CODE QUALITY ANALYTICS

CodeVox® - Listen to voice of your code to unlock untapped potential

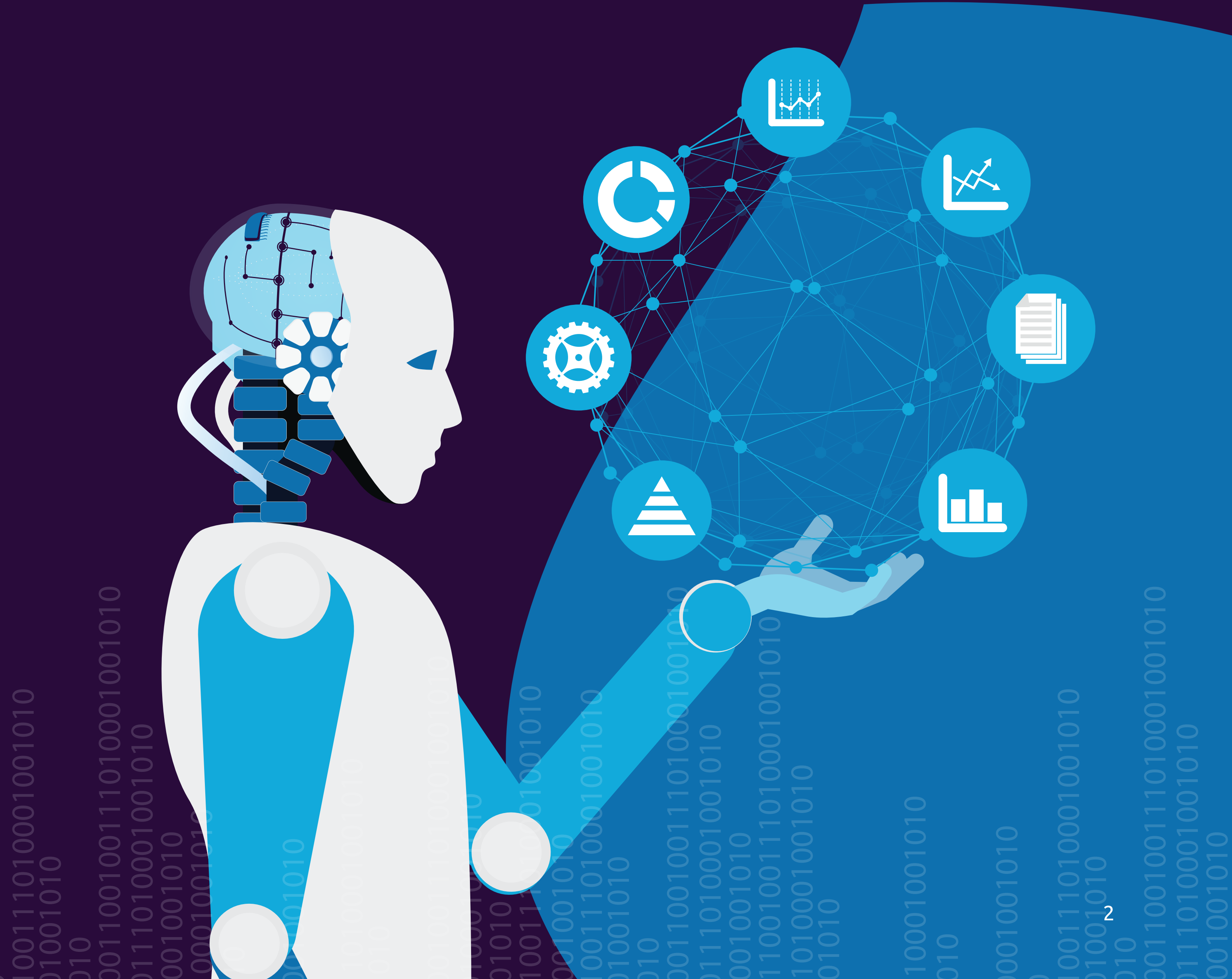
Overview

Ensuring and maintaining code quality with a rapidly growing software team is a huge challenge. Code quality improvement requires a well-developed strategy and adhering to it throughout the project's lifecycle. If left unaddressed, it will be only a matter of time before it causes problems and leads to great financial losses. Many software projects struggle to move forward because the code base becomes unstable and difficult to develop further. Technical debt is a term in software development that implies additional rework caused by poor source of code quality. Ignoring the periodical maintenance and refactoring of code can create huge technical debt during long projects.

