in Multiple Regression: An Empirical Comparison of Common Missing-Data Treatments," Educational and Psychological Measurement, vo.54, no.3, pp.573-593 (1994)



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 book5 using the similar customers' rating on book 5.

	Book 1	Book 2	Book 3	Book 4	Book 5
Customer X	1 (hate)	2 (unlike)	4 (like)	5 (love)	**Prediction: 4.51**
Similarity: +1.00 er A	1 (hate)	2 (unlike)	**?** (unread)	5 (love)	5 (love)
Similarity: +0.97 er B	2 (unlike)	**?** (unread)	4 (like)	5 (love)	4 (like)
Similarity: -0.97 er C	**?** (unread)	4 (like)	2 (unlike)	1 (hate)	1 (hate)

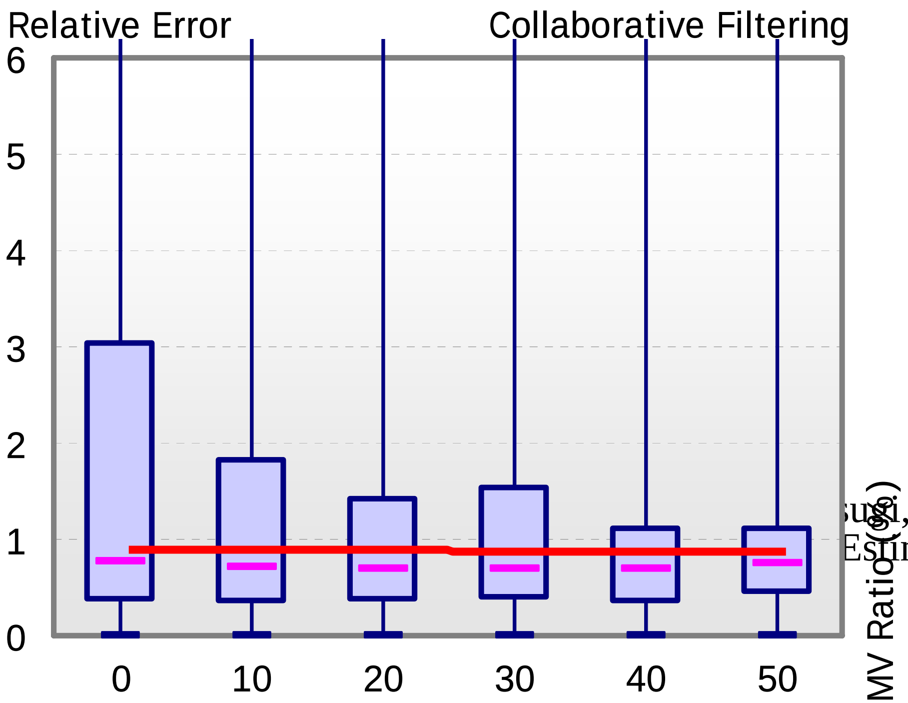
Estimation of Project X's Development Cost (required man-month)

- 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.

	Language	Dev. Type	Function Points	# of Staffs	Dev. Cost
Project X	Java	New	2000	7	Prediction: 59.99
Similarity: +1.00 t A	Java	New	? (MV)	8	50
Similarity: +1.00 t B	Java	? (MV)	2500	10	70
Similarity: 0.00 t C	? (MV)	Maintenance	5000	20	250

Experimental Evaluation of Estimation Accuracy [1]

- 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

Association rule mining
- Motivation -

- 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.

Input: project dataset

- Project dataset (project descriptive data) has
  - rows correspond to individual project
  - columns correspond to data field

ID	Development Type	Organization Type	Resource Level	Effort Specify Ratio	Effort Coding Ratio	Effort Test Ratio	•••
0001	New Development	Banking	A	80	230	200	•••
0002	Enhancement	Construction	B	120	200	360	•••
0003	Enhancement	Public Administration	B	60	260	400	•••
•••	•••	•••	•••	•••	•••	•••	•••

Output: extracted rules

- 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

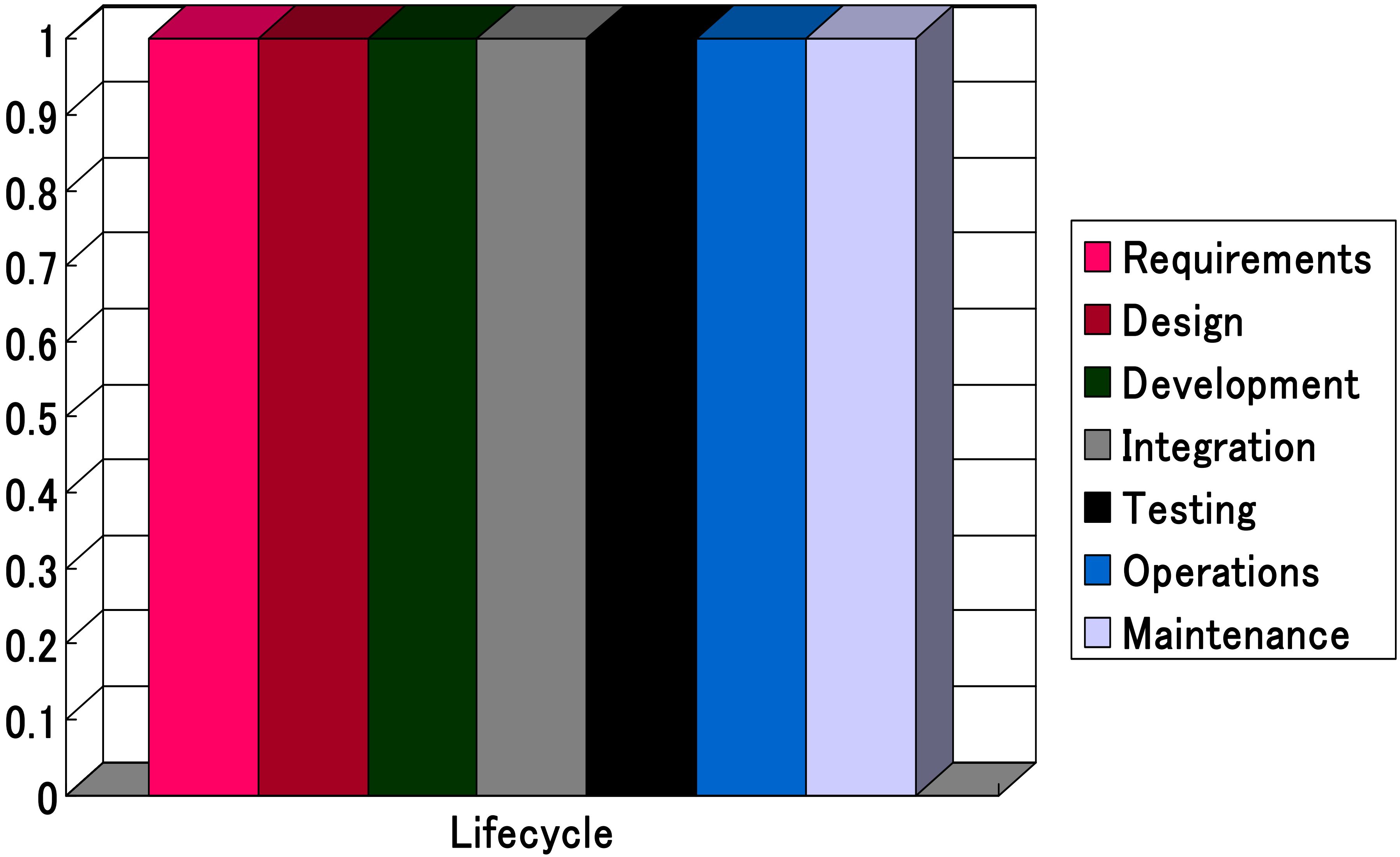
Analysis in your hands

- 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

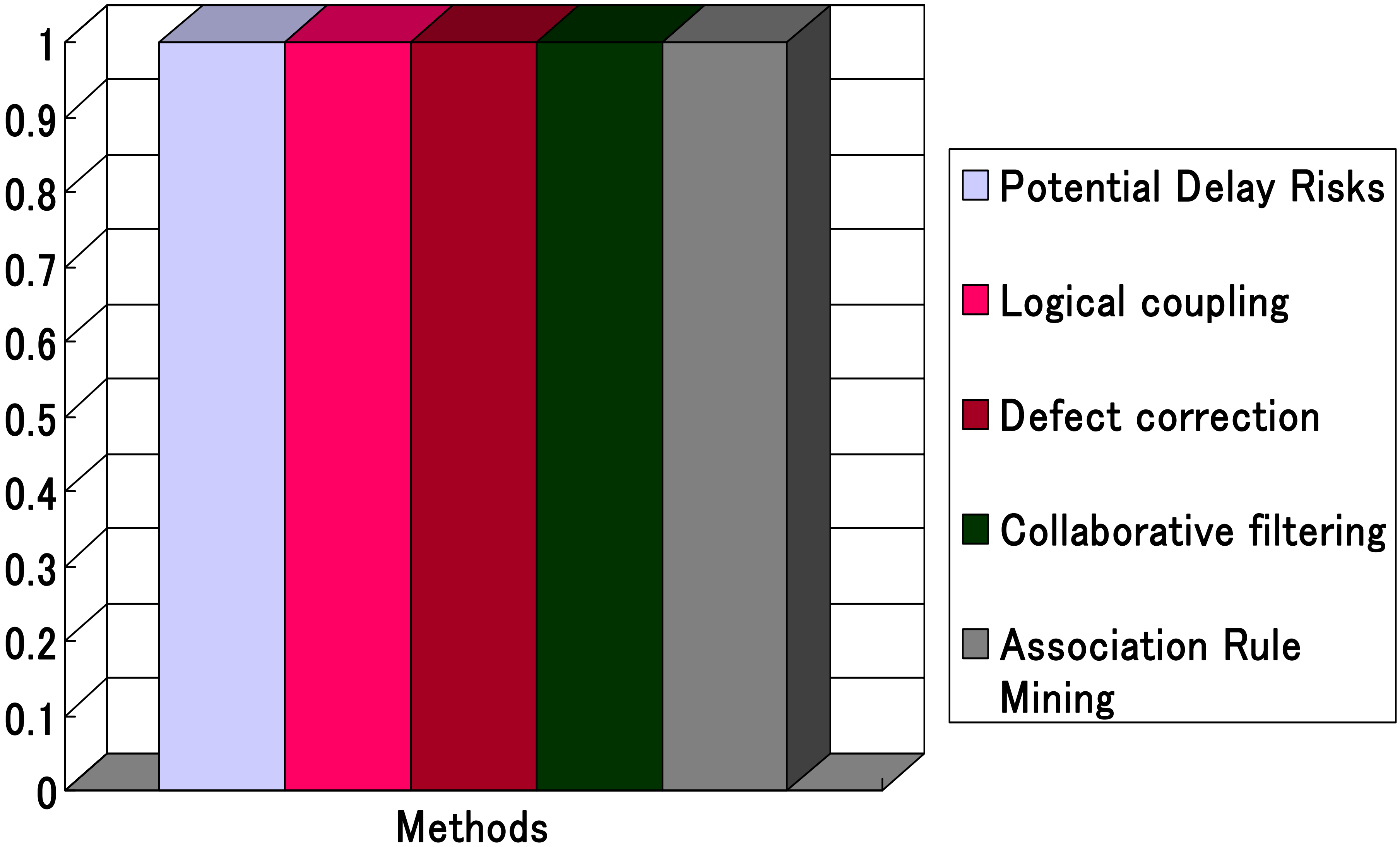
Which area most needs analysis?



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?



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

SIMSE

- 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).

SIMSE

- 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).

SIMSE

- 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

Navarro, Baker, and van der Hoek (2004).

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).

Simulation Based Project Management Training

- 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

Software Design Laboratory
established in 2005

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.

One year after

New project

The same kind of the mistake was made again.

By a new developer

The knowledge was not transferred.

-Short development periods
- High fluidity of developers

Goal and Approach

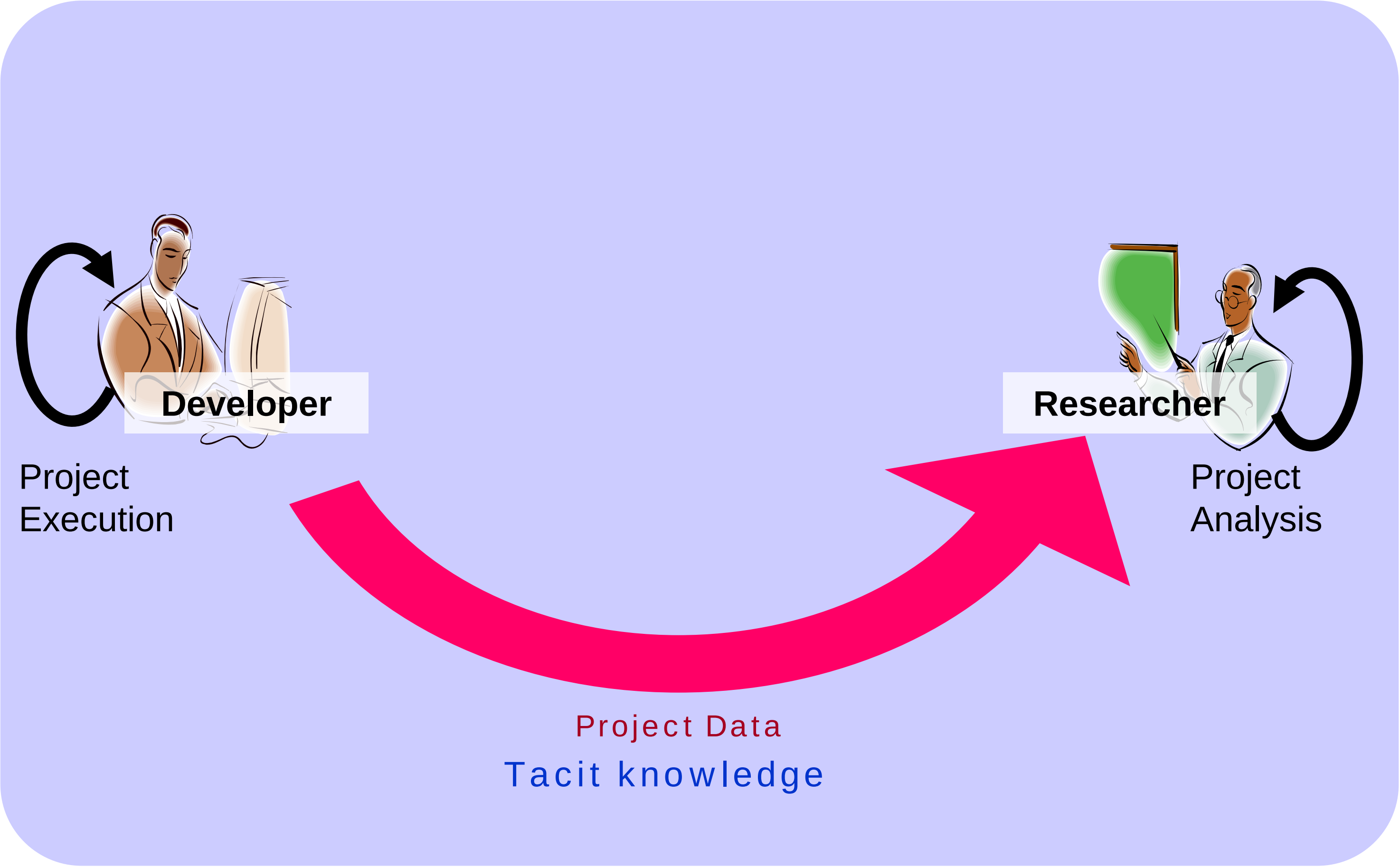
- Goal
  - Establishing knowledge feedback cycle from past projects to new projects.
- Approach
  - We propose a Knowledge Feedback Cycle (KFC) framework.
  - The following 3 tools play very important roles in the KFC framework.
    - Empirical Project Monitor (EPM) : automatically collects software 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

