The Journal of Systems & Software 171 (2021) 110836

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Large scale quality transformation in hybrid development organizations – A case study

Satya Pradhan *, Venky Nanniyur

Cisco Systems, Inc. San Jose, CA, USA

ARTICLE INFO

Article history:
Received 8 September 2019
Received in revised form 9 September 2020
Accepted 18 September 2020
Available online 22 September 2020

Keywords:
Quality transformation
Quality management system
Agile
Waterfall
Hybrid development organization

ABSTRACT

As the software industry transitions to a subscription-based software-as-a-service (SaaS) model, software development companies are transforming to hybrid development organizations with increased adoption of Agile and Continuous Integration/ Continuous Delivery (CI/CD) development practices for newer products while continuing to use Waterfall methods for older products. This transformation is a huge undertaking impacting all aspects of the software development life cycle (SDLC), including the quality management system. This paper presents a case study of a large-scale transformation of a legacy quality management system to a modern system developed and implemented at Cisco Systems. The framework for this transformation is defined by six distinct areas: metrics, process, measurement, reporting, quality analytics, and culture & leadership. Our implementation leveraged recent advances in Machine Learning (ML), Artificial Intelligence (AI), connected data, integrated operations, and big data technologies to solve the challenges created by a hybrid software development organization. We believe this case study will help researchers and industry leaders understand the benefits and potential challenges of such sizeable transformations.

1. Introduction

With the advent of cloud technologies, rapid advances in connectivity, mobility, analytics, and the transition to the software-as-a-service (SaaS) model, the software industry is going through a fundamental transformation impacting many different areas such as engineering culture, software development life cycle, quality management system, and engineering infrastructure. The success of companies depends on the successful planning and implementation of these transformations. Gartner, Forbes, McKinsey, and LNS Research have done extensive market research on this topic (Pettey, 2018; Forbes Insight, 2018; Handscomb et al., 2018; Miller, 2013). However, there is minimal guidance available on the details of the framework and implementation of these large-scale transformations. This paper attempts to close this gap by providing a case-study on the transformation of the quality management system developed at Cisco Systems, Inc.

The transition of the software industry from a traditional license and maintenance model to a subscription-based software-as-a-service (SaaS) model has already begun. As predicted by Gartner, "by 2020, all new entrants and 80% of historical vendors will offer subscription-based business models" (Pettey, 2018). Organizations are switching over to the agile software development method to respond to the changing customer needs, fluctuating market demands, competitive pressures, and disruptive technologies. Forbes recently surveyed 1007 leading companies across the globe (Forbes Insight, 2018). Half of these organizations have revenue between $100 million and $1 billion; the other half generate $1 billion or more in revenue. According to the research finding, 77% of these organizations are adopting an Agile framework, Scrum in particular, to spur growth and support digital transformation (Forbes Insight, 2018). However, this transformation has not been easy to implement at scale (Handscomb et al., 2018; Miller, 2013; Boehm and Turner, 2004). In Boehm and Turner (2004), Boehm and Turner highlight the need to balance the apparently opposite attributes of agility and discipline to achieve success in any organization.

At Cisco, we have learned through our varied experiences of successes and failures that this transformation cannot be completed overnight and may take several years to accomplish. In addition, organizations need to continue running their business using the older process (waterfall), infrastructure, and organizational framework during this transformation. From an engineering operations point of view, this transformation creates a hybrid organization concurrently using older (e.g., Waterfall) and newer (e.g., Agile and Continuous Integration Continuous Delivery (CI/CD)) development methodologies in different parts of the enterprise.

The case-study presented in this paper describes the large-scale transformation of the quality management system for a

* Corresponding author.
E-mail addresses: satpradh@cisco.com (S. Pradhan), vnanniyu@cisco.com (V. Nanniyur).

https://doi.org/10.1016/j.jss.2020.110836

hybrid development organization. Section 2 summarizes the work done by other researchers on the subject of quality transformation. The following section describes the challenges and opportunities with large hybrid development organizations. Section 4 presents the details of the transformation developed and implemented at Cisco Systems. This is followed by implementation and results in Section 5. Finally, we present the "Threats to Validity" and "Summary and Future Work" in Sections 6 and 7, respectively.

