

# Capability Assessment of Individual Software Development Processes Using Software Repositories and DEA

*Shen Zhang, Yongji Wang, Ye Yang and Junchao Xiao*

*Laboratory for Internet Software Technologies  
Institute of Software, Chinese Academy of Sciences*

# Agenda

- Introduction
- Related Work
- Approach Overview
  - Metric Extraction Method
  - Capability Assessment Method
- Experimental Results and Analysis
- Conclusions

# Agenda

- *Introduction*
- Related Work
- Approach Overview
  - Metric Extraction Method
  - Capability Assessment Method
- Experimental Results and Analysis
- Conclusions

# Background

- People are the largest source of variation in project performance
- Effective capability assessment of individual software processes is a key issue involved in validating the past and adjusting future development processes
- Current workforce practices
  - P-CMM
  - PSP
- Challenges
  - Burden on collecting measures manually
  - Privacy and integrity issues
  - Subjective bias

# Our Approach

- Integrate software repositories and data envelopment analysis (DEA) for the assessment of individual development process
  - Facilitate the process metrics extraction task by making use of software repositories as the data source
  - Decompose the process metrics at the individual level of granularity by exploring different human identities from various sources
  - A DEA model is presented for the quantitative capability assessment of individual development process

# Agenda

- Introduction
- *Related Work*
- Approach Overview
  - Metric Extraction Method
  - Capability Assessment Method
- Experimental Results and Analysis
- Conclusions

# Software Repositories

- Software repositories contain a wealth of valuable information
  - The progress of software projects
  - A detailed log of the evolution of human activities during development
- Liu et al. use CVS historical information to understand how students develop software
- Mierle et al. mine and analyze a set of assignment repositories to extract the quantitative indicators of students' performance