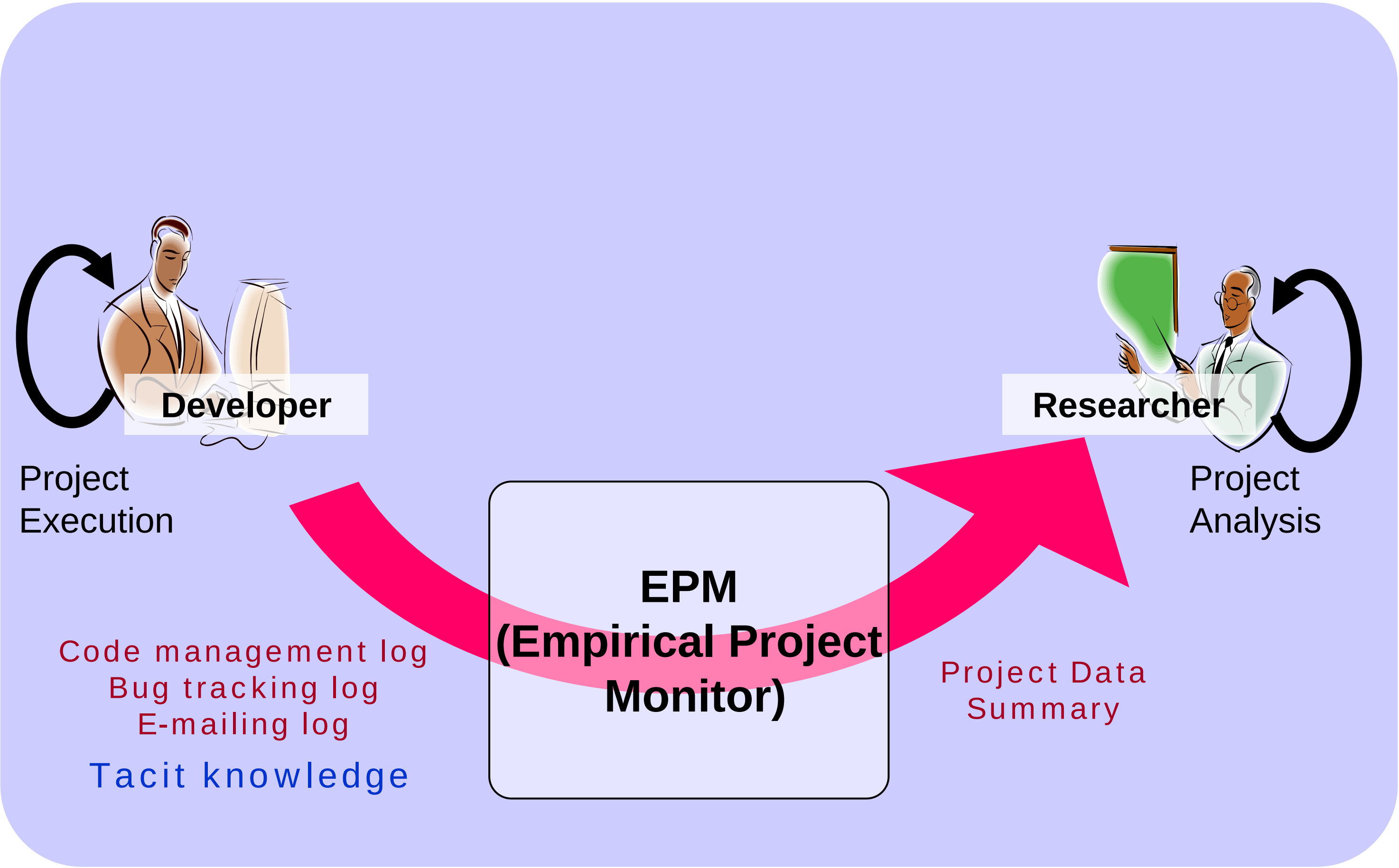
- Knowledge Feedback Cycle (KFC)
- Project Replayer (one of 3 key tools in the KFC)
    - Features of Project Replayer
    - Preliminary experiment
    - Results and discussion
- Conclusions and Future Work

Knowledge Feedback Cycle (KFC) Overview

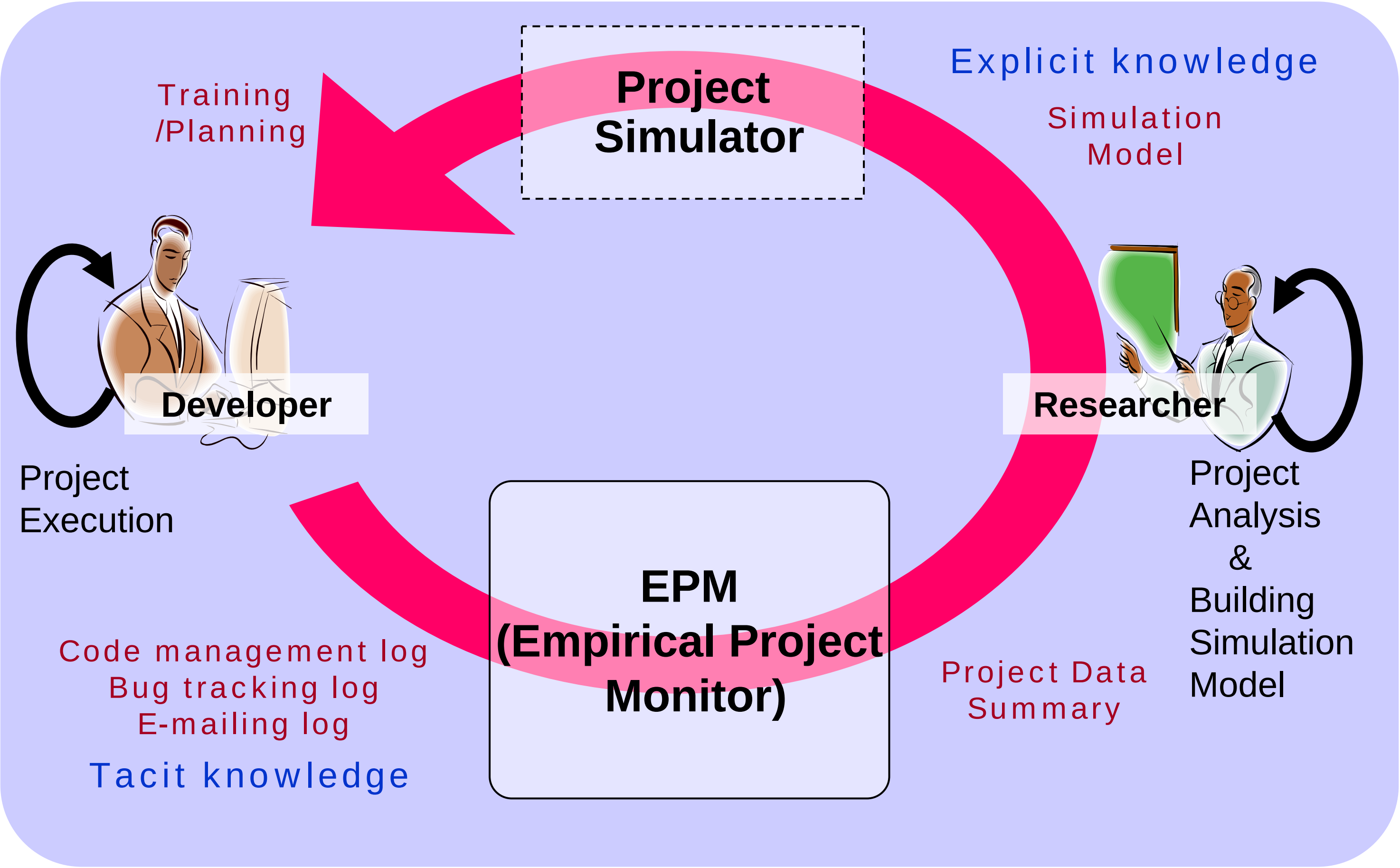


NAIST ease EASE PROJECT

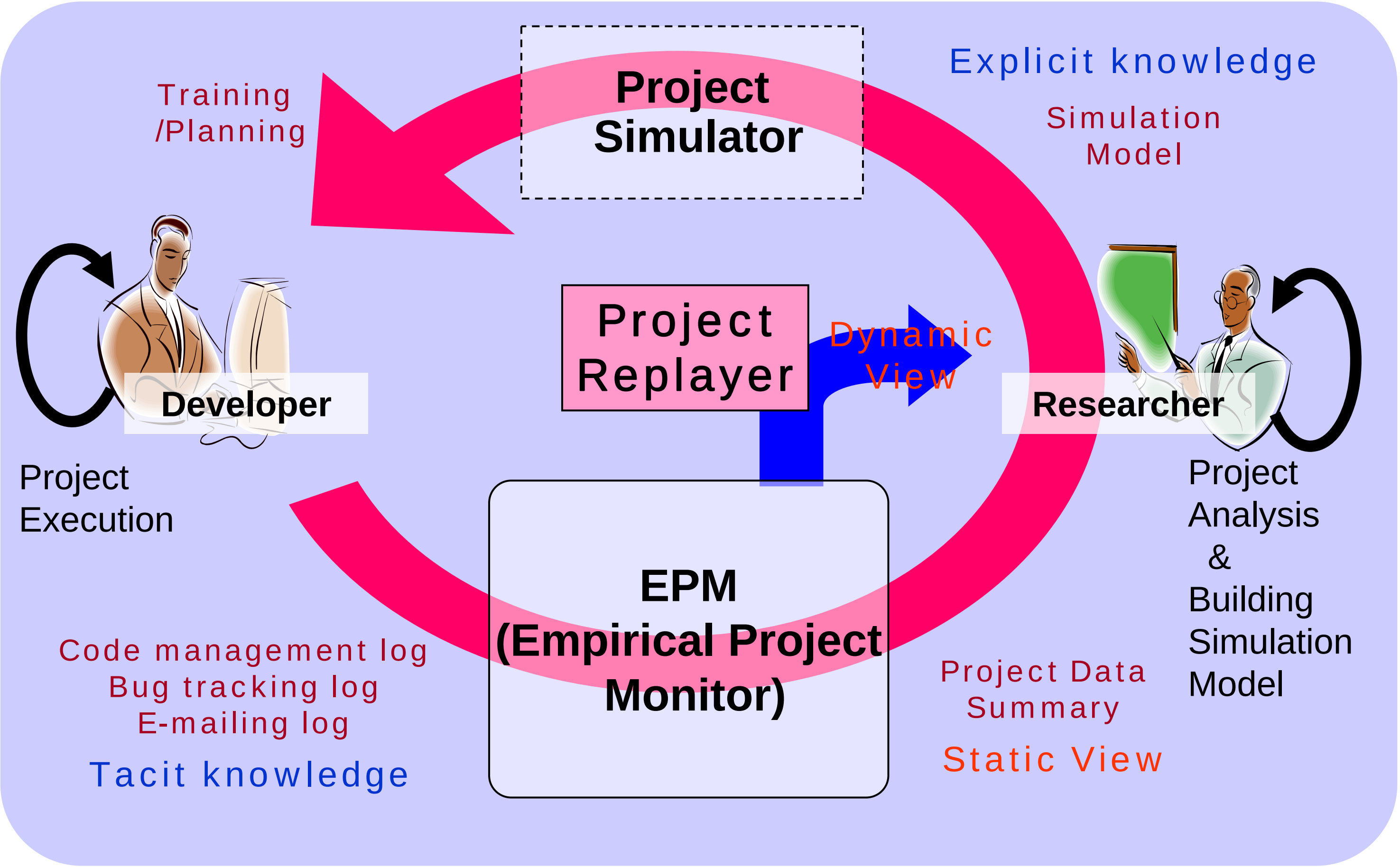
Knowledge Feedback Cycle (KFC) Overview



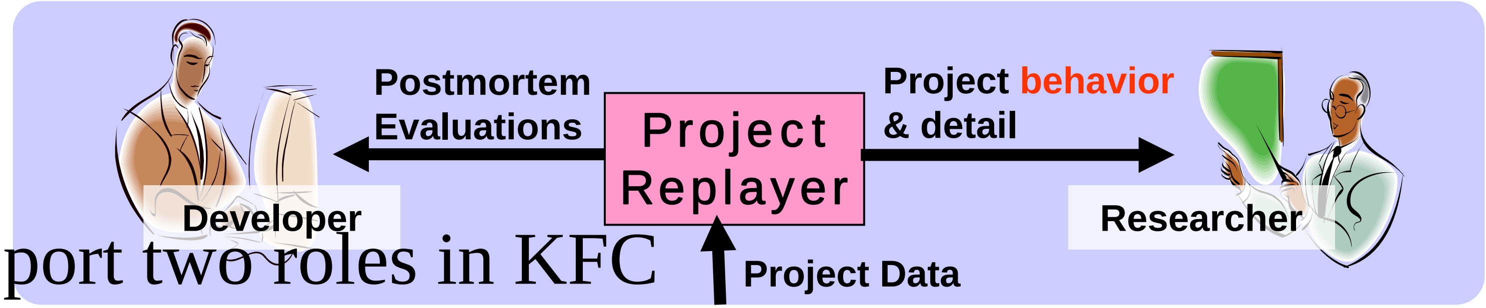
Knowledge Feedback Cycle (KFC) Overview



Knowledge Feedback Cycle (KFC) Overview



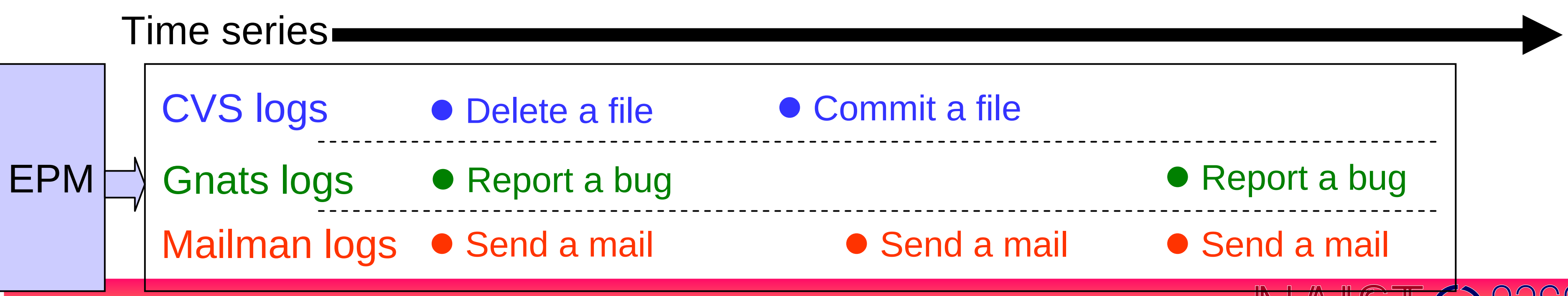
Project Replayer



- 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

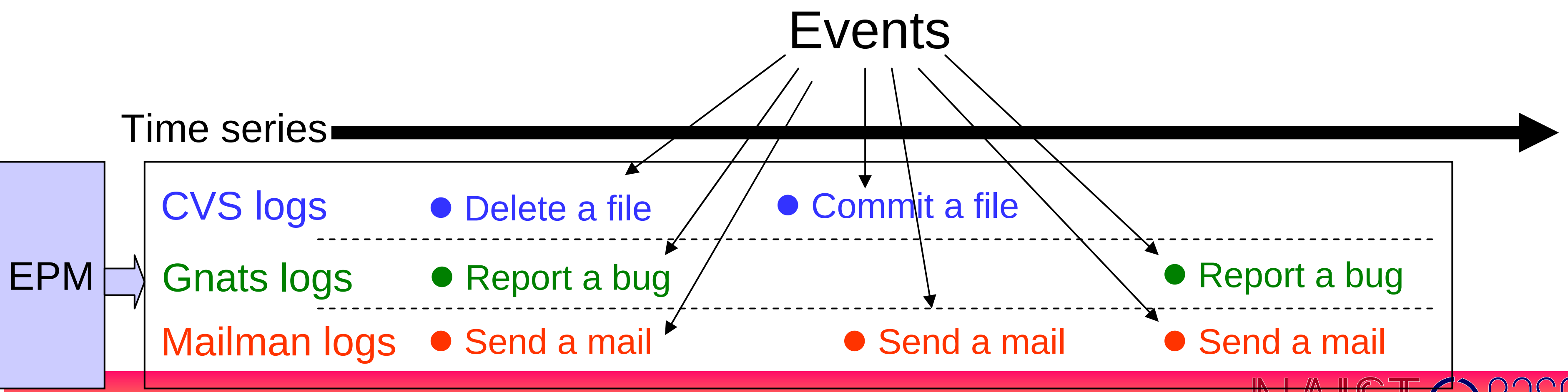
Showing Project data in a Dynamic Way

- Project Replayer reorganizes collected data by EPM, sorted in time order.