2. Related work

A comprehensive software quality management system has several critical elements — development and test methodologies, metrics, quality management process and tools, and organizational culture (Chemuturi, 2011; Ebert and Duarte, 2018). Horch has presented step-by-step best practices for quality management during the development and test activities over the entire software development lifecycle (Horch, 2003). These practices have become part of the agile development framework (Knaster and Leffingwell, 2019; Leffingwell, 2007) and have been adopted by many large software companies to develop high-quality software (Forbes Insight, 2018; Handscomb et al., 2018; Miller, 2013). There has also been a paradigm shift in software testing where traditional Quality Assurance (QA) groups have transformed into Quality Engineering (QE) organizations with an emphasis on test automation, shift-left testing, and behavior-based development (King, 2019; Yackel, 2020; Nauman, 2017).

In the area of quality metrics, Kan has provided a detailed summary of metrics and models used for the traditional waterfall-based development method (Kan, 2003). Benchmarks and best-practices to assess different software systems have been developed by Jones (2000). These quality metrics have been extended to large-scale hybrid (waterfall and agile) software development organizations (Pradhan and Nanniyur, 2019; Mistrik et al., 2016).

On the tools and infrastructure, several commercial software have been developed by companies to implement the quality management process (Quality Management Software, 2020; Nir, 2020). A complete quality management system needs to provide data connectivity between business information technology (IT) and operational technology (OT) (Jacob, 2017). The IT includes information on product lifecycle management, enterprise resource, and customer support, etc. The OT includes technologies used in source code management, testing framework, defect management system, software delivery infrastructure, and customer management system, etc. Given the complexity and scale of these infrastructures, no single commercially available software can meet the needs for a complete quality management system. Most of the medium to large enterprises, including Cisco Systems, use a combination of home-grown infrastructure, commercial tools, and opensource software for the quality management system (Jacob, 2017).

Organizational culture is an equally essential aspect of a comprehensive quality management system (Jacob, 2017; Anthony et al., 2019; Cognizant White Paper, 2010). Quality ownership by cross-functional executives and top management is critical for the success of any corporate-wide quality program (Jacob, 2017). KPIs play an essential role in this effort and help organizations focus on results, not on process or people, to achieve quality objectives (Jacob, 2017; Cognizant White Paper, 2010).

To the best of our knowledge, there is no technical publication available in the open literature on a comprehensive large-scale software quality transformation. All the published studies on this topic are done by consulting farms based on surveys of company executives and managers (Jacob, 2017; Anthony et al., 2019; Cognizant White Paper, 2010). These studies clearly show that many companies are implementing quality transformation at different levels and have not yet published their results. We are the first company to share the details of our quality transformation. We believe this paper to be beneficial for researchers and industry leaders in understanding the benefits and challenges associated with such a sizeable transformation. One of our main objectives in presenting this work is to encourage other companies to share their experiences.

3. The framework for quality transformation

Cisco has a large portfolio consisting of 100+ software and hardware offerings in the area of networking switches, routers, wireless, IoT, security, collaboration, and network management. The core networking group for routing, switching, and wireless products use a hybrid of Waterfall and Agile development methodology. Security, collaboration, and network management groups primarily use Agile methodology. Most of the core products are deployed in customer premises (referred to as on-prem products), while services and solutions from the collaboration and network management groups are available in the cloud. In the past, each product used its own quality metrics and measurement tools, resulting in different customer experiences. As the company increased focus on quality as a strategic differentiator, we had to develop a single and scalable quality management system that could be used for a diverse product portfolio.

The fluctuations in market demand, changes in customer expectations, and competitive pressures are some of the essential factors creating unique challenges for the quality management system. At the same time, advances in technologies such as big data, analytics, machine learning, and automation have created unique opportunities to address these challenges. Utilizing these opportunities to transform the quality management system is the key to improve customer satisfaction. This section summarizes the challenges associated with the hybrid organization and the legacy quality management system.