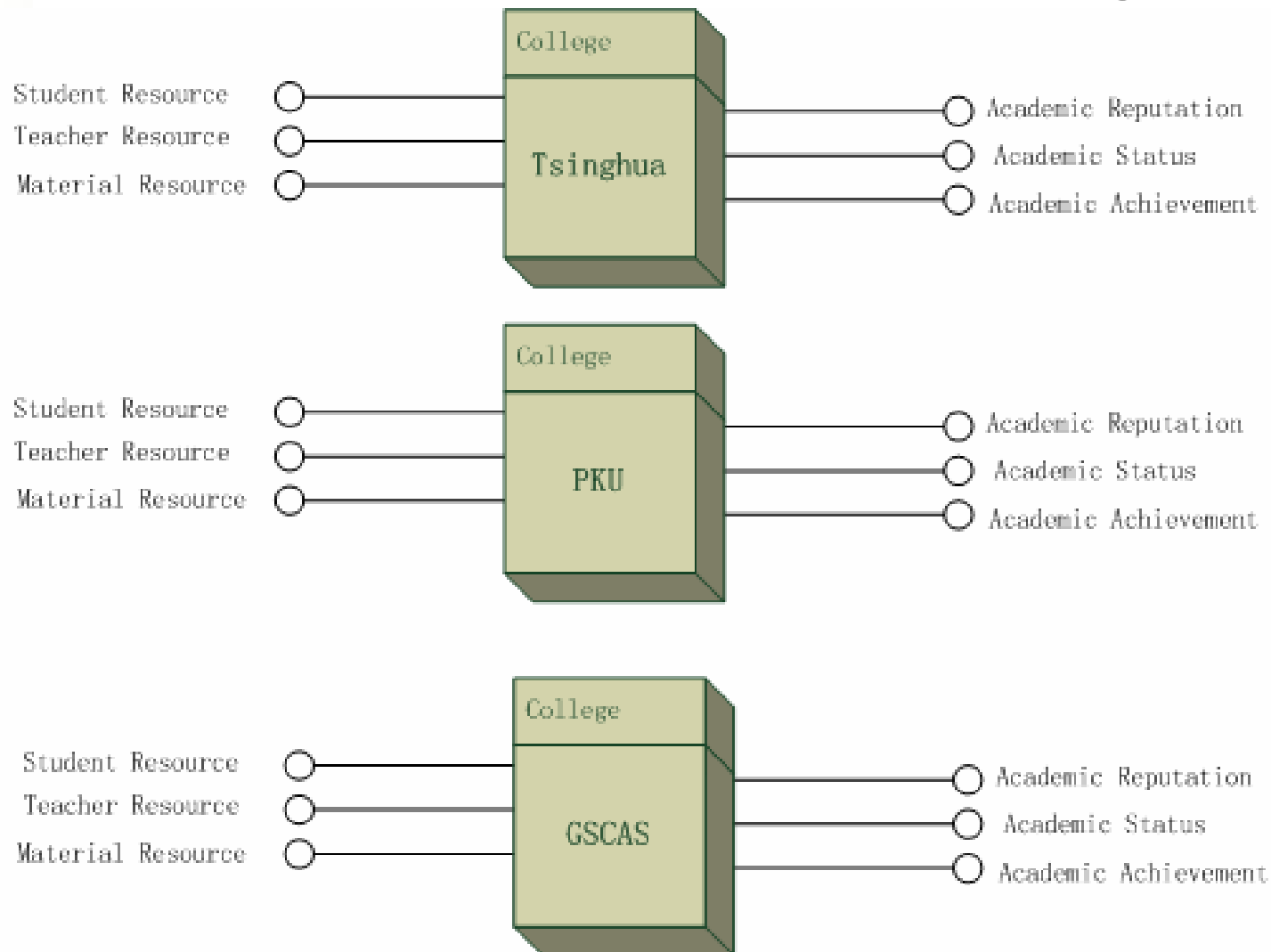
# Data Envelopment Analysis

- Data Envelopment Analysis (DEA) developed by A. Charnes and W. W. Cooper in 1978 is a non-parametric programming-based performance assessment model.
- DEA can be used to analyze the relative performance of a number of units, which can be viewed as a multi-input-multi-output system consuming inputs to produce outputs.

# Data Envelopment Analysis (cont.)

- Inputs are defined as any factors used as a resource by the Decision Making Unit (DMU) for producing something of value. It may also be any environmental factor that has a strong effect on how resources are consumed
- Outputs are the amounts of goods, services or other outcomes obtained by processing resources or, also, any factors that describes the qualitative nature of such an outcome