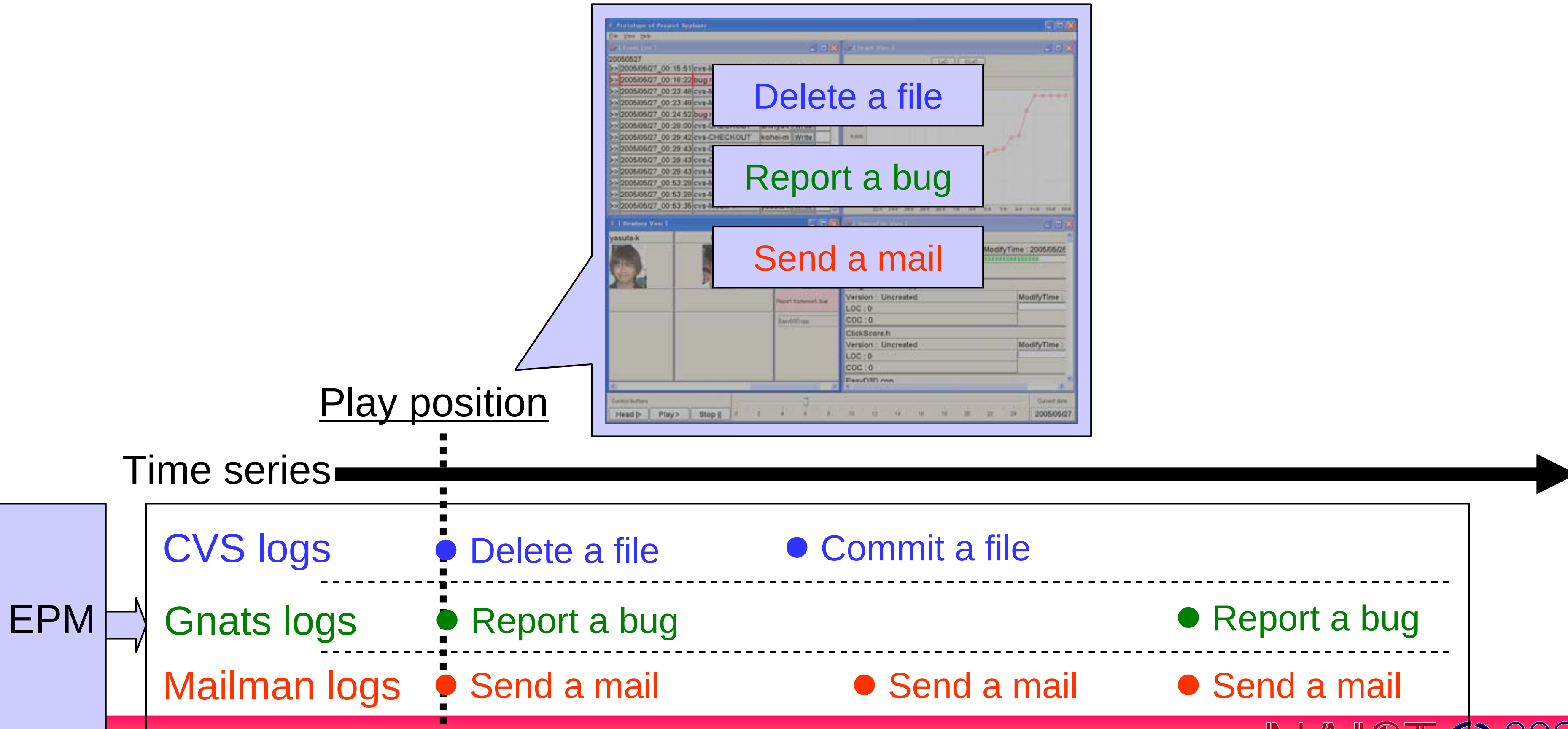
Showing Project data in a Dynamic Way

- Project Replayer reorganizes collected data by EPM, sorted in time order.



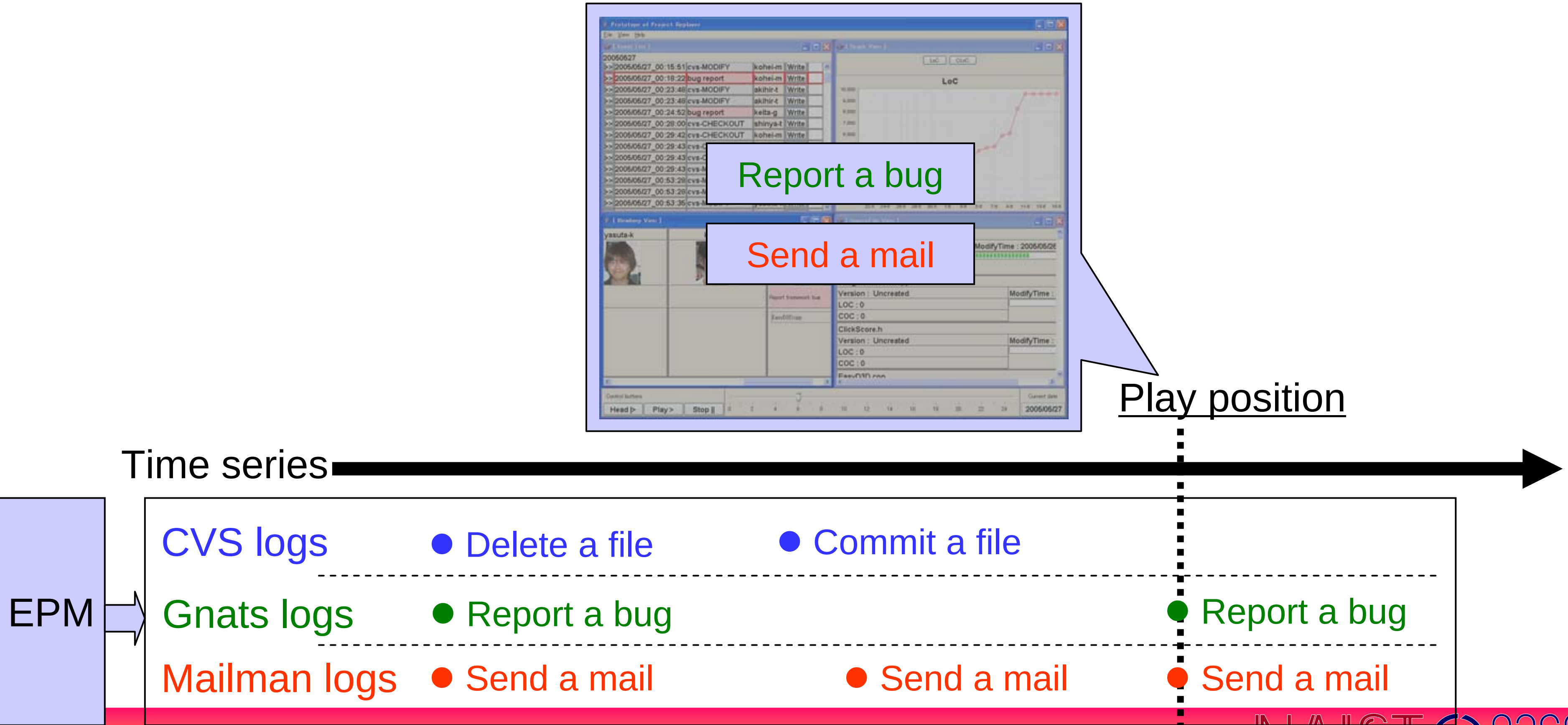
Showing Project data in a Dynamic Way

- Project Replayer shows events graphically and simultaneously.

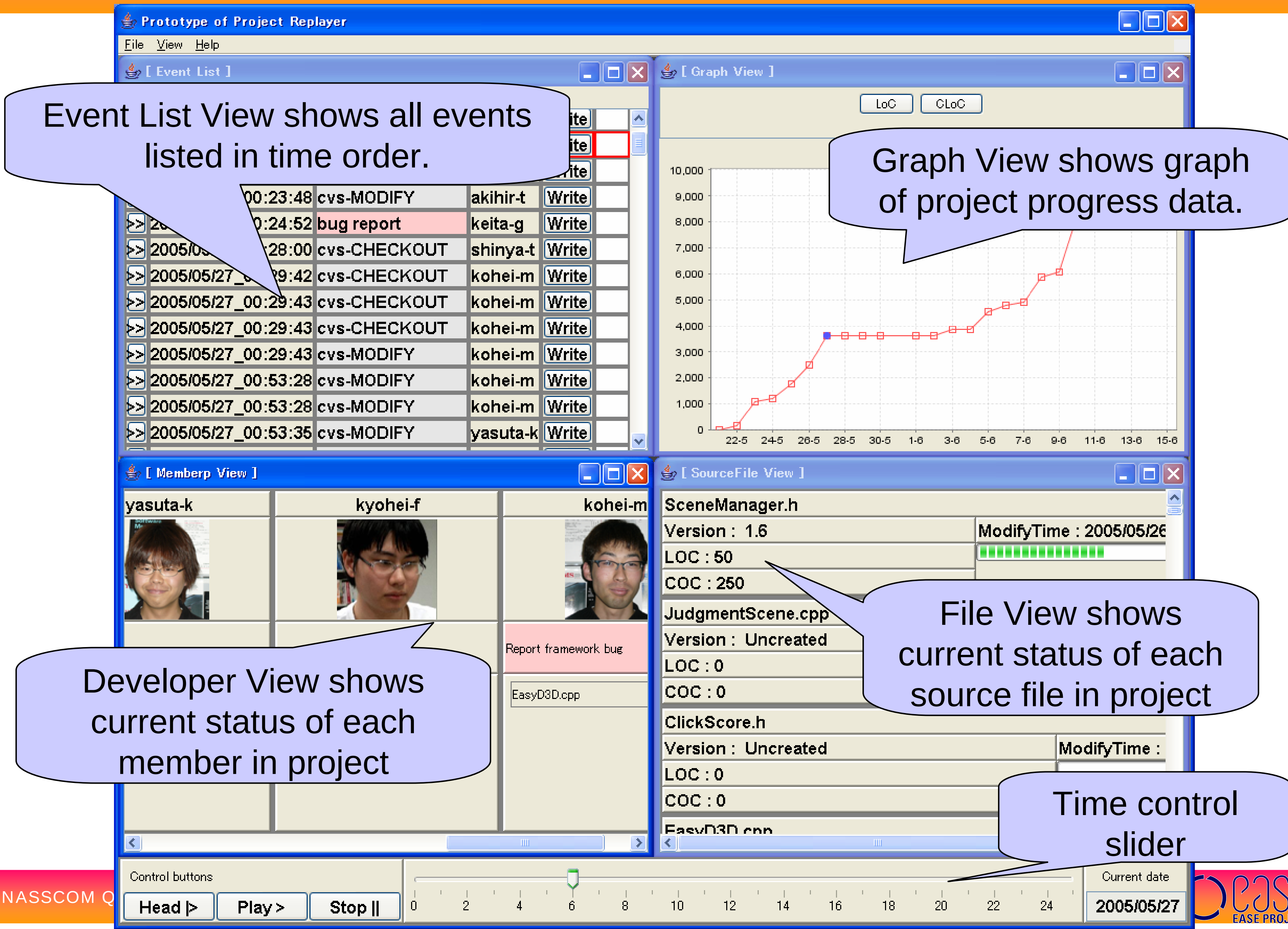


Showing Project data in a Dynamic Way

- Project Replayer shows events graphically and simultaneously.



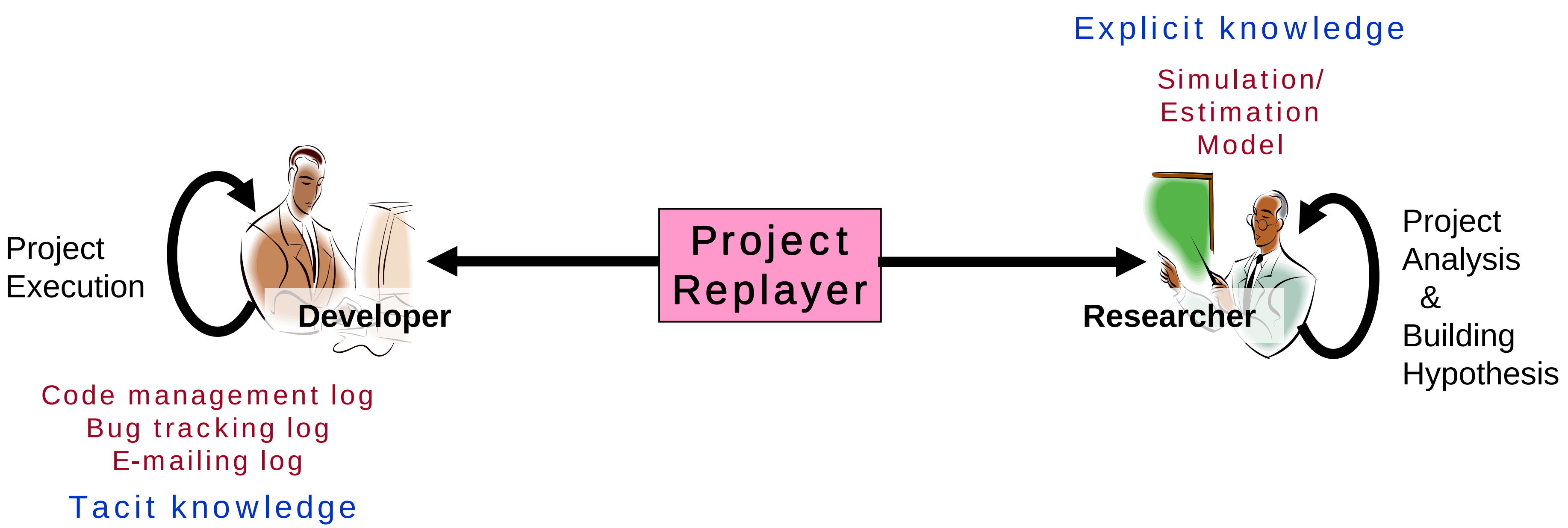
Views provided by Replayer



EASE PROJECT

Preliminary Experiment with Replayer in KFC

- Aims of the experiment
  1. Confirming benefits of developers and researchers with replaying.
  2. Confirming whether tacit knowledge can be transformed to explicit knowledge.