Maintaining high code quality is necessary for:

- **Robust software**
- **Sustainable program**
- **Easy to maintain and increased readability**
- **Lower technical debt.**

Code quality is subjective, and it varies from team to team. This point of view analyzes various code quality practices and aspects of applying analytics and machine learning techniques to improve software code quality.



Code quality analysis techniques

Most analysis tools operate on source code. The quality of source code is a key factor for a software development and its continuous monitoring is an indispensable task in the project. A set of practices has been followed by various organizations for the last four decades. The methodologies have not changed but are improved by continuous learning and adapting various tools and technologies. Figure 1 depicts the code quality analysis techniques that are widely adopted by most of the organizations. The proactive techniques that are carried out before each commit, i.e. thorough checking of software code before it changes into a source code repository are grouped as pre-commit code quality techniques. The techniques that are used after the commit are post-commit code quality techniques.

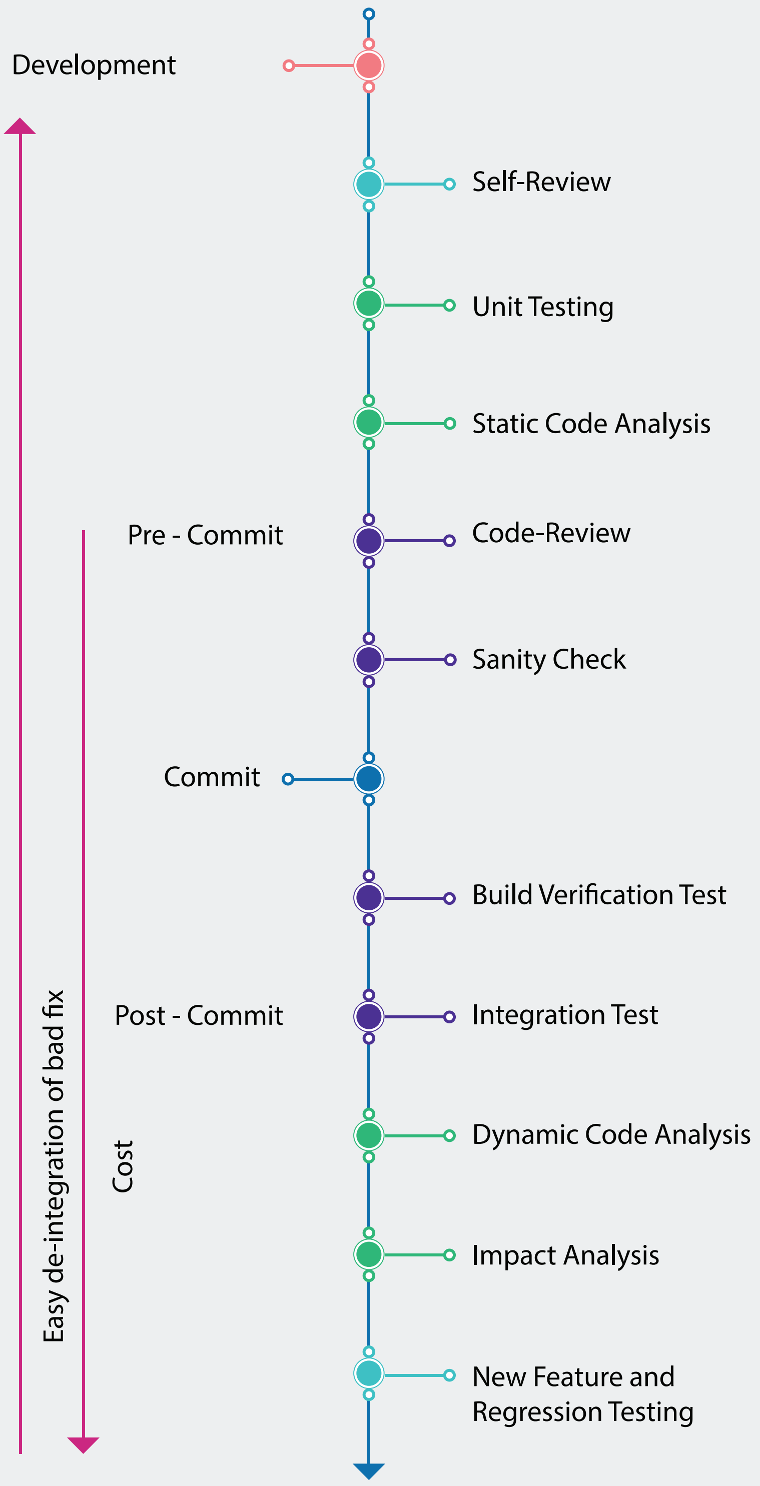


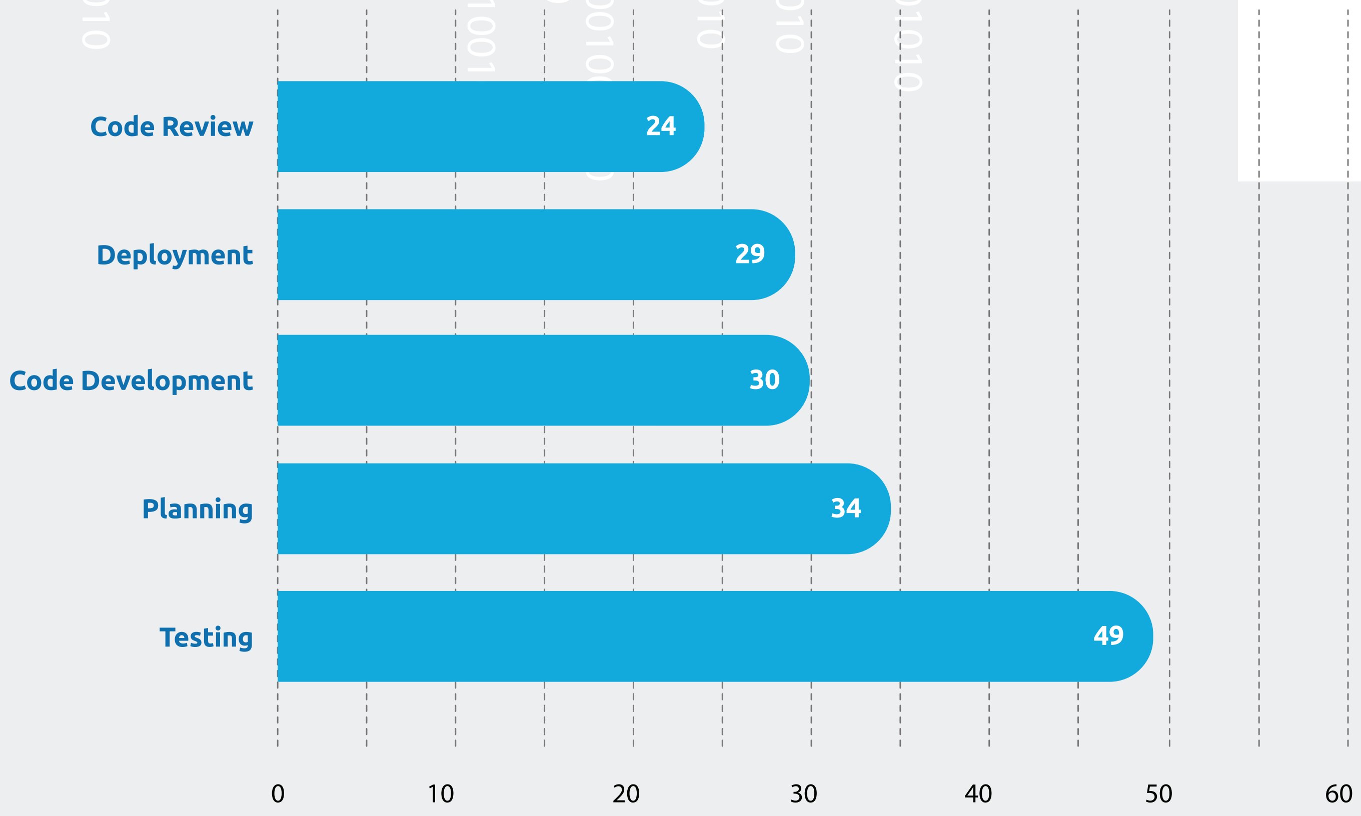
Figure 1 – Code Quality Analysis Techniques

- **Unit Testing**
- **Code Reviews**
- **Defect prevention techniques, check list**
- **Static Code Analysis**
- **Dynamic Code Analysis**
- **Impact Analysis**
- **QA Testing**

Shift-left initiatives have helped organizations to invest and benefit more on pre-commit code quality techniques such as unit testing, code reviews, use of static code analysis data, defect prevention check lists, and more. However, a recent GitLab survey that targeted developers and engineers found that testing is responsible for more delays than any other part of the development process.

