accenture technology

Accenture Technology Labs

Innovation is...being first on the wave

Software Development Analytics: Our Experiences and the Way Forward

Vikrant Kaulgud and Vibhu S. Sharma ☺

High performance. Delivered.

Strategy | Consulting | Digital | Technology | Operations

Agenda

- PIVoT - Why & What
- Learnings
- Trends

Some observations…

Process metrics like Cost performance does not translate to quality

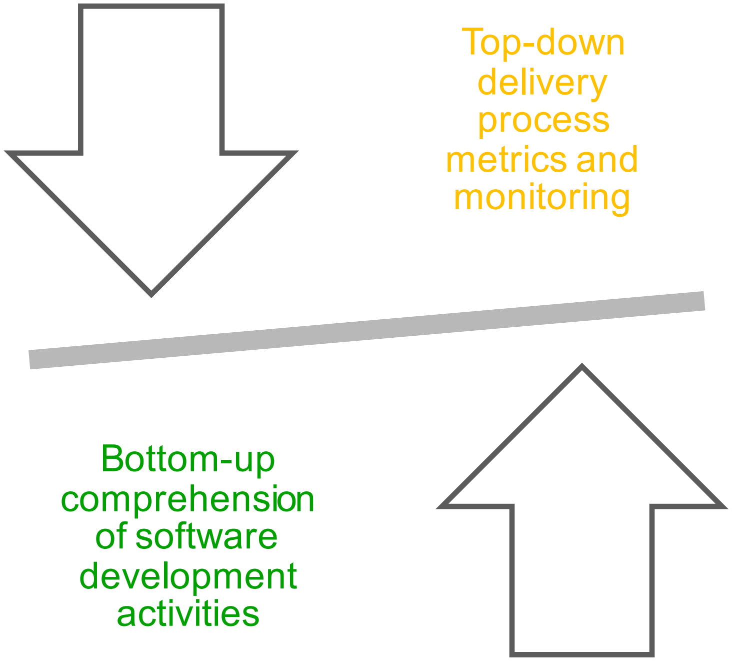
Training on development techniques may not lead to widespread adoption

Insights in to 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…

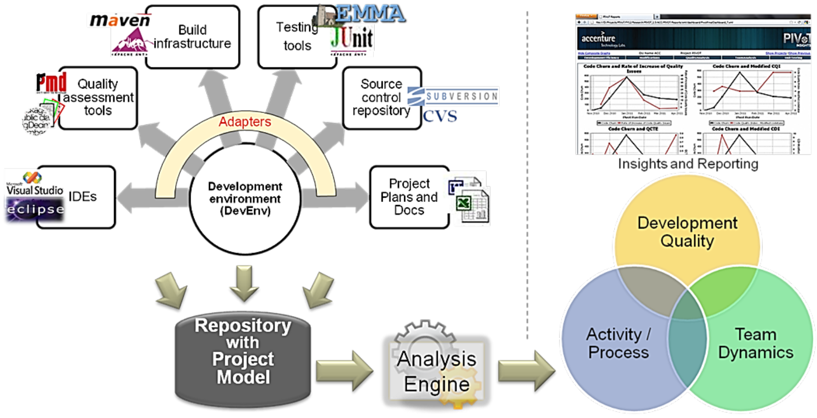


Top-down delivery process metrics and monitoring

Bottom-up comprehension of software development activities

- 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.

PIVoT framework

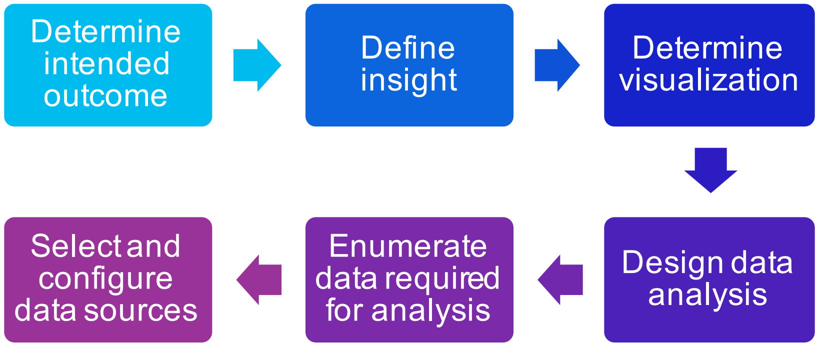


PIVoT* allows for in-process, automated and non-invasive collection and analysis of project environment data and presents a set of actionable insights and metrics to visualize the trajectory of the project.