3.1. Challenges with legacy quality management systems

Over the last two decades, quality management systems have significantly improved because of advances in technologies for data analysis, automation, data visualization, and reporting. However, significant gaps within the infrastructure, process, and organizational culture prevent the implementation of a full data-driven quality management system. The followings are some of the important challenges for the legacy quality management system (Jacob, 2017).

- *Poor metrics*: Every software development team uses metrics to measure and drive quality. However, in the absence of a consistent metrics program for the whole company, significant gaps in quality measurement can impact customer experience. The survey by LNS Research involving more than 500 companies reports that "37% of companies say POOR METRICS is a top roadblock to accomplishing quality objectives" (Jacob, 2017).
- *Fragmented data and systems*: Many of the tools over last two decades were developed to provide 24 × 7 online access to data. This has resulted in many specialized tools and systems providing access to different types of data. For example, we have developed different transactional and reporting tools to manage product defects, test activities, customer support requests, escalations, etc. These tools have been very useful in providing access to data. However, many of these tools remain mostly independent. The fragmentation of data sources and systems continues to be a major challenge at Cisco and other companies (Jacob, 2017).

- *Lack of end-to-end workflow automation*: As the products become large and complex with thousands of software components that are developed and managed by thousands of engineers across the globe, utilizing automation to manage end-to-end workflows is one of the key success factors for the quality management system. Much of the market has yet to take advantage of automated workflows and portals, and in fact, only 21% of the companies surveyed in Jacob (2017) have adopted an integrated quality management system.
- *Still stuck in diagnosis*: Recent advances in analytics can be used to unlock the insights captured within data. Many organizations, including Cisco, are still using analytics to diagnose and understand problems. We have yet to leverage the full power of analytics to provide prescriptive recommendations for engineers to act on.

3.2. Challenges with hybrid organizations

As companies transition to hybrid development processes, the quality management system in large organizations encounters following unique challenges.

- *Quality Measurement*: When the development environment includes elements from Agile, Continuous Integration and Continuous Development (CI/CD), and Waterfall methodologies, it is no longer possible to measure quality metrics as prescribed by one methodology. One must revisit the entire quality measurement system to ensure all metrics are appropriately defined and measured to drive quality decisions for the hybrid organization.
- *Quality Governance*: As companies increase focus on quality as one of the strategic differentiators, executives at all levels of the organization need to include quality as a key decision-making criterion. The quality governance model needs to have well-defined metrics and measures at all levels of an organizational hierarchy — engineers to engineering managers to directors to senior executives. The governance model must also have metrics and measures for each type of deliverable, including projects, programs, large solutions, and portfolios (Fig. 2).
- *Data Integration*: Company-wide data integration is a fundamental need for data-driven decision making. To that end, all quality related data on defects, customer sentiment, escalation, source code, and revenue must be considered while managing and planning software development activities.
- *Standardization*: To provide consistent customer experience, all products and services delivered by a company must provide a consistent level of quality irrespective of the development processes used. This creates a unique challenge for standardization while maintaining the creativity and speed of response expected from a modern software organization.

3.3. Opportunities

The most recent decade has seen significant advances in connectivity, mobility, scalability, analytics, and big data, providing many opportunities to solve legacy as well as new challenges. These opportunities are called Quality 4.0, which blends new technologies with traditional quality methods to arrive at new optimums in operational excellence, performance, and innovation (Jacob, 2017). The new technologies that enable Quality 4.0 are machine learning/ artificial intelligence (ML/AI), connected devices, integrated operations, big data, cloud computing, new applications, and new forms of collaboration like social media and blockchain. These create several unique opportunities to transform quality management systems to the next level. Some of these opportunities are:

- Develop a connected environment that uses data and analytics to provide actionable insight.
- Use deep learning to translate organizational and product-level quality goals into prescriptive actions for individual engineers.
- Develop and implement models based on ML/AI to assess and enhance the effectiveness of the end-to-end software development process.
- Combine engineering data with the customer experience and revenue information to prioritize and implement quality improvement initiatives.