Activities such as manual impact analysis, identifying the right set of test cases to execute, inadequate code reviews, and unit testing, and the non-deterministic nature of system are the key reasons for the testing delay. Testing is an important activity that ensures code quality, but the results are deferred till the end of the cycle.

Graph 1 - Most delays encountered in the development process



Source: GitLab – 2019 Global Developer Report

A summary of various aspect of code quality techniques

Methodology	Stage	Tools and Aids	Efforts	Challenges	Opportunities
Unit Testing	Pre-commit	Unit testing tools			Auto-generation of unit test
Code Reviews	Pre-commit	Code review check lists, Automated code review tools like DeepCode	Extensive	Availability of SME bandwidth	
Static Code Analysis	Pre-commit	SCA tools	Manual review of SCA reports, and corrective actions	Manual review of SCA reports, Stand alone data	Analytics on time series data
Build Varification Test	Pre-commit		Low	Unimpactful, Static set of test	Dynamic build verification test
Integration Test	Pre-commit				
Dynamic Code Analysis	Pre-commit	DCA tools	Extensive	Unimpactful with long duration	Automated workflow, to use more often
Impact Analysis	Pre-commit		Extensive	Manual, effort intensive, person dependent, incomplete due to unavailability of bandwidth	Automated impact analysis
QA Testing	Pre-commit			Test selection and prioritization	Automated test selection and prioritization

Opportunities to overcome the code quality technique challenges

In a nutshell, there's a lot room for improvements to overcome current limitations, particularly:

- Decisions based on sampling of data – often the lack of band width and tools results in incomplete analysis
- In the absence of collective analysis approach, decision making on few inputs can be misleading
- Power of time, series of data – currently, trends and patterns are not mined or underutilized
- Code smells are assumed to indicate bad design that leads to poor code quality; code and design smells play a key role during impact analysis
- Dynamic code analysis (DCA) tools create a huge amount of data which may also contains a good amount of noise. Muting the noise and mining meaningful information out of DCA output is a challenge
- SME knowledge is limited to specific areas; it is distributed among multiple SMEs and is inconsistent
- Need for a system to provide quick impact analysis recommendations
- Untapped benefits of mining software repositories
- Inadequate utilization of the computing power, emerging data analytics, and machine learning advancements.

Data sources around source code

Data is the new oil, a huge amount of data is being generated, stored, and maintained over years within the software code and around it. Following are the type of the data in and around source code:

• Behavioral data (commit history)
• Attitudinal data (code, design review)
• Interaction data (dynamic code analysis, code check in comments, defect notes)
• Descriptive data (code quality attributes like defect density, etc., self-declared information).

Figure 2 – Data sources, a goldmine

Data Sources

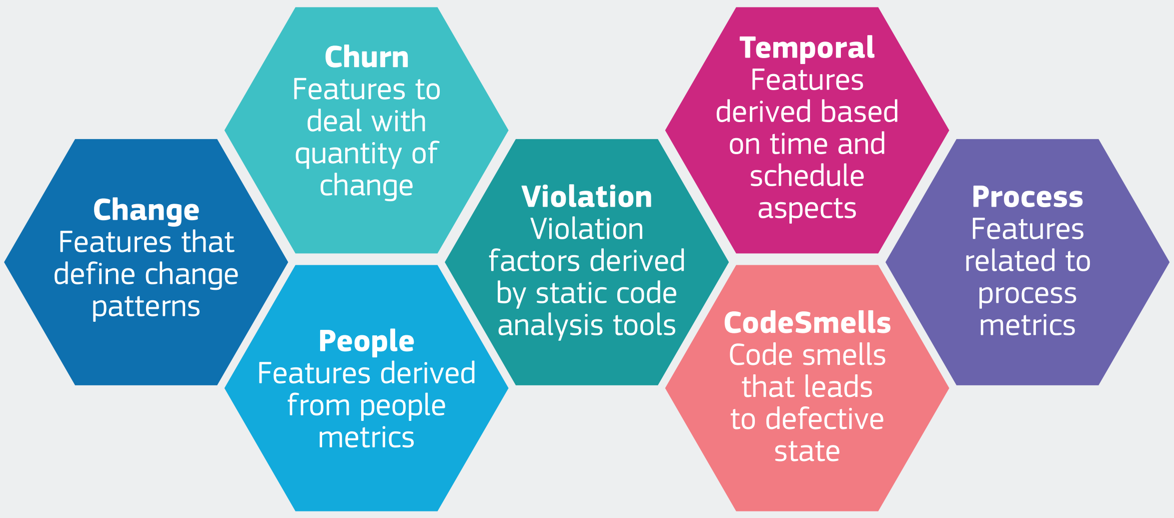
Code Smells	Source Code Content
Defect Data	Dynamic Code Analysis Data
Review Comments	Transaction Data
Developer	Static Code Analysis Data
Build Management System	Requirement, Use Cases

