Agenda

- PIVoT - Why & What
- Learnings
- Trends
Some observations…

- Process metrics like Cost performance does not translate to quality
- Training on development techniques may not lead to widespread adoption
- Insights into an individual’s importance to a project were non-trivial
- Usage of tools does not automatically lead to increase in quality
- Process Warnings: It’s simply too late to fix!
Motivation behind our work...

Top-down delivery process metrics and monitoring

- Process monitoring (e.g., CMMI) standardized measurement and monitoring of software delivery in a largely technology-agnostic manner.
- Process monitoring does not always provide ‘right warnings at the right time’.
- On the other hand, data generated by software development activities like version control actions, tool executions etc. provide fine-grained and “in-process” view of development.
- Insights can be generated near real-time, allowing proactive intervention.

Bottom-up comprehension of software development activities
PIVoT framework

Framework principles

• Insights are based on on-ground activity data
• Data should be collected at source of activities, without human intervention
• Data collection should be non-invasive

Insights design approach: Outcome-oriented design

1. Determine intended outcome
2. Define insight
3. Determine visualization
4. Select and configure data sources
5. Enumerate data required for analysis
6. Design data analysis

PIVoT* allows for in-process, automated and non-invasive collection and analysis of project environment data and presents a set of actionable insights and metrics to visualize the trajectory of the project.
PIVoT project model & insights

[Diagram of PIVoT project model]

- Code Quality
  - Code Quality Index: CQI
  - Top Code Quality Issues and Categories
  - Rate of Increase of Code Quality Issues

- Component Testing
  - Quality of Component Testing: Effort (QCTE)
  - UnTested and Poorly Tested Components
  - Coverage for Complex Components

- Development Efficiency
  - Cost of Development Index: CDI
  - Per package Code Churn trends
  - Top code contributors

- Code Churn
  - Per package code contributors

- Team Analysis
  - Highly Connected Resources
  - Poorly Connected and Disconnected Resources
  - Programmer Fragmentation

- Composite Reports
  - Code churn and rate of increase of quality issues
  - Code churn and QCTE
  - Code churn and CDI

[Graphs and charts related to code churn and quality issues]
Key learnings

Positive behavioral change can be effected by a well-designed insight.

Metrics and insights need to be intuitive defined to for project managers to adopt and take action on them.

Integrating the insights obtained through development data and associating it with the Agile release / sprint plans and roles helps manage feature velocity and technical debt.

Planned roles change and are important to understand team dynamics. The changes are visible only through objective on-ground data.

Data-driven approach to measuring micro-process adoption provides fine-grained insights for methodology adoption.

Balancing between insights and privacy is a generally important consideration for any tool that uses data or meta-data to identify patterns and suggests recommendations.
Way forward – trends impacting further research in software development analytics

<table>
<thead>
<tr>
<th>Crowd sourced Development</th>
<th>Development in the cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Large-scale distribution</strong> of developers and resulting ineffectiveness of traditional process monitoring approaches.</td>
<td>A big advantage of development in the cloud, is the <strong>automatic standardization</strong> of IDEs, tools etc.</td>
</tr>
<tr>
<td>For a client-critical software project, it is essential to know how the ‘crowd’ developers are adopting key practices such as frequent unit testing, code quality analysis etc.</td>
<td>The <strong>consistent data collection</strong> challenge can be mitigated by using cloud-based IDEs.</td>
</tr>
<tr>
<td><strong>Standardization</strong> of the development environment used by crowd-developers and data collection is a challenge</td>
<td>Cloud IDEs provide an easy way to capture all development activities leading to creation of better project models, better activity correlation and <strong>more precise insights</strong>.</td>
</tr>
<tr>
<td>Modernization of <strong>distributed awareness</strong> techniques to handle new configuration management systems and techniques, large-scale distribution, and offering better visualization and actions to developers is required</td>
<td></td>
</tr>
</tbody>
</table>
## Way forward – trends impacting further research in software development analytics

### Paradigm of IoT

Usage of the sensors and actuators IoT paradigm to software development leads to interesting new ways software development analytics.

“Software actuators” embedded in code can remove subjectivity in actions taken based on certain insights.

**Actuator design**, **DSLs for rapid development of Actuators**, **testing of automated actions** are different areas that need to mature to make full use of the IoT paradigm.

### Emergence of Cognitive

Maintaining relevance of insights and **prescribing contextual actions** is key for long-term adoption of a software development analytics framework.

**Learning from human actions** and refining metric thresholds and actions is important.

Incremental and **adaptive learning** of faults and causality.
Summary

Significant changes in the environments in which insights are generated (Crowd and Development in Cloud), new insights required for managing ever-increasing distributed developer workforce (distributed team awareness and coordination), insights driving intelligent software (IoT inspired ‘software actuators’), and finally insights frameworks that leverage cognitive techniques for inferencing patterns and learning are the basis for future research in software development analytics.