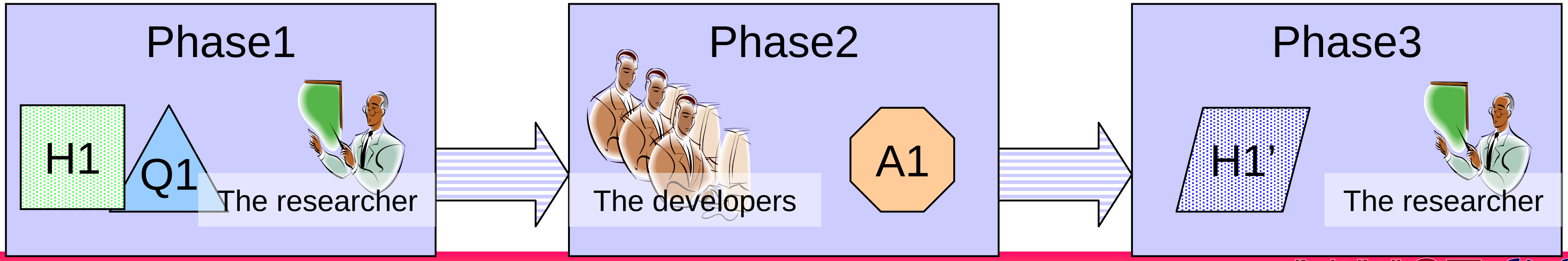
Setting of preliminary experiment

- 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)

Procedure of the experiment

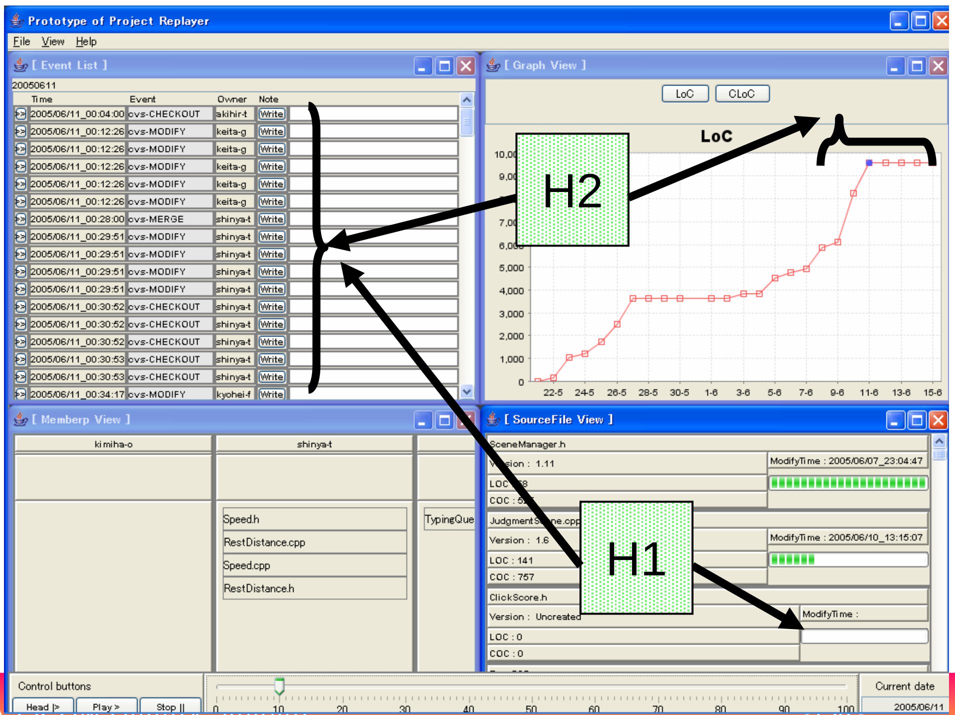
- The experiment was carried out after the target project was finished. Project data was automatically collected using EPM.

- 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.



Results - In Phase1 (1/2)

- 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.”



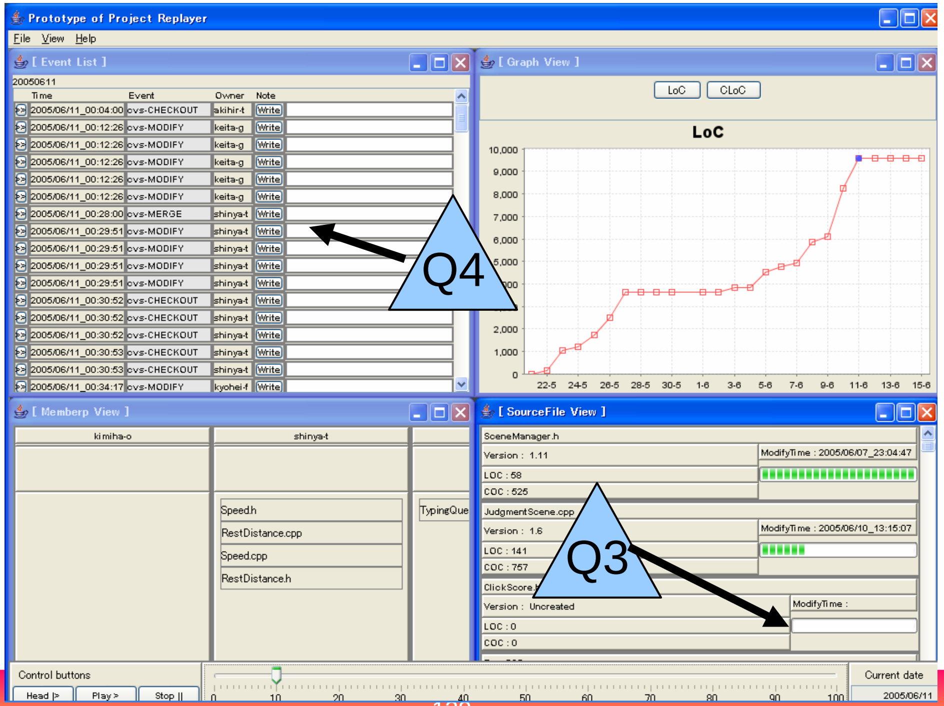
H1 :Developers did not start making some modules until a few days before deadline. The researcher expected that these modules have low quality. Therefore, the researcher established H1.

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.

Results - In Phase1 (2/2)

- 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)

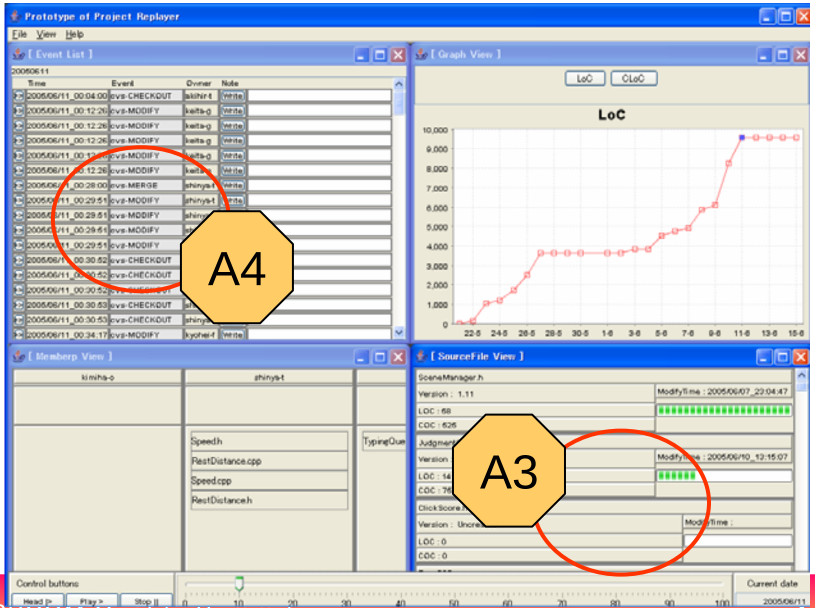
We only show the question directly related to 2 hypothesis.



Results - In Phase2

- 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 day were maintenance stage after the actual dead line.

NAIST ease EASE PROJECT

Results - In Phase3 (1/2)

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.

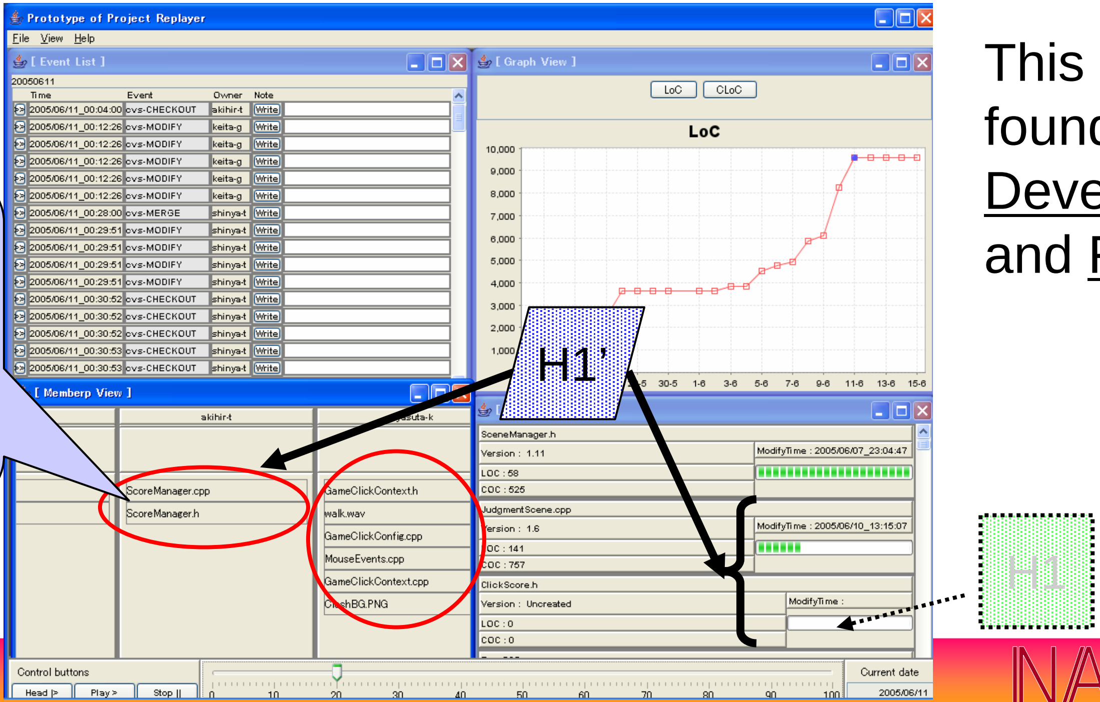
Additional analysis was performed using Project Replayer to refine H1.

Results - In Phase3 (2/2)

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”

Inexperienced developer was assigned to the low quality modules



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

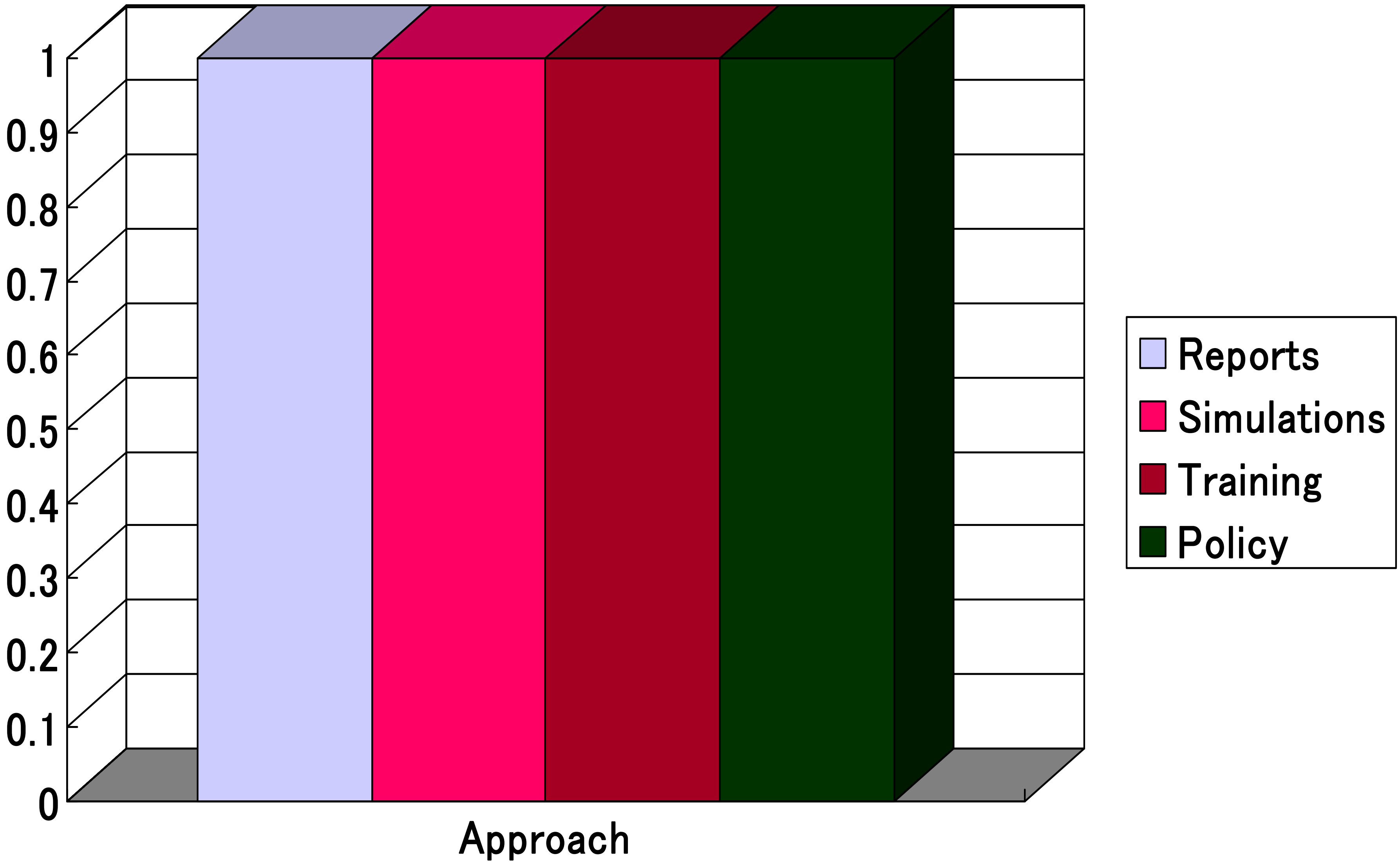
- In teams
- Estimate the effectiveness of reports, simulations, training in your organization
- Estimate the effort needed (project size) to create reports, simulations, training in your organization

Which do you think would be most effective in your organization?