# Example: Compare the scientific research level of colleges



m different inputs to produce s different outputs

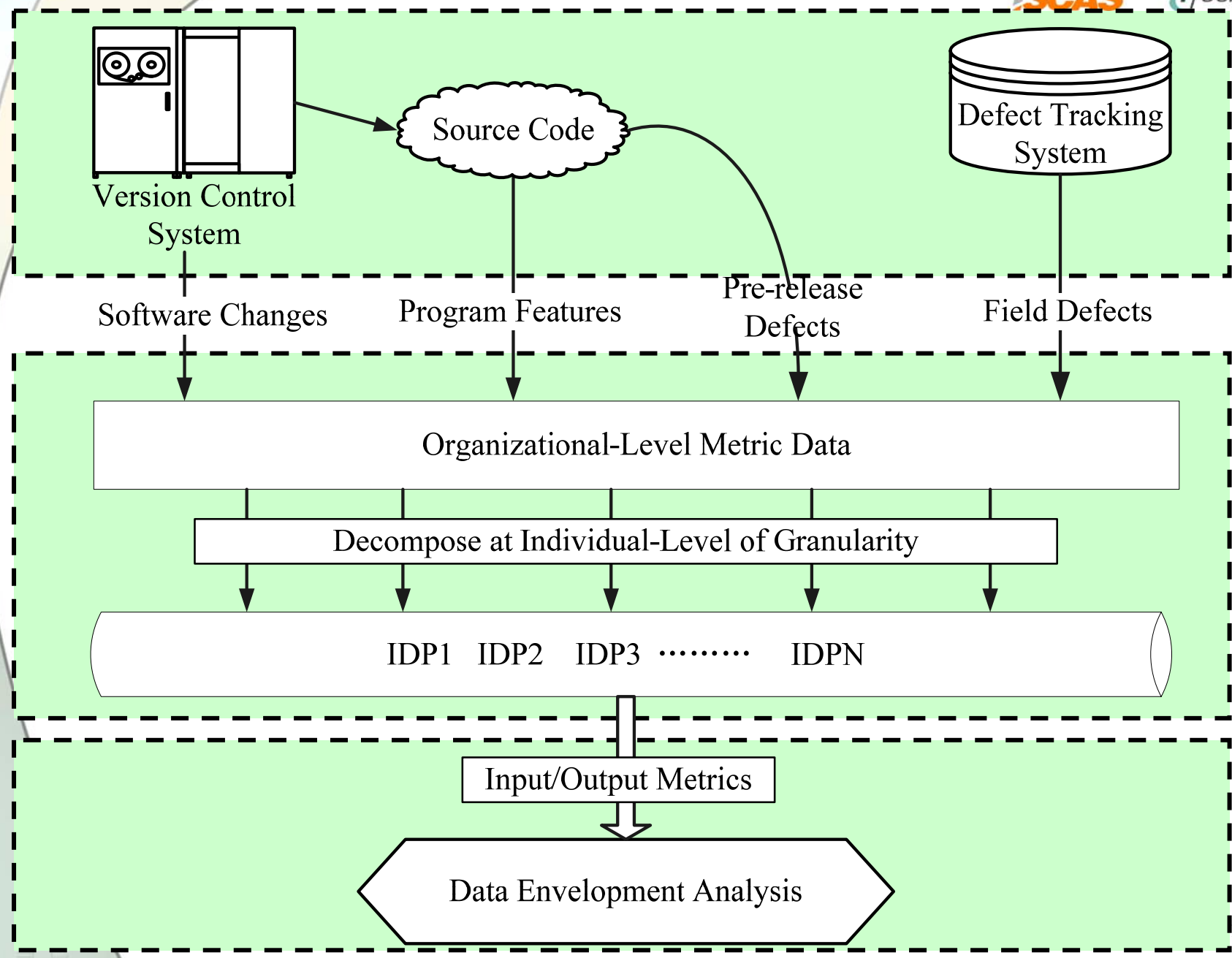
$$\text{input metrics: } x_j = (x_{1j}, x_{2j}, \dots, x_{mj})^T > 0, \quad j = 1, \dots, n$$

$$\text{output metrics: } y_j = (y_{1j}, y_{2j}, \dots, y_{sj})^T > 0, \quad j = 1, \dots, n$$

$$\text{DEA basic Model} = \left\{ \begin{array}{l} \max \frac{\mu^T y_{j0}}{\omega^T x_{j0}} = \theta_{j0} \\ \frac{\mu^T y_j}{\omega^T x_j} \leq 1, \quad j = 1, 2, \dots, n, \\ \mu \geq 0, \omega \geq 0, \mu \neq 0, \omega \neq 0. \end{array} \right.$$

# Agenda

- Introduction
- Related Work
- *Approach Overview*
  - Metric Extraction Method
  - Capability Assessment Method
- Experimental Results and Analysis
- Conclusions



# Characteristics

- Extract data from the following information sources to return four groups of predictors
  - Program features mined from source code
  - Software changes recorded in version control system
  - Pre-release defects exposed by static analysis tool
  - Field defects found in defect tracking system
- Adopt a decomposition strategy to decompose the organizational-level metrics to the individual-level
- Propose the DEA model to deal with the multi-input-multi-output capability assessment of individual development process