4. Quality transformation at cisco systems

The challenges described in the last section are very significant in achieving an effective data-driven quality management system. Though the end goals of this transformation can be clearly defined, the path to get there depends on the current state of affairs in the organization. This paper presents a general framework developed and implemented at Cisco Systems to transform the legacy quality management system for waterfall organizations into a modern Quality 4.0 system. The framework consists of the following six components:

1. Metrics standardization
2. Process standardization
3. Measurement
4. Reporting
5. Quality analytics
6. Culture and leadership

Using our experiences at Cisco Systems, we explain the general framework and technology we used to achieve the transformation. The quality transformation was initiated because of both top-down and bottom-up initiatives. The top-down initiative was started with an executive mandate to develop a new framework for improving the customer experience for all products in the company. At the same time, several bottom-up initiatives were under development within engineering groups to address technical challenges caused by data fragmentation, process inconsistency, and operational issues. These initiatives resulted in several new ideas and eventually became the quality transformation framework summarized in this paper.

The standardized metrics and processes presented below can be directly applicable to other companies going through a similar transformation. Also, the architecture for our reporting and quality analytics applications can be used by other organizations. However, the measurement system and tools described in this paper are very specific to the infrastructure used in Cisco. Therefore, any organization going through a similar transformation needs to develop its own measurement system based on the tools and infrastructure used in their company.

4.1. Metrics standardization

One of the fundamental differences between Agile and Waterfall methods is the way in which defects are reported. For example, many defects found and fixed within a sprint are never reported in Agile, while all defects found in Waterfall are reported. As a result, Agile teams report far fewer defects from internal testing than Waterfall teams for the same level of quality in a codebase. Therefore, any metric that is normalized with respect to the total number of defects, such as defect removal effectiveness (DRE) or phase containment, will have very different values for Waterfall and Agile teams. Besides, Agile and Waterfall teams often define additional metrics to measure activities done within their development frameworks. It becomes challenging to

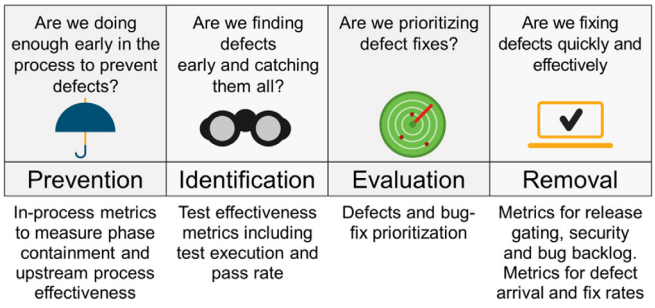

Fig. 1. PIER framework for quality metrics.

define quality metrics when different parts of the organization use different development methods. The metrics program we developed addresses these and other challenges for large engineering organizations consisting of close to 15,000 engineers. The framework for our metrics program is summarized below. The details are presented in Pradhan and Nanniyur (2019).

As discussed by Kan (2003), different companies have used Basili's goal-question-metrics (GQM) framework (Basili et al., 2014) to develop quality metrics programs. The effectiveness of this approach depends on the quality and completeness of questions asked and, therefore, is subjective. To reduce the subjectivity in quality management, we developed a two-dimensional measurement framework using Prevention–Identification–Evaluation–Removal (PIER) and the Software Development Life Cycle (SDLC). The details of these two dimensions are discussed below.

The PIER Framework for Quality Measurement

Defects are fundamental to software quality management. Extensive research has been done over last several decades on different aspects of defect management and associated costs (Jones and Bonsignour, 2011; Boehm et al., 2000). We have developed a novel framework using the basic principles of defect management as the foundation of our quality measurement system. This framework, called PIER, consists of the following categories (Fig. 1):

- *Prevention*: Are we doing enough to prevent defects?
- *Identification or Inspection (Testing)*: Are we finding defects early and are we finding all defects?
- *Evaluation*: Are we prioritizing defect fixing?
- *Removal*: Are we fixing defects accurately and in a timely manner?

Each of these categories consists of a set of metrics to measure the effectiveness of different defect management activities. The details of the quality metrics in each category are given in Pradhan and Nanniyur (2019).