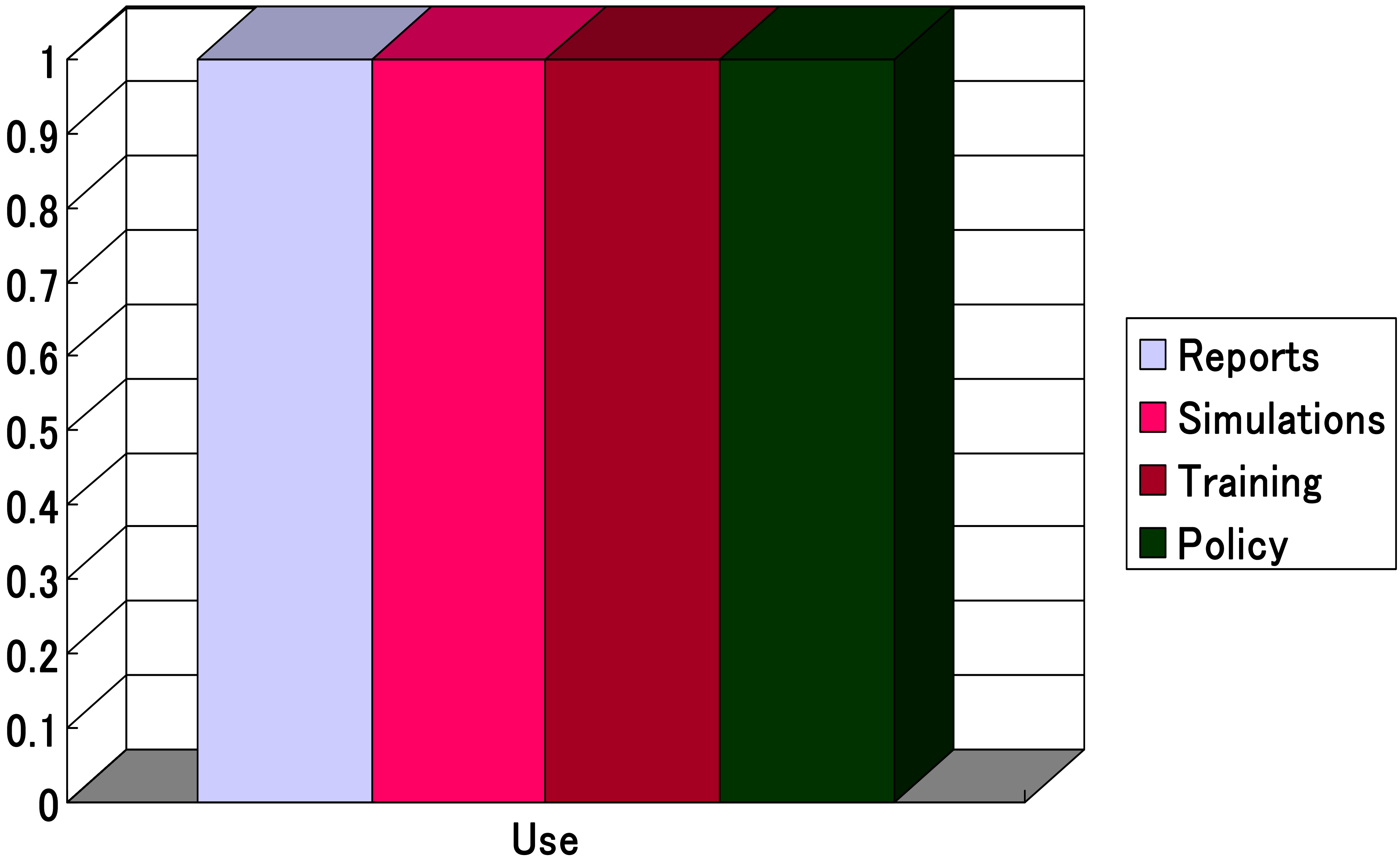
- Reports
- Simulations
- Training
- Policy Directive

Which do you actually use?

Which is most effective?



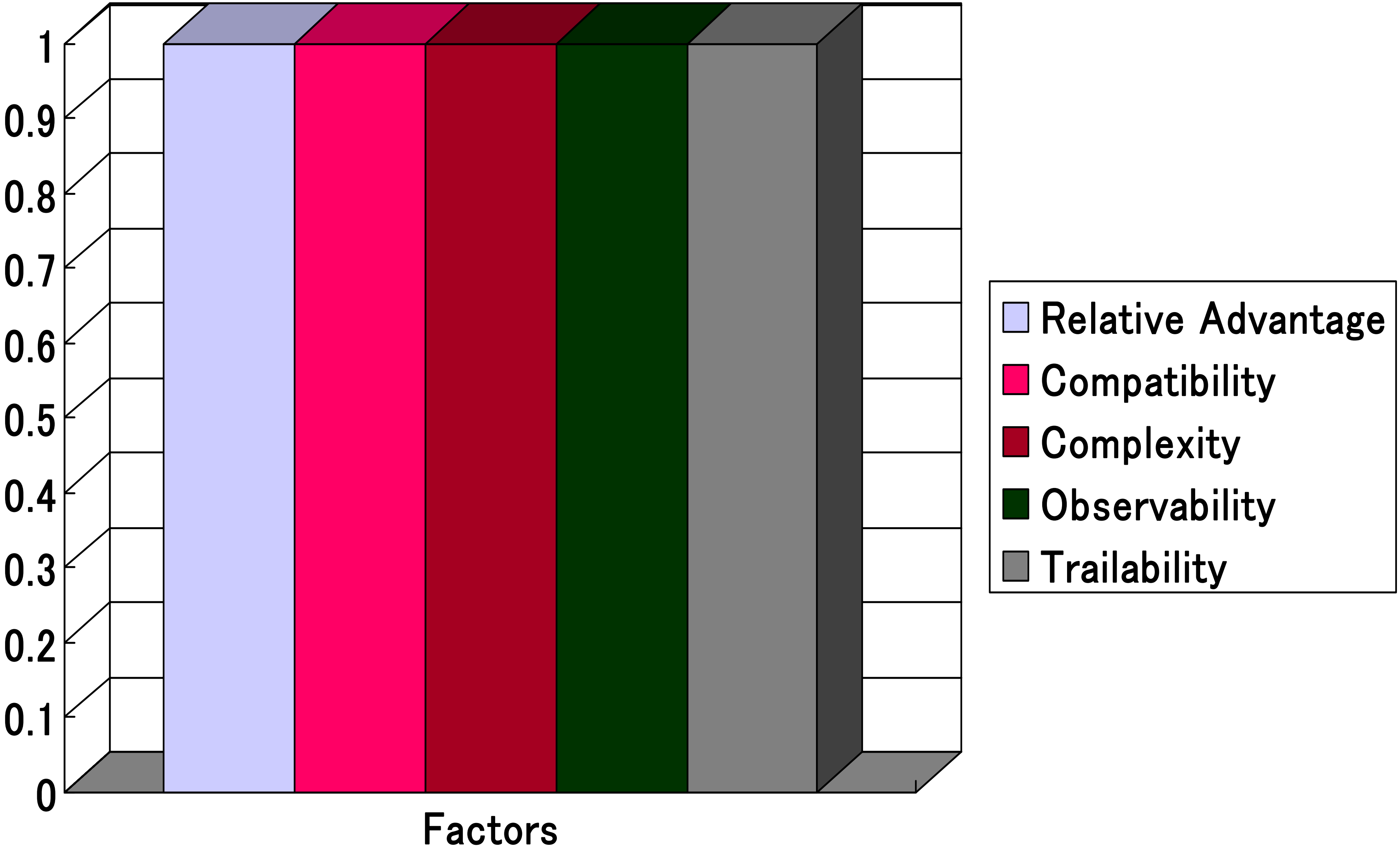
Which is most effective?



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?

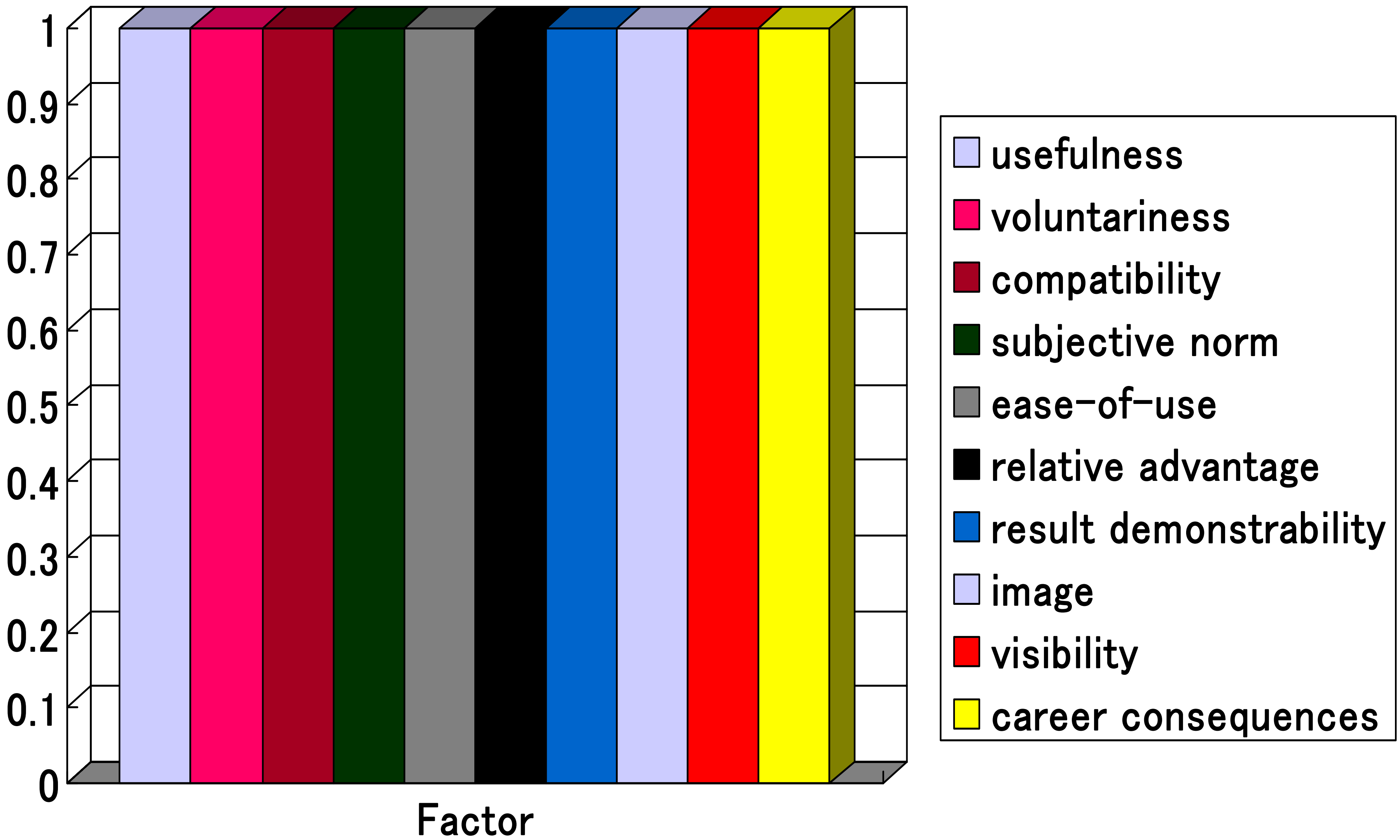


Acceptance of New Methods

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).

Acceptance of New Methods



Break!

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

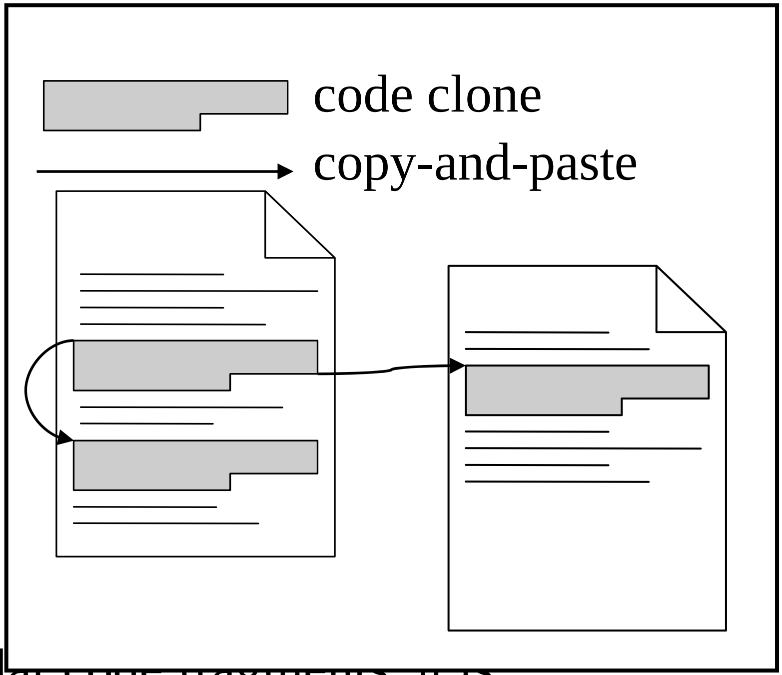
Talk Structure

- 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!



Simple Example

```
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];
}
```

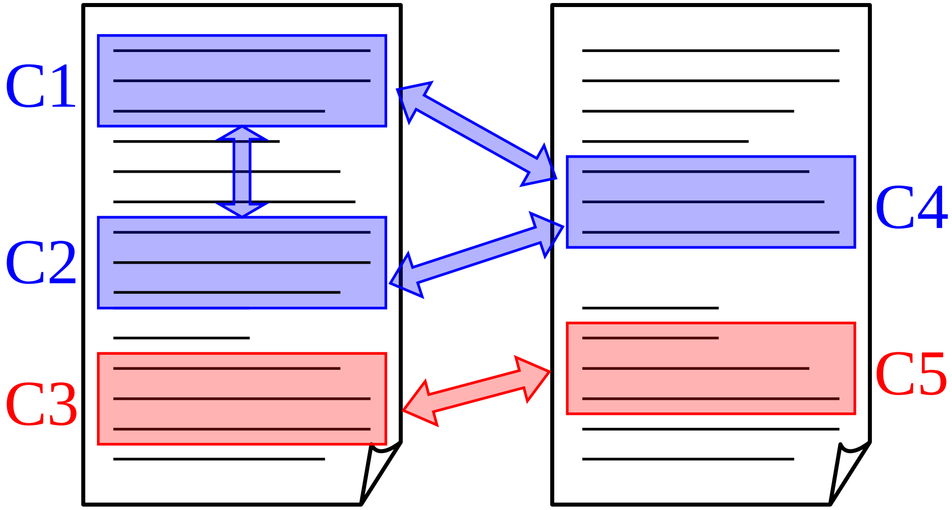
Definition of Code Clone

- 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 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

CCFinder and Associate Tools

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)	

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

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.