# Agenda

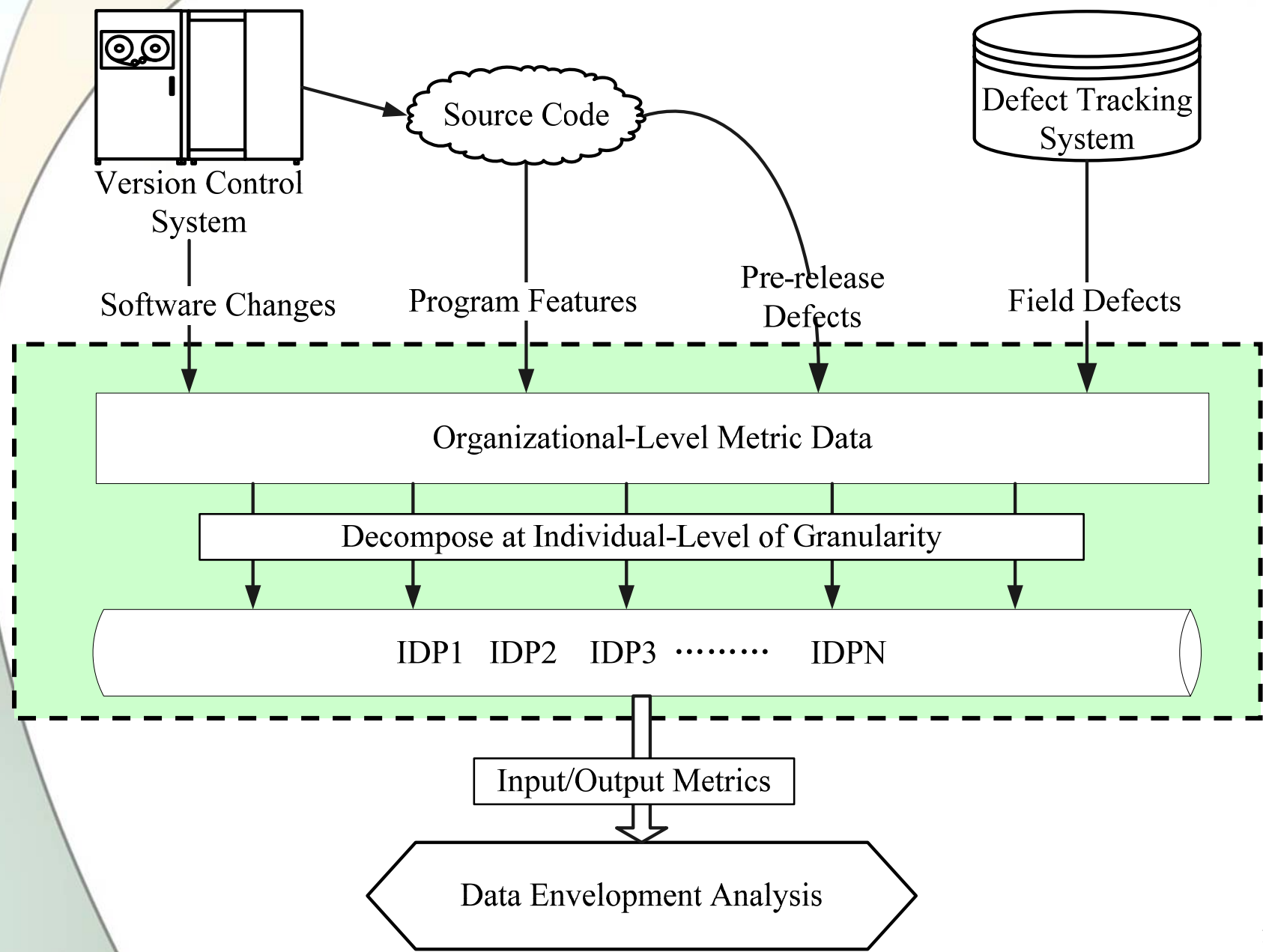
- Introduction
- Related Work
- Approach Overview
  - *Metric Extraction Method*
  - Capability Assessment Method
- Experimental Results and Analysis
- Conclusions

# Four Groups of Predictors

- Product metrics
  - Predictors-1: come from parsing the source code directly, such as the metrics related the number of distinct operations and statements
  - Predictors-2: come from the output of static analysis on the source file, such as the buffer overflows
- Development metrics
  - Predictors-3: extracted from version control system, such as average number of lines of code added or deleted per day
  - Predictors-4: come from defect tracking systems, such as number of defect records with high severity level

## Individual-Level of Granularity

- Product and development metrics are applicable to the process assessment at the organizational level
- The data of the product and process metrics can be attributed to all of the people involved in the development work
- Organizational-level metrics are decomposed to individual-level