SDLC Milestones and Quality Checkpoints

The other dimension of the quality measurement framework ensures that quality is measured at every major milestone during the software development life cycle (SDLC). Our SDLC process for the hybrid development enterprise is an extension of the SAFe framework for Agile organizations. We used the same basic building blocks of the SAFe framework, which consists of the agile sprint, program, large solution, portfolio, and enterprise (Knaster and Leffingwell, 2019). Then, we superimposed relevant waterfall activities used in hybrid organizations on each of the building blocks (Fig. 2). We identified *Enforcement Points* during the software development/testing activities where a set of quality criteria are enforced, and *Decision Points* where executive-level quality decisions are made. As part of the metrics standardization process, we used the PIER framework to define quality metrics

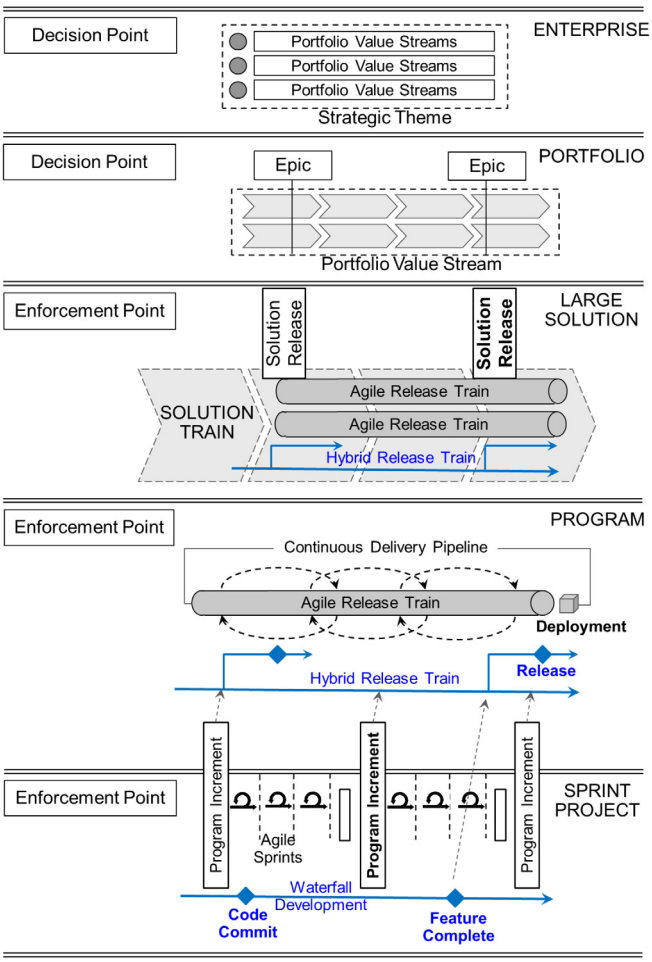

Fig. 2. A framework for complete quality management.

for each enforcement and decision point in the SDLC. The two-dimensional framework using PIER and SDLC building blocks helped us develop a set of specific and measurable criteria for the quality management system. The details of the SDLC framework are described below.

- **Sprint/Project**: At the heart of this framework are the Waterfall projects or Agile Scrum Teams who are responsible for the development and testing of software functionalities or features. This building block has two quality enforcement points: *code commit* and *feature complete*.

- **Program**: A number of projects and Scrum Teams together form organizational structures called Waterfall Release Train (WRT), Agile Release Train (ART), or Hybrid Release Train (HRT). The *Release* and *Deployment* milestones (Fig. 2) are quality enforcement points where pre-defined criteria are enforced before the software is given to customers.
- **Large Solution**: This is used by organizations to support large and complex solutions that typically require multiple WRTs, ARTs, and HRTs. The solution release trains used for delivering complex solutions to customers enforce solution specific release criteria at the *Solution Release* enforcement point.
- **Portfolio**: Several solution release trains are combined to form a portfolio that is aligned with the enterprise strategy. The quality management system defines a set of metrics that enable quality-related decisions at the portfolio level. The

decision metrics measure the effectiveness of different quality activities such as defect prevention, defect removal, and escape reduction, etc. within the portfolio. These metrics are reviewed and decisions on quality improvement initiatives are made at quarterly Engineering Portfolio Review (EPR) meetings.