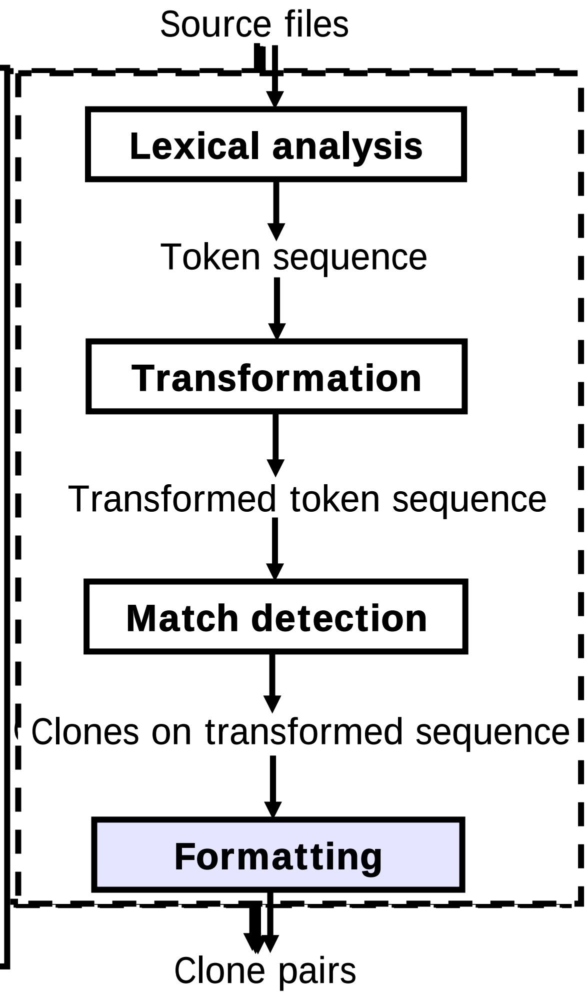
CCFinder Detection Process

static | void | foo | (|) | throws | RESyntaxException | { | String | a

[|] | = | new | String | [|] | { | "123,400" | ,

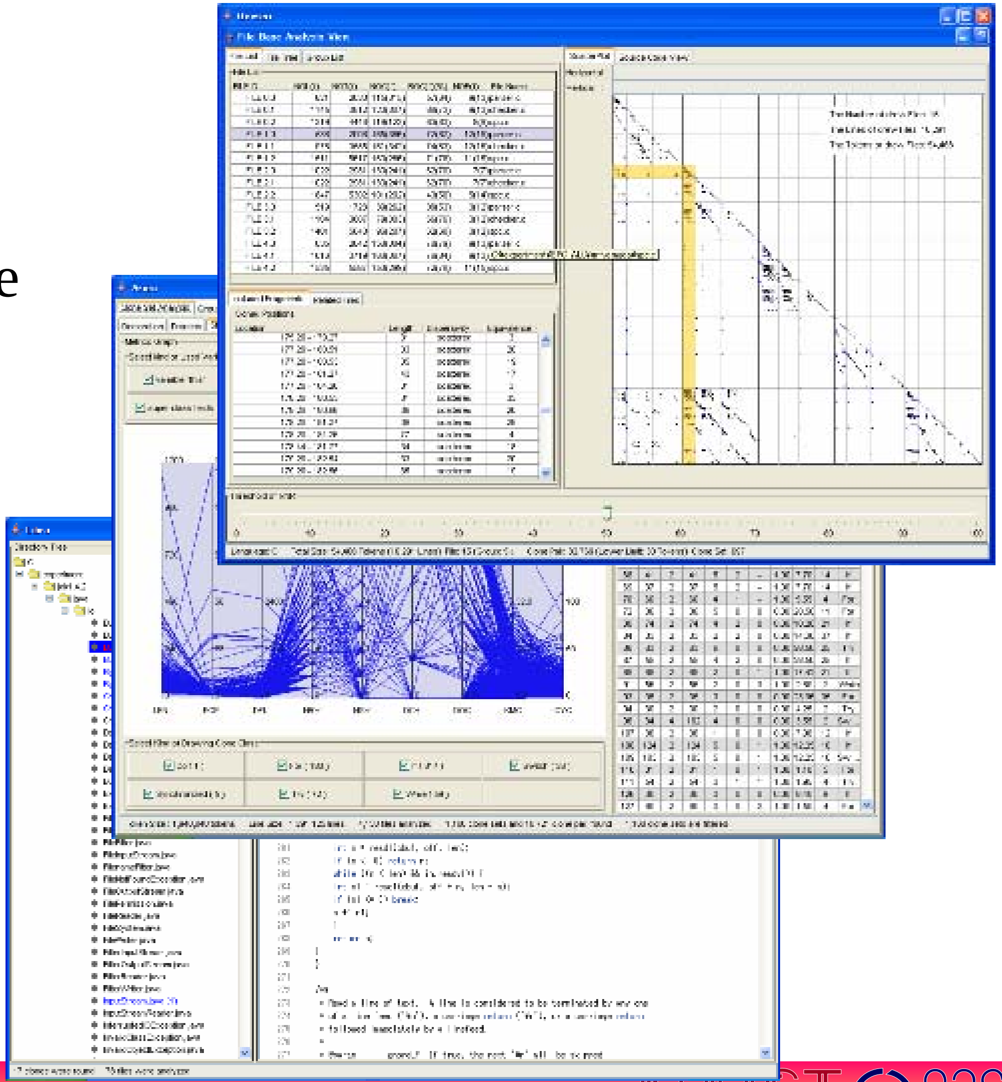
```
static void foo() throws RESyntaxException {
    String a[] = new String [] { "123,400", "abc", "orange 100" };
    org.apache.regexp.RE pat = new org.apache.regexp.RE("[0-9,]+");
    int sum = 0;
    for (int i = 0; i < a.length; ++i)
       if (pat.match(a[i]))
          sum += Sample.parseNumber(pat.getParen(0));
    System.out.println("sum = " + sum);
}
static void goo(String [] a) throws RESyntaxException {
    RE exp = new RE("[0-9,]+");
    int sum = 0;
    for (int i = 0; i < a.length; ++i)
       if (exp.match(a[i]))
          sum += parseNumber(exp.getParen(0));
    System.out.println("sum = " + sum);
}
```

) | , | }



Gemini Outline

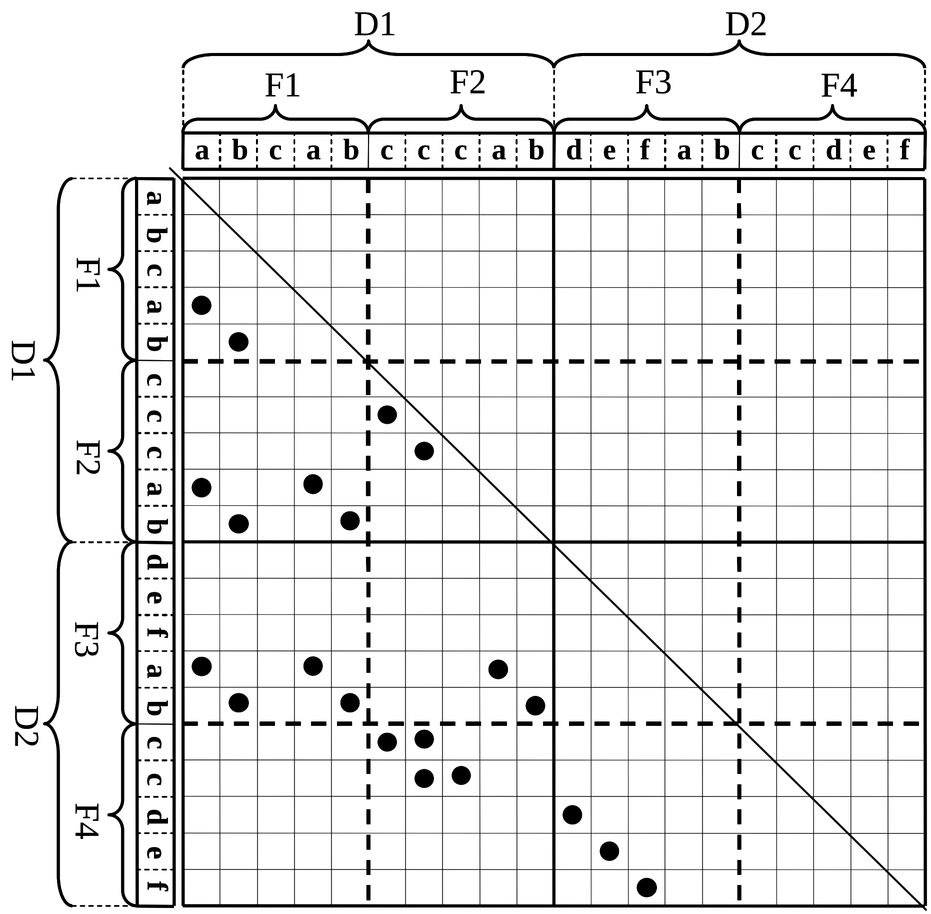
- Visualize code clones detected by CCFinder
  - CCFinder outputs the detection result to a text file
- Provide interactive analyses of code clones
  - Scatter Plot
  - Clone metrics
  - File metrics
- Filter out unimportant code clones



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)



F1, F2, F3, F4 : files
D1, D2 : directories

Gemini Clone and 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***

Refactoring Support System: Aries (1)

- 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

Code clones which CCFinder detects

Code clones which Aries extracts

fragment 1

```
609:    reset();
610:    grammar = g;
611:    // Lookup make-switch threshold in
612:    if (grammar.hasOption("codeGenM
613:        try {
614:            makeSwitchThreshold = g
615:            //System.out.println("setti
616:        } catch (NumberFormatExcept
617:            tool.error(
618:                "option 'codeGenMa
619:                grammar.getClassN
620:                grammar.getOption(
621:            );
622:        }
623:    }
624:
625:    // Lookup bitset-test threshold in the
626:    if (grammar.hasOption("codeGenBi
627:        try {
628:            bitsetTestThreshold = gra
```

fragment 2

```
623:    }
624:
625:    // Lookup bitset-test threshold in the gr
626:    if (grammar.hasOption("codeGenBitset
627:        try {
628:            bitsetTestThreshold = gramm
629:            //System.out.println("setting
630:        } catch (NumberFormatException
631:            tool.error(
632:                "option 'codeGenBitsetT
633:                grammar.getClassNam
634:                grammar.getOption("co
635:            );
636:        }
637:    }
638:
639:    // Lookup debug code-gen in the gram
640:    if (grammar.hasOption("codeGenDebu
641:        Token t = grammar.getOption("co
642:        if (t.getText().equals("true")) {
```

Refactoring Support System: Aries (2)

- 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.

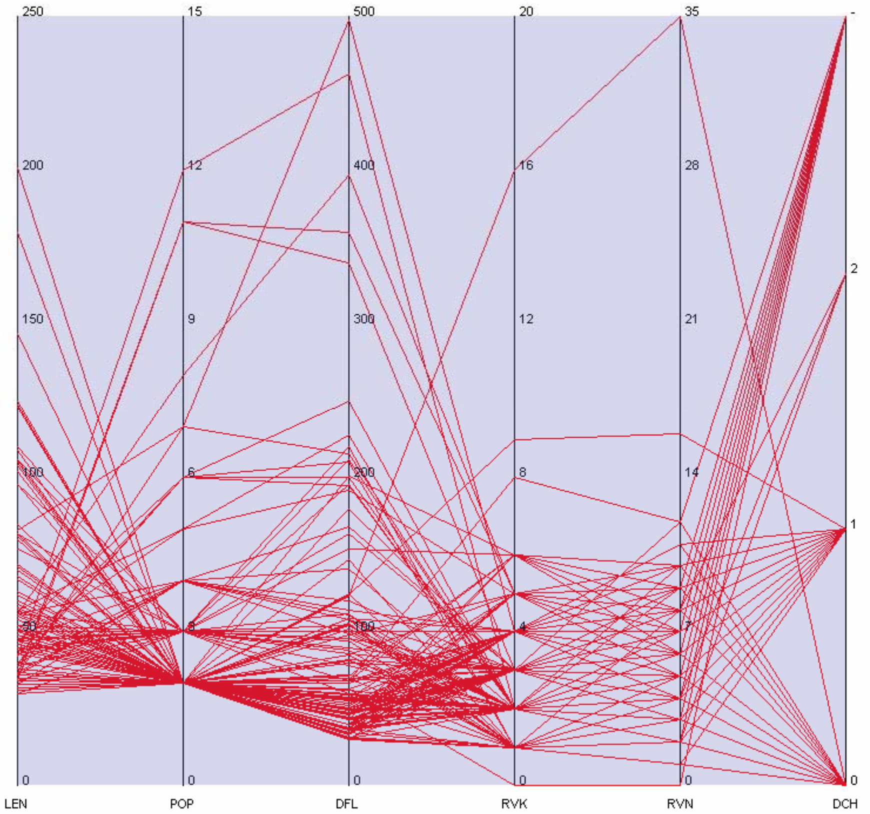
[1]: M. Fowler: Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.
[2]: http://www.refactoring.com/, 2004.

Cancer

Metrics Graph

Select kind of RVK Variables

☑ variable "this" ☑ variable "super" ☑ local variables ☑ this class fields ☑ super class fields ☑ interface fields



Select Kind of Drawing Clone Class

☐ Class (0) ☐ Interface (0) ☑ Method (48) ☑ Constructor (1) ☐ Static (0) ☑ Do (1)

☑ For (12) ☑ If (67) ☐ Switch (0) ☐ Synchronized (0) ☑ Try (8) ☑ While (10)