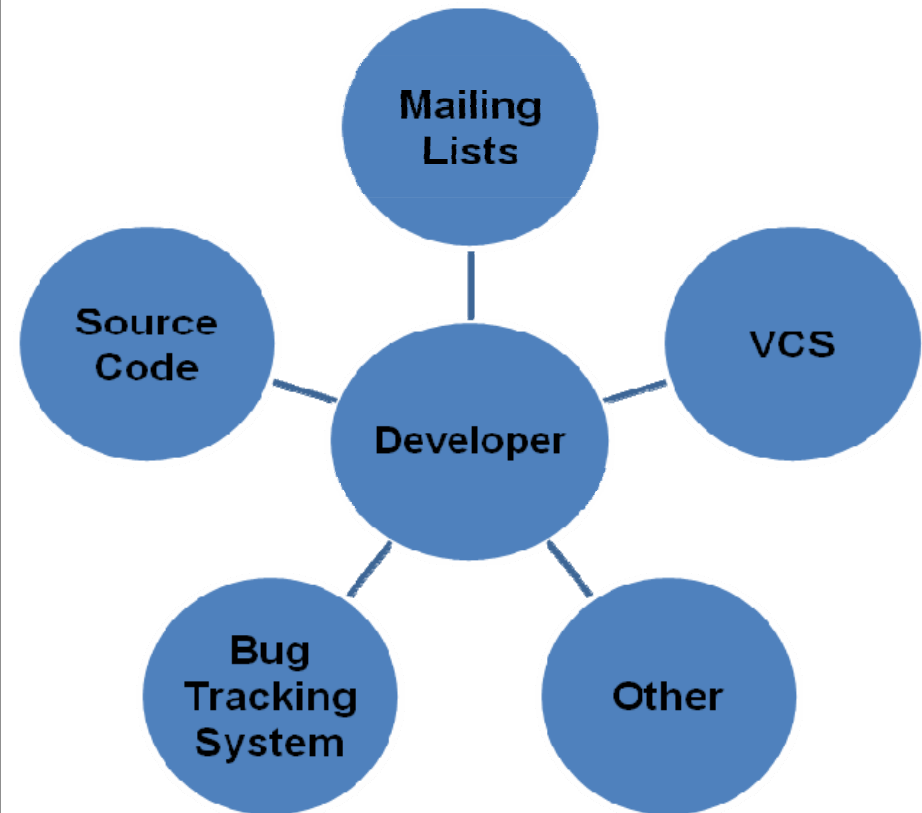


# Decomposition strategy

- Each person can appear with many identities in various repositories
  - real life names appear in copyright notices
  - user account with associated e-mail addresses
- Identifying the developers' identities from various sources
- Match each piece of metric data to certain IDP by referring to these user identities

# Example of the Identities of the Developer

| Data Source         | Primary Identities   |
|---------------------|----------------------|
| Mailing lists       | username@example.com |
| Mailing lists       | Name Surname         |
| Source Code         | Name Surname         |
| Source Code         | username@example.com |
| Source Code         | \$id: username\$     |
| Versioning System   | Username             |
| Bug Tracking System | username@example.com |



# Example of PMD Outputs

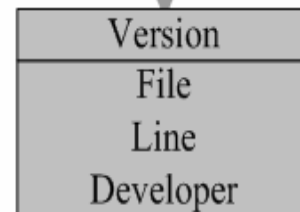
```
<?xml version="1.0" encoding="UTF-8"?>
<pmd>
  <file name="c:\jeditshell\jcmd\JcmdProcess.java">
    <violation line="43" rule="UnusedPrivateField"
      ruleset="Unused Code Rules" priority="3">
      Avoid unused private fields such as 'CTRL_C'
    </violation>
    <violation line="53" rule="CallSuperInConstructor"
      ruleset="Controversial Rules" priority="3">
      It is a good practice to call super() in a constructor
    </violation>
  </file>
  <file name="c:\jeditshell\jcmd\JcmdProcessTest.java">
    <violation line="33" rule="AtLeastOneConstructor"
      ruleset="Controversial Rules" priority="3">
      Each class should declare at least one constructor
    </violation>
    <violation line="50" rule="LocalVariableCouldBeFinal"
      ruleset="Optimization Rules" priority="3">
      Local variable could be declared final
    </violation>
  </file>
</pmd>
```