- **Enterprise**: Portfolios in different groups are aligned with the strategic theme of the enterprise to facilitate quality decisions at the Quarterly Business Reviews (QBR). Like portfolio reviews, QBR evaluates the effectiveness of quality programs in each portfolio, and corrective actions are defined to achieve enterprise strategic objectives.

4.2. Process standardization

The metrics program presented in the last section has been implemented for all Cisco products deployed on customer premises (aka on-prem products). This includes most products from all business units in the company and constitutes more than two-third of Cisco's revenue. We have developed and implemented several standard processes to measure and drive quality programs at enforcement and decision points. The enforcement points include three major software development milestones:

- *Sprint/Project Quality*: Our processes focus on quality from the very beginning of software development activities. The project teams implement quality criteria when the code is committed to the source repository. Agile teams have specific quality criteria as part of their definition of done (DoD) for sprints. We have also developed and implemented detailed quality criteria for feature completion. These criteria together help us measure and drive quality improvement in the early development period.
- *Program/Release Quality*: We have developed and implemented a standard release quality management process. Based on our novel PIER framework described in the previous section, this process helps in comprehensive quality management during the entire release life cycle.
- *Solution Quality*: As in the case of release quality, our solution delivery process implemented a set of distinct quality criteria before solutions are delivered to customers.

While driving quality for the entire company, decisions on software feature velocity versus quality and resource prioritization are made at the executive level. We use two executive review meetings, which were already in place, to make quality decisions at the highest level in the company:

- The portfolio level quality decisions are made at the quarterly Engineering Portfolio Review (EPR) with the executive leadership team.
- The enterprise-level quality strategic decisions are reviewed and prioritized in the Quarterly Business Review (QBR).

We have developed a hierarchical approach for metrics definition and process implementation for different enforcement and decision points. The metrics for the enforcement points are defined using the PIER framework. The quality metrics or indices for the decision points are calculated using a metrics aggregation scheme (Pradhan and Nanniyur, 2019). The summary of the metrics hierarchy is shown in Fig. 3.

4.3. Measurement

Accurate and complete measurement during the entire product life cycle is another essential element of our quality transformation. As reported by LNS research, much of the industry is still struggling to find an effective way to use data-driven and evidence-based decision making (Jacob, 2017). We also experienced similar challenges within Cisco.

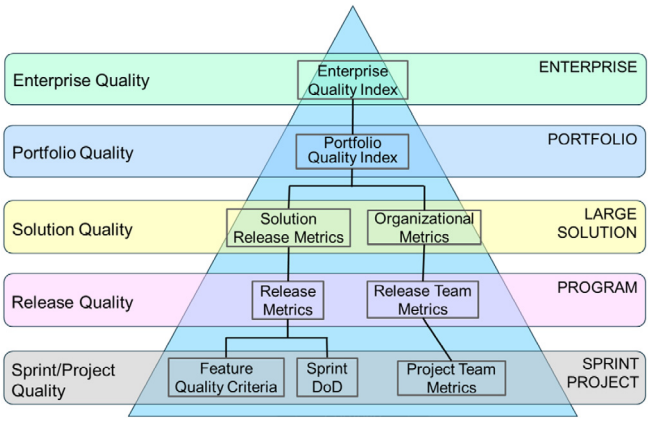

Fig. 3. Metrics hierarchy for the effective implementation of quality programs.

In the context of software development, defects are no longer the only measure of quality. Information and data from many different sources are used to measure various aspects of software quality (Kan, 2003). Like most large organizations, Cisco uses different transactional applications for collecting and managing information for budget, headcount, source code, defects, test activities, customer service requests, escalation tickets, etc. In addition, data is organized based on ownership (e.g., engineering, technical support, and product management, etc.) and applicability (e.g., solution, release, product, and component, etc.). Data from all these applications are available to different product groups in the company through Cisco Data Lake.