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



Determine intended outcome

Define insight

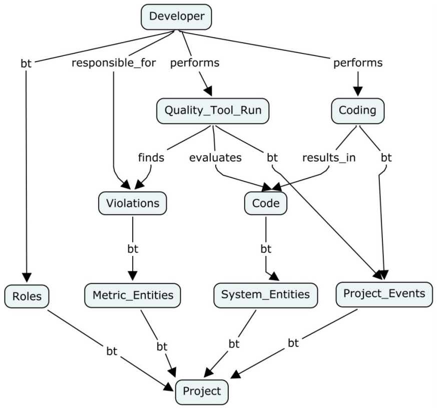
Determine visualization

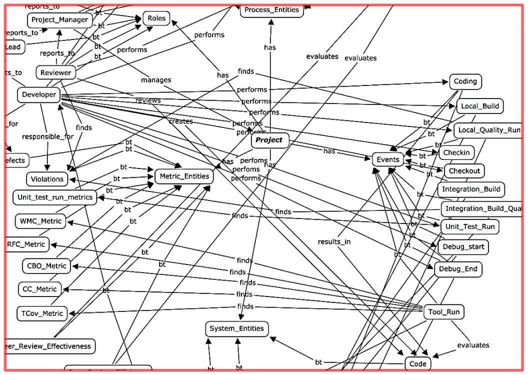
Design data analysis

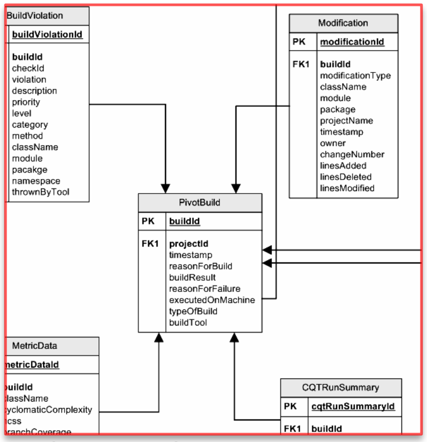
Enumerate data required for analysis

Select and configure data sources

PIVoT project model & insights







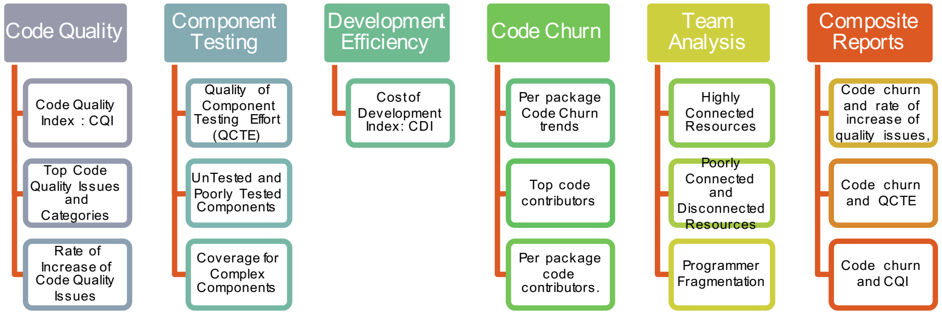
BuildViolation
buildViolationId
buildId checkId violation description priority level category method className module pacakge namespace thrownByTool

Modification	
PK	modificationId
FK1	buildId modificationType className module package projectName timestamp owner changeNumber linesAdded linesDeleted linesModified

PivotBuild	
PK	buildId
FK1	projectId timestamp reasonForBuild buildResult reasonForFailure executedOnMachine typeOfBuild buildTool

MetricData
netricDataId
uildId lassName yclomaticComplexity css ranchCoverage

CQTRunSummary	
PK	cqtRunSummaryId
FK1	buildId



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

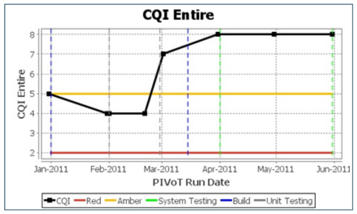
Component Testing
- Quality of Component Testing Effort (QCTE)
- UnTested and Poorly Tested Components
- Coverage for Complex Components

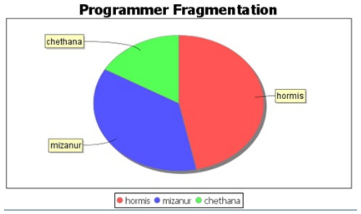
Development Efficiency
- Costof Development Index: CDI

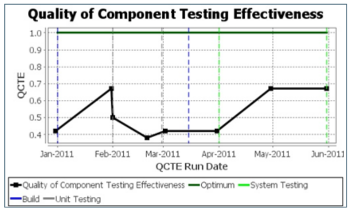
Code Churn
- Per package Code Churn trends
- Top code contributors
- Per package code contributors.

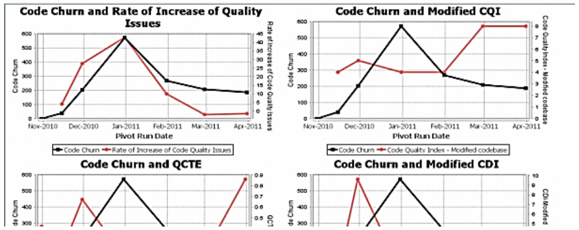
Team Analysis
- Highly Connected Resources
- Poorly Connected and Disconnected Resources
- Programmer Fragmentation

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









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

Crowd sourced Development

Large-scale distribution of developers and resulting ineffectiveness of traditional process monitoring approaches.

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.

Standardization of the development environment used by crowd-developers and data collection is a challenge

Modernization of distributed awareness techniques to handle new configuration management systems and techniques, large-scale distribution, and offering better visualization and actions to developers is required

Development in the cloud

A big advantage of development in the cloud, is the automatic standardization of IDEs, tools etc.

The consistent data collection challenge can be mitigated by using cloud-based IDEs.

Cloud IDEs provide an easy way to capture all development activities leading to creation of better project models, better activity correlation and more precise insights.

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