File      Lines Number      Error Category

<http://pmd.sourceforge.net/>

# Combination Process

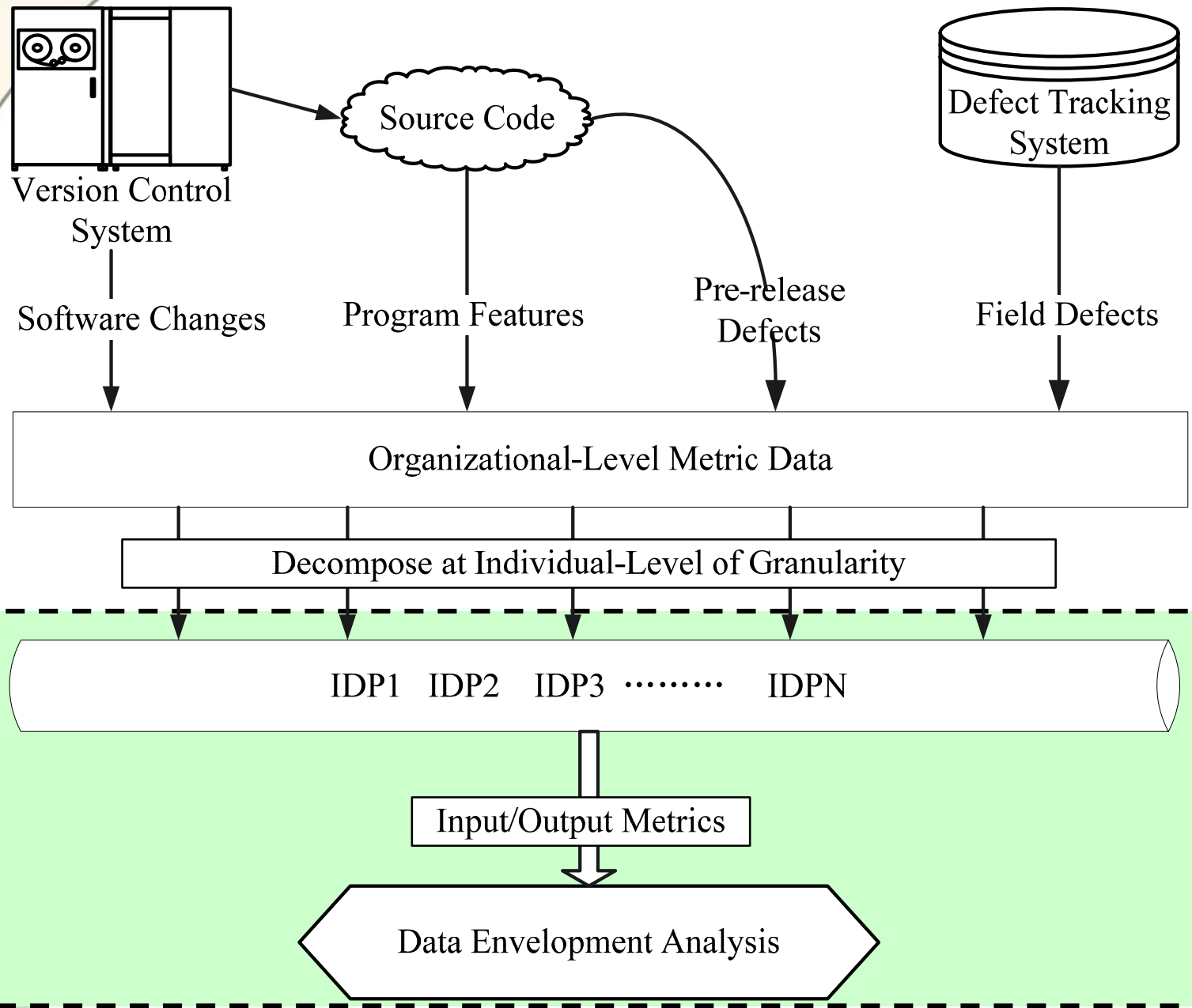
```
<pmd >
  <file name="c:\jeditshell\jcmd\JcmdProcess.java">
    <violation line="43" rule="UnusedPrivateField"
      ruleset="Unused Code Rules" priority="3">
      Avoid unused private fields such as 'CTRL_C'
    </violation>
  </file>
</pmd>
```



```
1.1 (ezust 15-Jun-06): public class JcmdProcess extends Process
1.1 (ezust 15-Jun-06): {
1.1 (ezust 15-Jun-06):   private String jcmdDirectory;
1.1 (ezust 15-Jun-06):   private Process jcmdProcess = null;
1.1 (ezust 15-Jun-06):   private Integer returnValue = null;
1.1 (ezust 15-Jun-06):   static private final int CTRL_C = 3;
```