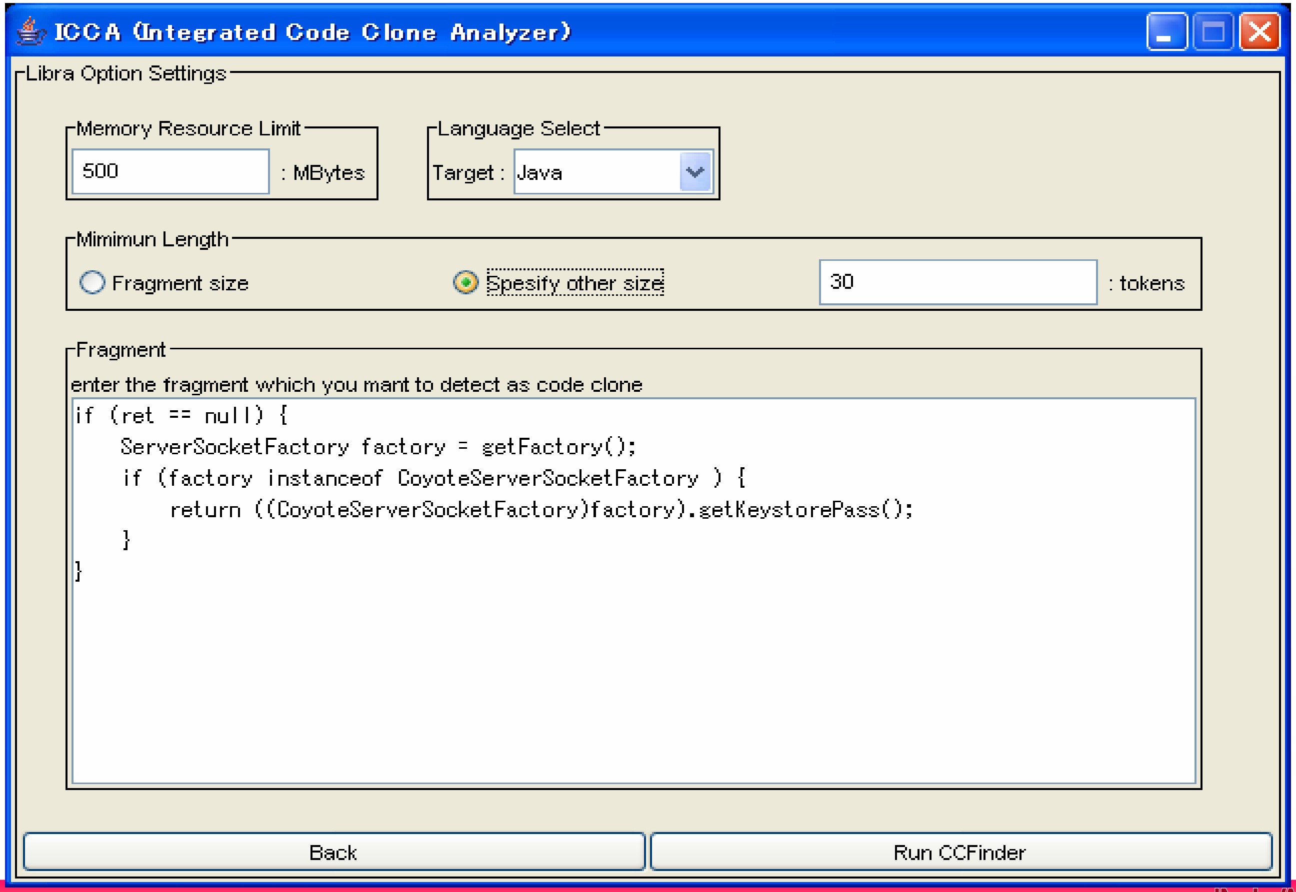
Clone Class List

LEN	POP	DFL	RVK	RVN	DCH	Type
48	2	48	3	9	0	If
49	2	49	3	3	0	For
40	2	40	2	3	0	Method
105	2	105	1	1	0	Method
45	3	90	3	4	0	Method
102	2	102	2	7	-	Method
36	4	108	1	1	0	Method
40	2	40	3	4	0	Try
44	2	44	3	4	-	While
51	2	51	3	6	-	For
62	2	62	2	2	0	Method
39	6	195	2	3	0	For
36	4	108	3	4	0	Method
70	2	70	3	3	1	Method
56	2	56	3	6	1	If
44	3	88	1	5	-	For
82	2	82	3	3	0	Method
67	2	67	4	11	0	Method
46	2	46	3	3	-	If
80	2	80	3	5	-	If
180	2	180	3	12	-	If
85	2	85	5	9	0	Method
43	2	43	3	4	0	While
49	2	49	3	6	0	While
34	3	68	4	8	-	Do
36	2	36	3	4	0	While
37	2	37	2	5	1	Method
68	2	68	3	4	1	While
47	5	192	5	5	0	If
58	2	58	2	5	1	Method
98	2	98	5	8	1	Method
38	2	38	2	4	1	If
38	2	38	2	4	2	Try
31	2	31	1	2	0	If
37	2	37	6	8	0	If
33	2	33	5	6	0	If
54	2	54	3	4	0	Method
51	2	51	4	6	0	Method
57	2	57	5	10	0	If
40	2	40	4	6	1	If
39	3	78	2	3	0	While
57	2	57	2	4	0	If
60	2	60	2	3	0	While
43	2	44	2	3	1	Method
37	2	37	3	4	0	While
35	2	35	3	4	0	If
124	2	124	9	16	1	Constructor
49	2	49	2	2	0	For
47	2	47	4	6	0	If
36	2	36	2	5	0	If
57	2	57	2	6	0	Method
35	2	36	3	5	-	If
68	2	68	6	9	0	If
36	2	36	4	5	0	Method
58	2	58	4	6	0	If
85	2	85	1	5	1	Method

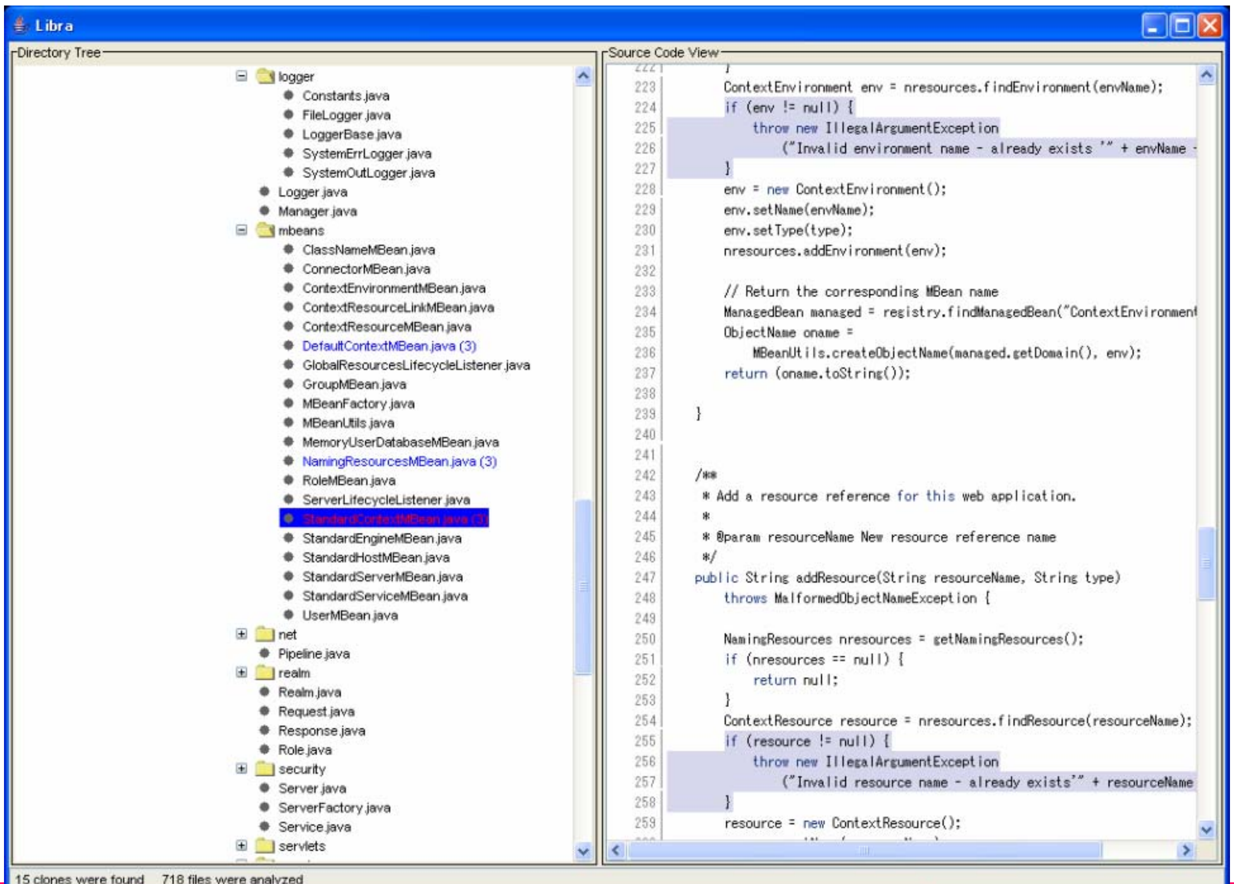
Token Size : 214243 tokens Line Size : 137354 lines 514 files analyzed 147 clone classes and 544 clone pair found

Change Support System: Libra

- Input a code fragment



- Find clones between the input and target



Applications

Code Clone Seminar

- 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

This is YOUR presentation!

- 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?

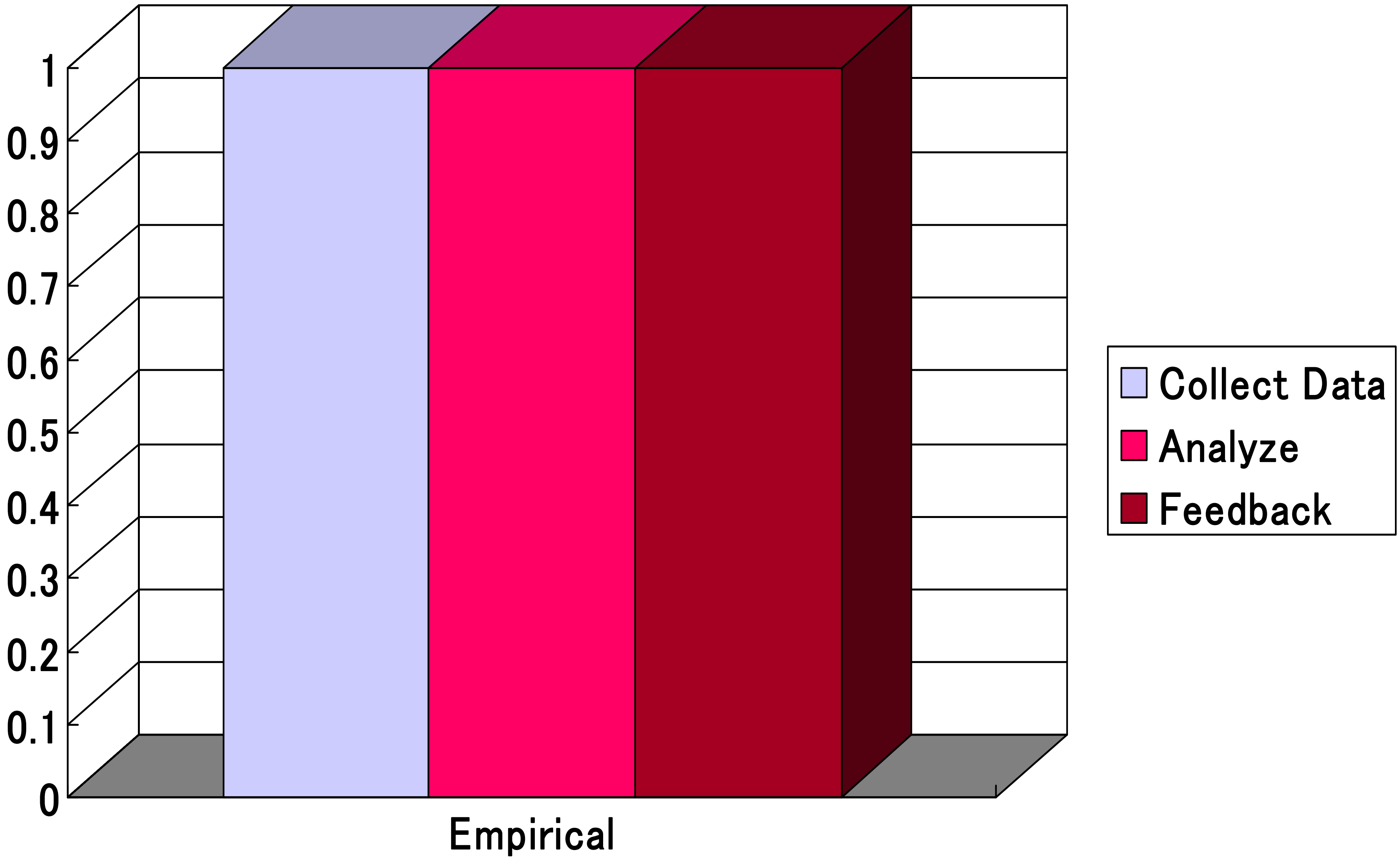
Team Exercise

- 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?

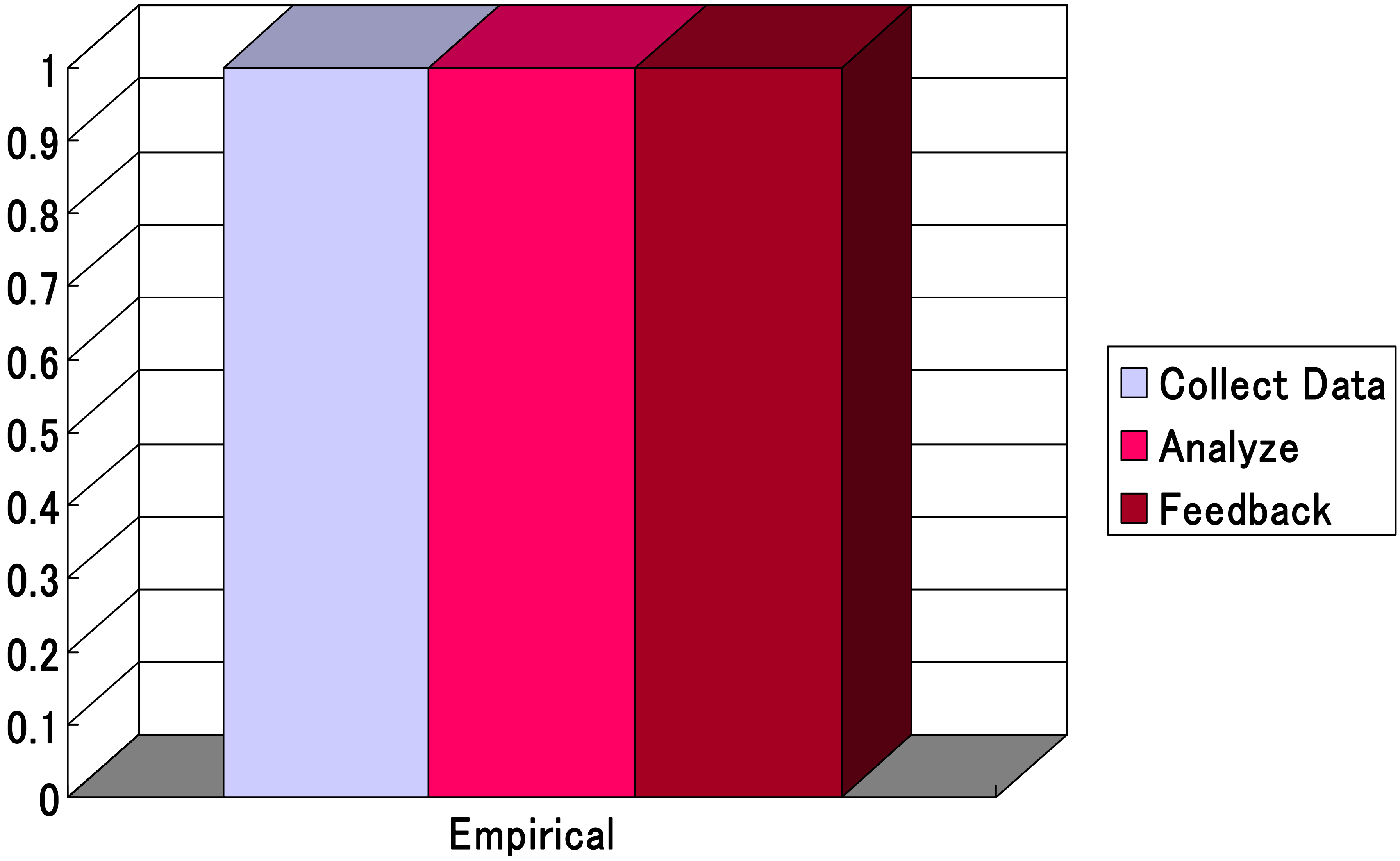
Large Group

- 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?