Unit Test Results

Figure 2 represents a possible set of data sources around source code that represent an untapped gold mine of insights. Most of the code quality techniques look at the content of the data for deriving insights. Content-specific quality analysis is programming-language specific, person dependent, time intensive, and derives a very limited insight. Along with content, other non-content aspects derived from meta data around code help get the complete picture of code quality. For example, commit logs, a behavioral data of source code helps to derive insights such as change patterns, developer acquaintance to the changed code. These features (characteristics or properties are referred by features or input variables in data science) are simple and capable of determining the quality.

Existing techniques use any of the data sources in isolation. Factoring features referred as code quality features derived from multiple repositories that provides recommendations with higher accuracy. Code quality features can be categorized into following categories:

Figure 3 – Code Quality Features


Churn
Features to deal with quantity of change
Change
Features that define change patterns
People
Features derived from people metrics
Violation
Violation factors derived by static code analysis tools
Temporal
Features derived based on time and schedule aspects
CodeSmells
Code smells that leads to defective state
Process
Features related to process metrics

Analytics and machine learning for code quality

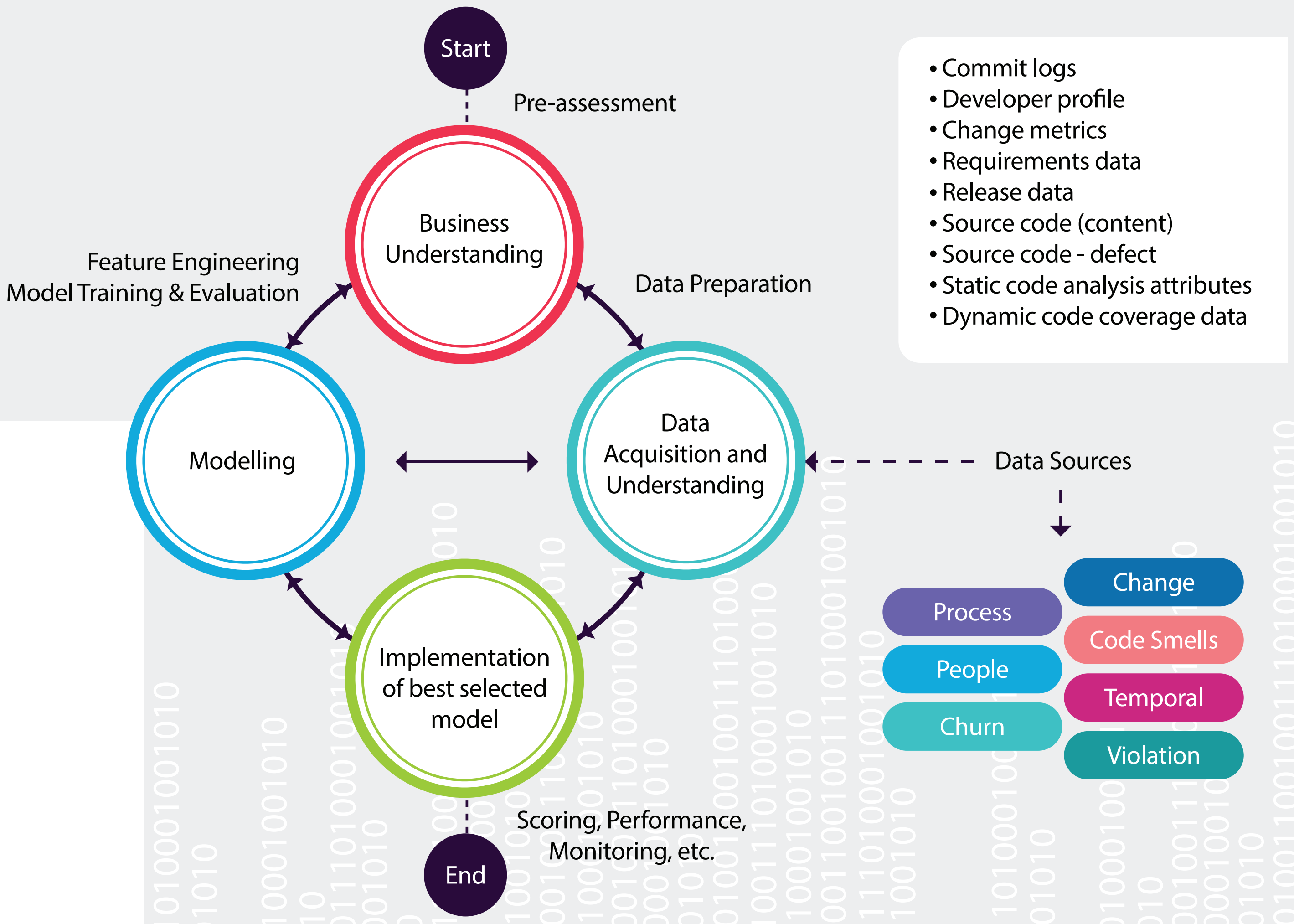
An analytics and machine learning based system applies algorithms to data and uses machine learning models to make intelligent decisions automatically based on discovered relationships, patterns, and knowledge from data. Algorithms such as regression, classification, clustering are used to extract knowledge, patterns, and relationships. Mining prior experience helps to identify dominant patterns, these learnings or patterns can be used as predictors for a future outcome.