# Agenda

- Introduction
- Related Work
- Approach Overview
  - Metric Extraction Method
  - *Capability Assessment Method*
- Experimental Results and Analysis
- Conclusions



# MIMO Analysis

- IDP is a MIMO process
  - Input can be cost, effort, ...
  - Output can be program size, defect density, ...
- m different inputs to produce s different outputs

$$\text{input metrics : } x_j = (x_{1j}, x_{2j}, \dots, x_{mj})^T > 0, \quad j = 1, \dots, n$$

$$\text{output metrics : } y_j = (y_{1j}, y_{2j}, \dots, y_{sj})^T > 0, \quad j = 1, \dots, n$$

$$(D_{BC^2}) = \begin{cases} \min(\theta) & \text{--- } [0,1] \\ \sum_{j=1}^n X_{ij} \lambda_j \leq \theta X_{i0} \\ \sum_{j=1}^n Y_{kj} \lambda_j \geq Y_{k0} \\ \sum_{j=1}^n \lambda_j = 1 \quad \lambda_j \geq 0, j = 1, \dots, n \end{cases}$$

- Capability score  $\theta$  range from 0 to 1.
- Peer weight  $\lambda$   $IDP_{j0} = \lambda_1 IDP_1 + \dots + \lambda_j IDP_j + \dots + \lambda_n IDP_n$

# Agenda

- Introduction
- Related Work
- Approach Overview
  - Metric Extraction Method
  - Capability Assessment Method
- *Experimental Results and Analysis*
- Conclusions

# Input and output assessment indicators of IDPs

| Indicator             | Type   | Meaning   | Unit          |
|-----------------------|--------|---|---------------|
| <i>Total Schedule</i> | Input  | The sum of actual time devoted to individual software process $i$   | Person hour   |
| <i>Cost</i>           | Input  | Total money invested in individual software process $i$ during the development period                       | Chinese yuan  |
| <i>Scale</i>          | Output | Total source lines of code produced by individual software process $i$                                      | LOC           |
| <i>Pre-Defect</i>     | Output | Total static errors found across all files in individual software process $i$                               | Pre-defects   |
| <i>Field Defect</i>   | Output | Total defect records found in individual software process $i$ during the development period in all releases | field defects |

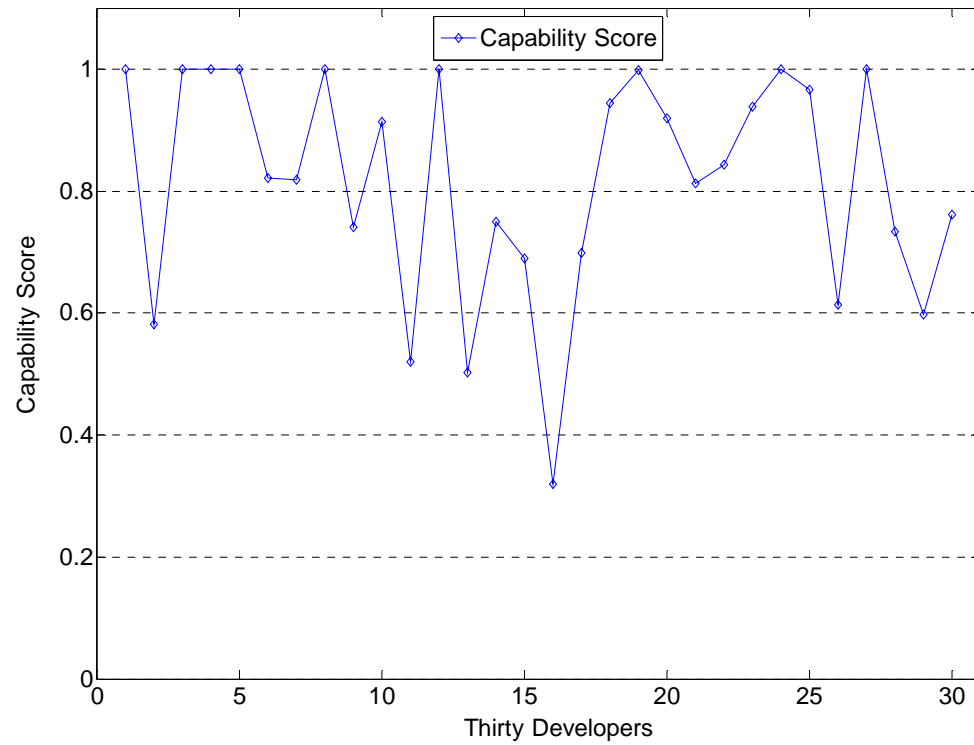
# Statistics for the 30 IDPs'