We have developed a quality reporting and analytics application using data from all transactional systems. The reporting application is the single source of truth for all quality metrics and analytics for the entire company. The architecture of the quality measurement system is shown in Fig. 4.

4.4. Reporting

There are two main objectives of the reporting application: (i) provide a single source for all quality metrics in the company, and (ii) provide trending and insight to allow effective execution of quality programs. The measurement framework shown in Fig. 4 meets the first objective. To achieve the second objective for effective execution, we have designed our reporting application based on role-based personas (Pruitt and Grudin, 2003; Almaliki et al., 2015). In this approach, real-time quality reports are available for different groups who are responsible for strategic decisions and execution of quality programs. These reports consist of the following six categories or modules:

- Leadership View
- Engineering Quality
- Manufacturing Quality
- Customer Experience Quality
- Product Quality and Compliance
- Analytics and Predictions

The detailed objectives of each category are summarized in Fig. 5. In addition to a role-based design, the reporting application uses a master data management (MDM) system to address the challenges with data fragmentation (Section 3). These modules in our reporting application enable data-driven decision making at different enforcement and decision points.

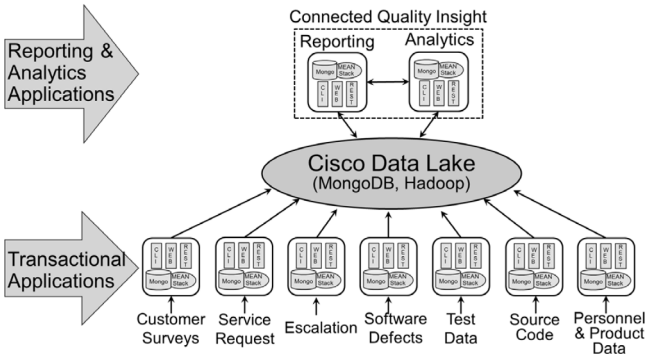

Fig. 4. Data flow and measurement framework.

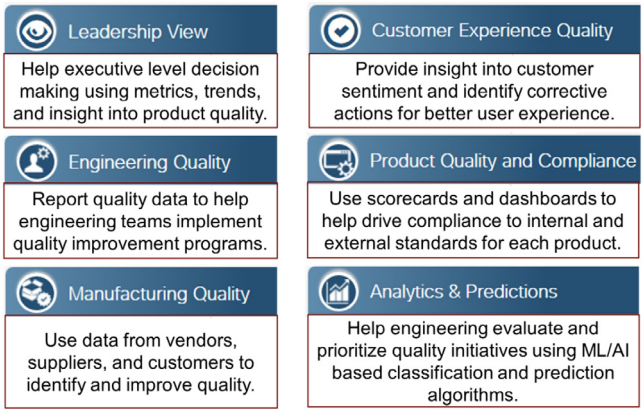

Fig. 5. Major modules in reporting & analytics application.

4.5. Quality analytics

One of the goals of the transformation is to shift the paradigm from "reactive troubleshooting of field quality" to "prescriptive in-process quality management" so that quality issues are identified and fixed before the software is released to customers. To that end, we have focused on two areas of quality analytics: (i) defect classification, and (ii) defect prediction. Details of each of these areas are presented below.

Defect Classification

No matter how effective the quality management program is in an organization, there will always be outstanding open defects before the software is released to customers. Therefore, it is a common practice to evaluate all open defects and fix critical or high impacting ones before releasing the software. Though conceptually simple, it is always a challenge to define and identify critical defects that must be fixed. Defect classification using severity and priority is as old as the software industry. Orthogonal Defect Classification (ODC) is another method used by some in the industry (Chillarege et al., 1992). At Cisco, we developed a defect classification system over the years to meet the specific needs of our customers. Besides severity and priority classification, we use the following definitions for identifying critical defects in our software:

- *Show Stopper (SS)*: A defect that would cause a severe impact on a customer's environment. These are typically less critical than catastrophic failures and may impact a limited number of customers.
- *Operationally Impacting Defects (OID)*: A defect that can be deployment gating to some customers but not as critical or widespread to be classified as SS.
- *Regression Defect*: A regression defect represents functionality that was working in the previous version of the software and is broken in the current version.
- *Potential Customer Found Defect (CFD)*: A defect that is found internally and, if not fixed, has the potential of being found by the customer.