Plan

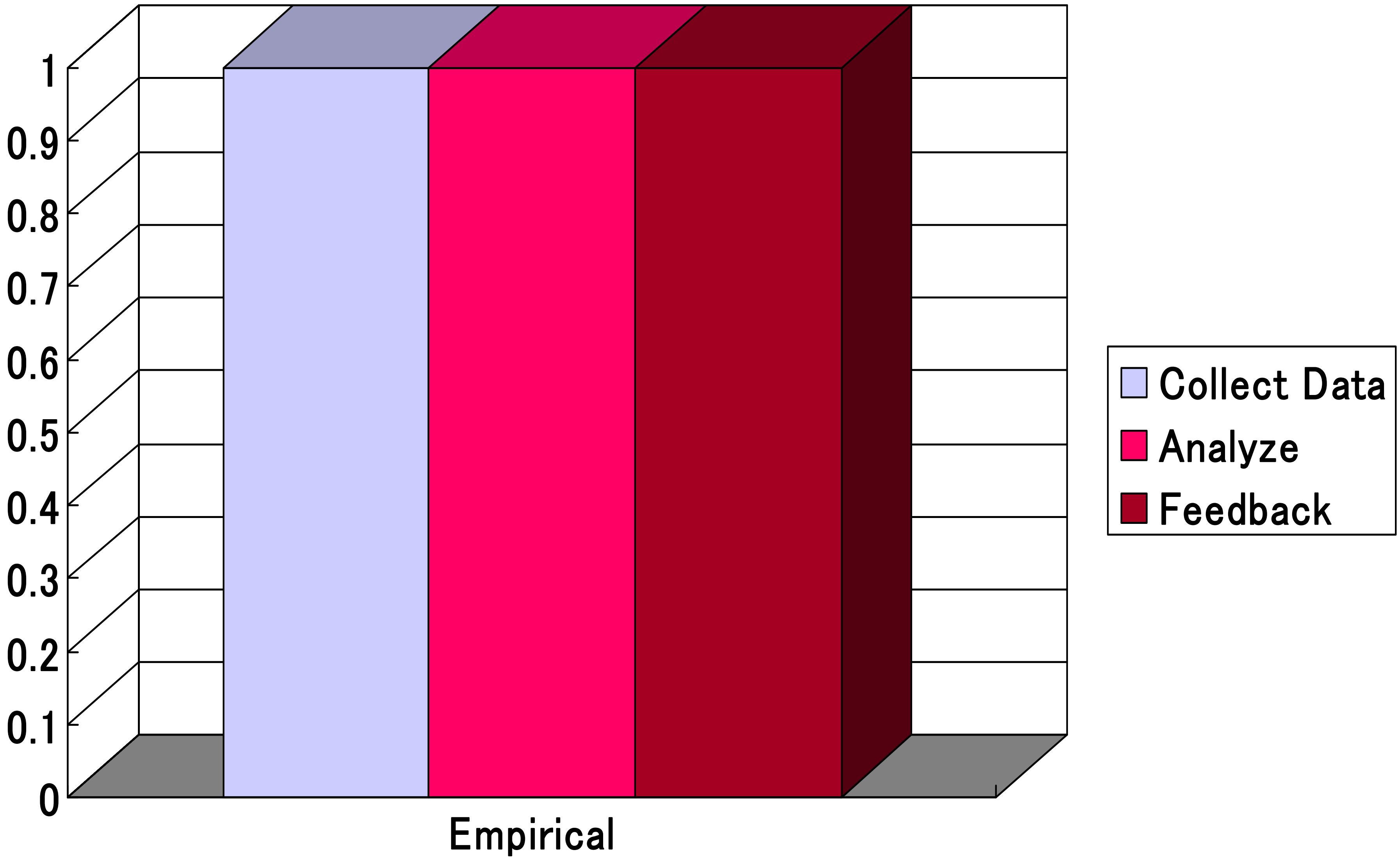

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Empirical
Collect Data
Analyze
Feedback

Do

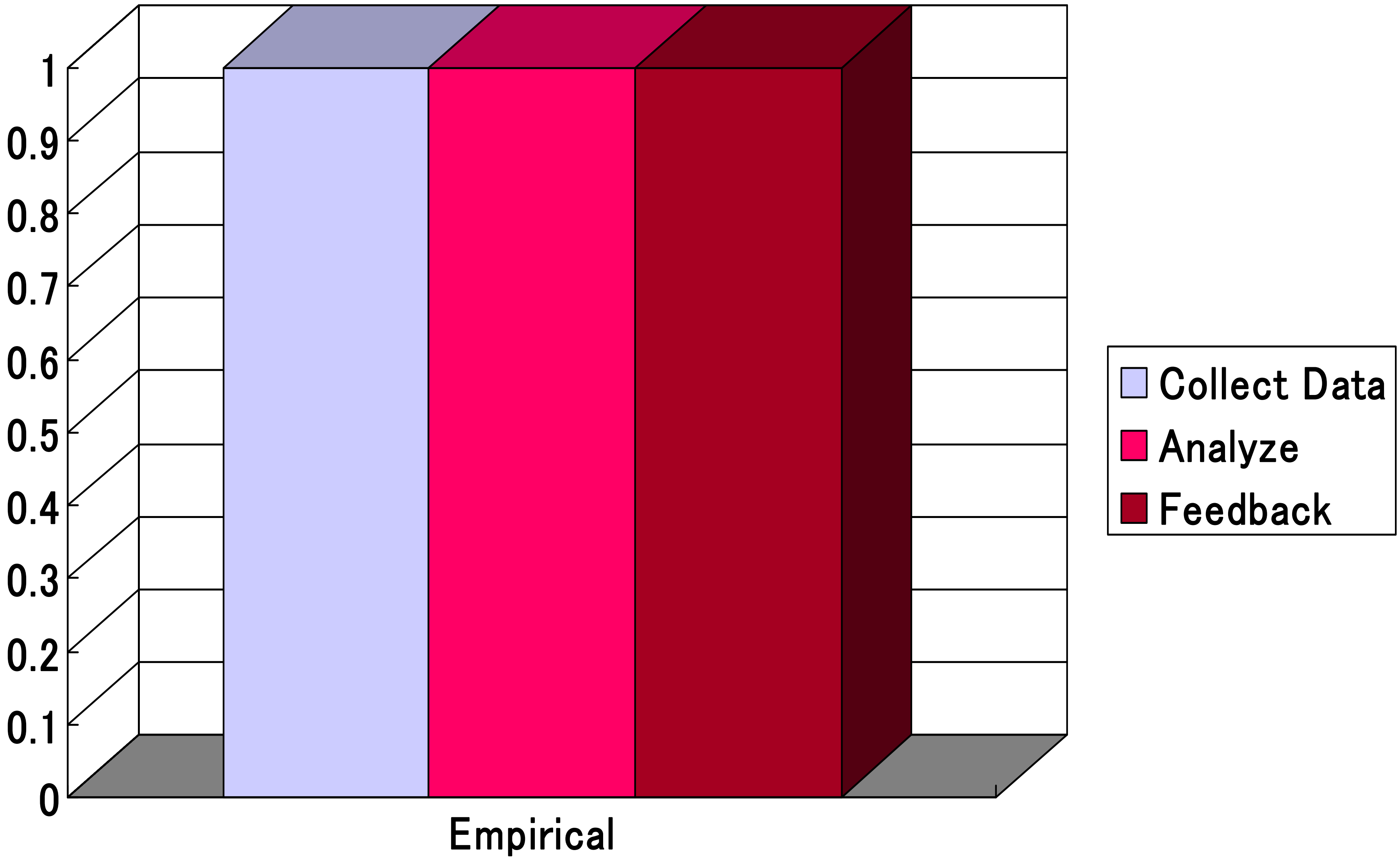


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Empirical
Collect Data
Analyze
Feedback

Check


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Empirical
Collect Data
Analyze
Feedback

Act


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Empirical
Collect Data
Analyze
Feedback

Break!

Summary, Next Steps, and Conclusions

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

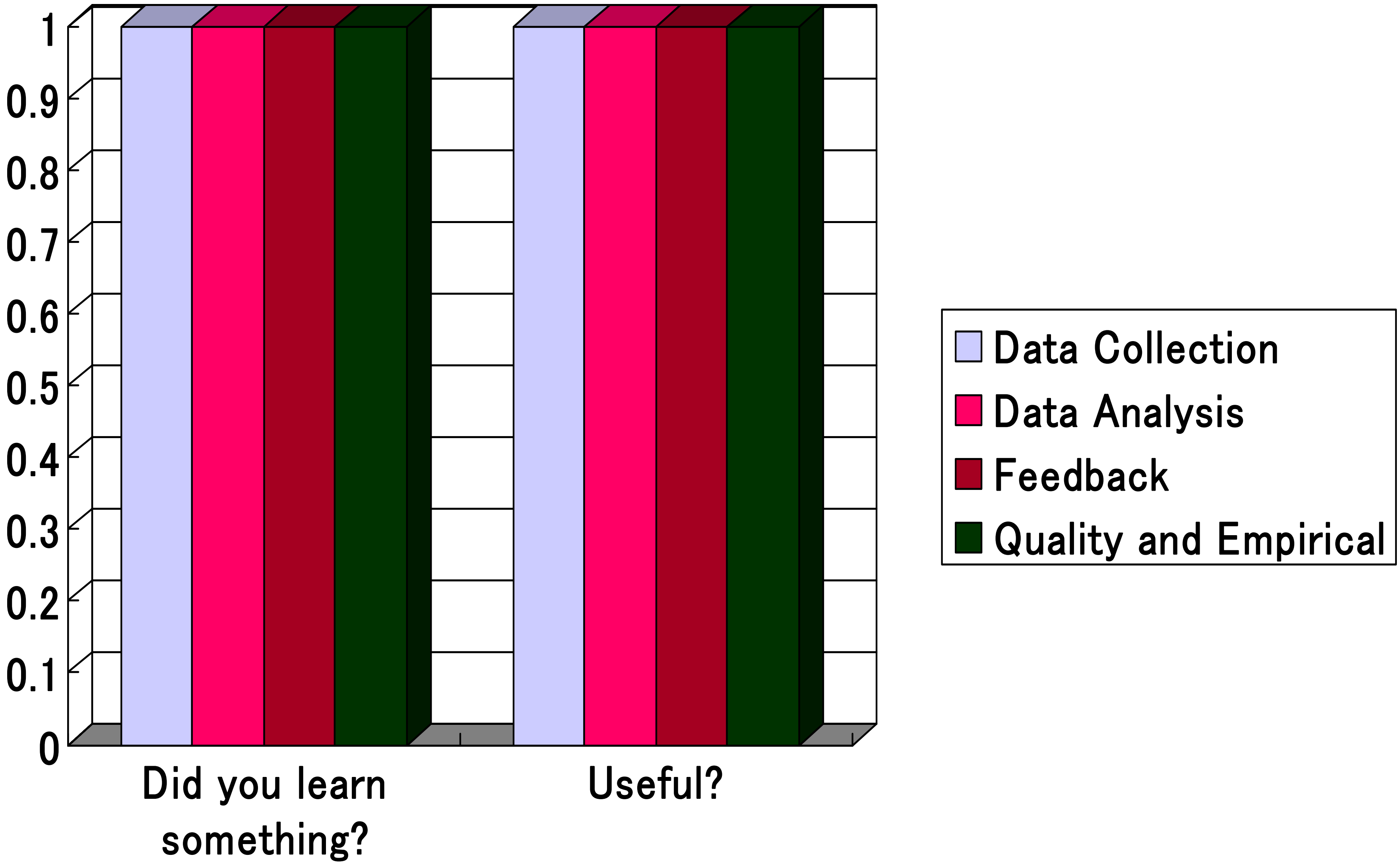
Next Steps?

- 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.

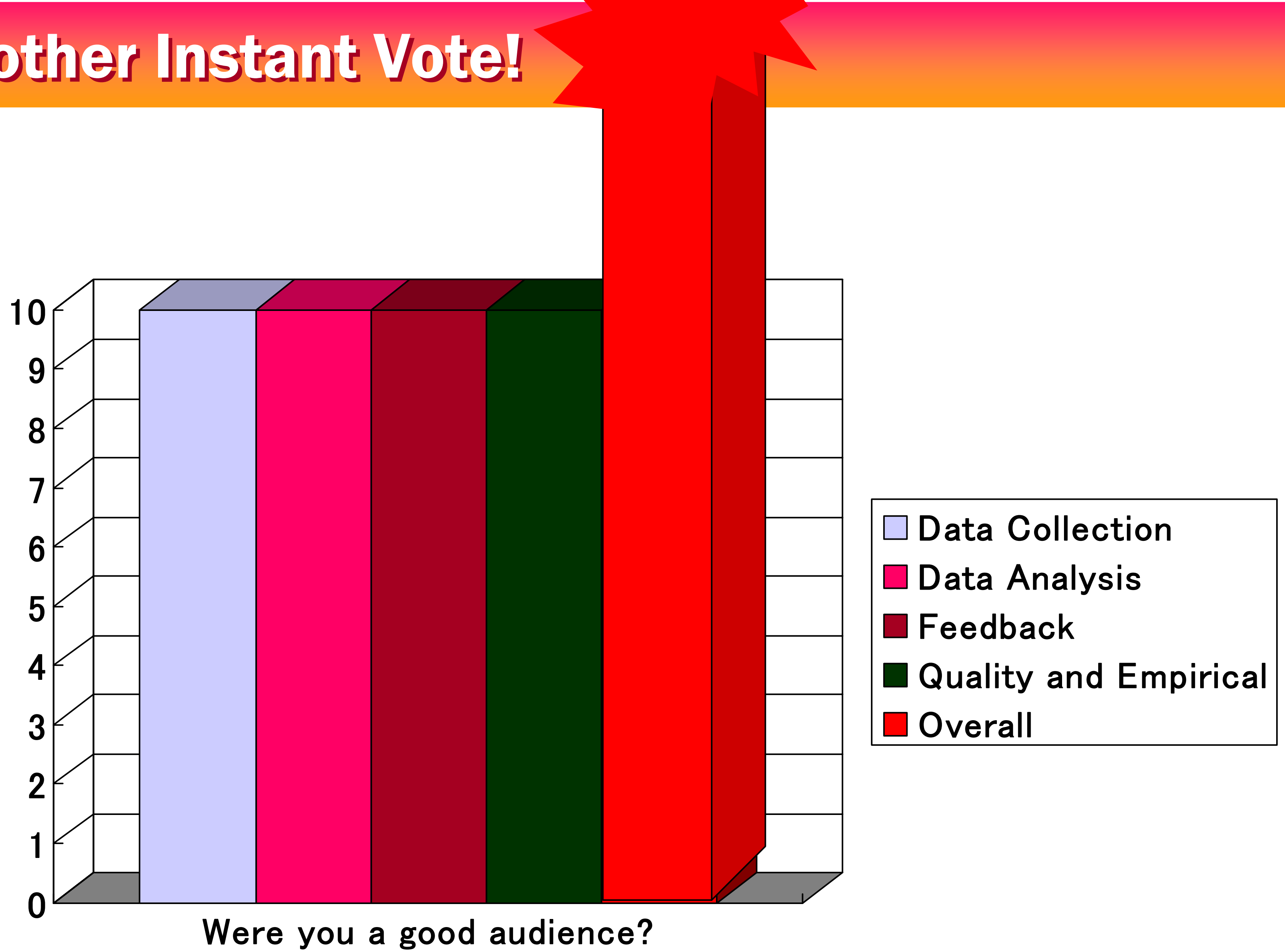
Your Questions?

- If you want to send us questions later, send them to mbarker@computer.org

Instant Vote!



Another Instant Vote!



Thank you all!
どうもありがと

(Remember to turn in your takeaway sheets!)