## input/output metrics

| IDP | TS    | Cost   | Scale | Pre-Defect | Field Defect | IDP | TS   | Cost  | Scale | Pre-Defect | Field Defect |
|-----|-------|--------|-------|------------|--------------|-----|------|-------|-------|------------|--------------|
| 1   | 9130  | 291561 | 79248 | 1156       | 18           | 16  | 1548 | 43670 | 6118  | 6746       | 163          |
| 2   | 595   | 17503  | 5061  | 2460       | 87           | 17  | 585  | 23433 | 6352  | 6600       | 67           |
| 3   | 12054 | 454094 | 83144 | 2230       | 42           | 18  | 748  | 28891 | 10891 | 2863       | 95           |
| 4   | 1737  | 47061  | 24020 | 6517       | 82           | 19  | 662  | 18457 | 4418  | 7414       | 176          |
| 5   | 6137  | 166813 | 72578 | 3053       | 54           | 20  | 395  | 12209 | 4749  | 7487       | 164          |
| 6   | 3855  | 112353 | 34593 | 5943       | 131          | 21  | 2333 | 80532 | 25938 | 5495       | 86           |
| 7   | 575   | 19617  | 7177  | 4532       | 131          | 22  | 635  | 23904 | 7632  | 7512       | 132          |
| 8   | 1903  | 56987  | 12964 | 8311       | 188          | 23  | 1902 | 55517 | 10696 | 6555       | 187          |
| 9   | 861   | 34423  | 9451  | 7541       | 133          | 24  | 352  | 10252 | 5228  | 6548       | 114          |
| 10  | 310   | 11179  | 4059  | 6257       | 153          | 25  | 428  | 17498 | 4476  | 5555       | 179          |
| 11  | 908   | 28278  | 6318  | 7041       | 168.         | 26  | 2005 | 68137 | 15677 | 6560       | 153          |
| 12  | 250   | 10219  | 3972  | 7162       | 178.         | 27  | 406  | 14479 | 6285  | 7556       | 157          |
| 13  | 683   | 27526  | 5379  | 6470       | 119          | 28  | 675  | 20640 | 6445  | 7561       | 143.         |
| 14  | 6245  | 250275 | 33206 | 6428       | 143          | 29  | 1167 | 46381 | 8708  | 7548       | 148.         |
| 15  | 495   | 19451  | 5338  | 7336       | 96           | 30  | 535  | 21229 | 5203  | 8545       | 158          |

# Capability Score $\theta$



# Agenda

- Introduction
- Related Work
- Approach Overview
  - Metric Extraction Method
  - Capability Assessment Method
- Experimental Results and Analysis
- *Conclusions*

# Conclusions

- Present a novel combined approach for capability assessment of individual software processes and practices
- Facilitate the process metrics extraction task by making use of software repositories as the data source, enabling recording of data effortless and more accurate
- The DEA model has been adopted to measure the capability of individual software processes under MIMO constraint
- Empirical illustrations from a practical case study illustrate the effectiveness of the proposed methodology

# Thank You!