In the absence of an automated system, the architects, subject matter experts (SMEs) might be using a set of heuristics, mental shortcuts manually to assess code quality. These heuristics can be mined via a heuristics mining framework and can be automated via an analytical model. The idea is not to replace the SME intelligence but to augment and accelerate SME decision making.

CodeVox®

Capgemini's CodeVox solution listens to the voice of your code and related artifacts. Leveraging analytics and machine learning CodeVox:

- Factors data from multiple repositories such as source code, code reviews, project management, defect management, and static code analysis
- Establish the linked data across multi-silo repositories
- Extract patterns from historical data
- Derive 50+ change, churn, temporal, code smells, violation, and people metrics which are unique value propositions of CodeVox
- Implement self-learning riskier release code/component predictor model using analytics and machine learning
- CodeVox model can be invoked on-demand or scheduled to get the real-time recommendations on riskier components and files.



The above picture illustrates a high-level overview of CodeVox modelling. To know more about CodeVox reach out to us.

Conclusion

Code quality improvements can be achieved by applying analytics and machine learning on top of the content and meta data of code along with data coming from other related repositories. Recommendations from mining only content have provided very little accuracy (between 20–30%) and recommendation based on code, meta data, and other related repositories achieved 70–90% accuracy. In general, applying analytics and machine learning on code and code-related artifacts helps organizations to achieve quicker releases and better code quality. The recommendations coming from automated impact analysis increase the code quality with following benefits:

- Developing and running small unit tests for the risky components/files
- Multi-layer code reviews done for hot files
- Dynamic build verification tests
- QA teams focus test on the high-risk areas
- Corresponding test cases execution cadence have increased for hot files
- Improved quality with early deduction of defects.

Recommendations coming from analytics and machine learning models augment teams to take quicker decisions and they are not a replacement for human intelligence. Frequent auditing of effectiveness of the models with statistical parameters such as precision, recall, F-measure and enhancing models with missing heuristics will improve the correctness of the models.

Reference

1. GitLab – 2019 Global Developer Report, https://about.gitlab.com/developer-survey/
2. World Quality Report 2019, https://www.capgemini.com/in-en/research/world-quality-report-2019/

Authors

Vivek Jaykrishanan
Sr. Director, Product and System Testing
Digital Engineering and Manufacturing Services

Jagadeesh Venugopal
Principal Architect, Product and System Testing
Digital Engineering and Manufacturing Services

Capgemini

About Capgemini

Capgemini is a global leader in consulting, digital transformation, technology and engineering services. The Group is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year+ heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. Today, it is a multicultural company of 270,000 team members in almost 50 countries. With Altran, the Group reported 2019 combined revenues of €17billion.

Visit us at

www.capgemini.com

GET THE FUTURE YOU WANT

About Capgemini's Digital Engineering and Manufacturing Services

Capgemini's Digital Engineering and Manufacturing Services brings together deep domain expertise to lead the convergence of Physical and Digital worlds through technology, engineering and manufacturing expertise to boost our clients' competitiveness. A recognized leader with over 10,000 engineers across the globe and 30+ years of experience, Capgemini's comprehensive portfolio of end-to-end solutions enables global companies to unlock the true potential of their product portfolios and manufacturing efficiencies.

To learn more please contact:

marketing.dems.global@capgemini.com