In a typical release process, engineers manually classify defects as one of the above categories. This is a time-consuming process requiring 2–4 person-hours for each defect. Depending on the availability of engineers, the manual evaluation process is difficult to complete in a timely manner. Therefore, we have automated the defect classification process to save engineering time and complete classification in a timely manner. This section summarizes the machine learning (ML) based defect classification algorithms for SS, OID, and regression defects. Automatic identification of potential CFDs using deep learning is summarized in Shobha et al. (2018).

We used defect data for the last two years and used four different algorithms (Table 1) to develop the classification models. Our training dataset consisted of all 146 attributes available within our defect management system. We performed different data cleansing activities by removing data elements with missing values, duplicate variables, and outliers. We also performed exploratory data analysis using Chi-square tests and generated feature importance graphs to eliminate unnecessary attributes with p-value ≥ 0.05. At the end of the exploratory data analysis, the number of attributes was reduced from 256 to 16.

The major challenge in building the classification model was the imbalance in the dataset. For regression classification, we had only 14% observations with Regression = "Yes". In the dataset for the SS model, the number of defects with SS = "Yes" was 15%. Similarly, the dataset for the OID model had 9.6% defects with OID = "Yes". To deal with the imbalance in the dataset, we used the over-sampling algorithms called Synthetic Minority Over-Sampling Technique (SMOTE) and Random Over Sampling Examples (ROSE) (Bunkhumpornpat et al., 2009; Lunardon et al., 2014). We used a set of well-known measures (accuracy, recall, precision, and F1-score) to evaluate the performance of the ML algorithms (D'Ambros et al., 2012). As shown in Table 1, the Random Forest algorithm was the most effective algorithms used to develop the classification models.

We then applied the random forest models to the testing dataset to estimate the defect classification error. Table 2 shows the false positives and false negatives for SS, OID, and regression classification using the Random Forest algorithm. For the defect classification problem, it is essential to reduce false negatives as much as possible so that we do not miss critical defects before releasing software to customers. A higher value for false positives is not a major issue from a quality perspective. A higher value means we need additional time and resource to fix defects identified as false positives. As a result, a high value of false positives will impact the project schedule.

The % of false positives and false negatives are shown in Table 2. Note that the regression value in the learning dataset is based on objective observation of the test output. On the other hand, the SS and OID information are based on a qualitative (subjective) evaluation by engineering and technical support teams. The higher false negative values for SS and OID models could be because of the subjective aspect of the dataset. However, based on our experience, false negative error of less than 10% is quite acceptable for release operations.

Defect Prediction

Reliable prediction of software defects is one of the key areas of research in software engineering (D'Ambros et al., 2012;

Table 1
Comparison of different modeling algorithms for regression, SS, and OID classifications.

Algorithm	Accuracy	Recall	Precision	F1-Score
Show Stopper (SS) classification				
Naïve bayes	0.60	0.65	0.53	0.58
Random forest	**0.82**	**0.82**	**0.78**	**0.80**
Logistic regression	0.66	0.50	0.62	0.56
XGBoost classifier	0.70	0.59	0.66	0.62
Operationally Impacting Defect (OID) classification				
Naïve bayes	0.60	0.81	0.56	0.66
Random forest	**0.95**	**0.99**	**0.91**	**0.95**
Logistic regression	0.69	0.70	0.67	0.68
XGBoost classifier	0.77	0.85	0.72	0.78
Regression defect classification				
Naïve bayes	0.65	0.60	0.64	0.67
Random forest	**0.92**	**0.96**	**0.88**	**0.92**
Logistic regression	0.64	0.63	0.64	0.64
XGBoost classifier	0.73	0.72	0.74	0.73

Table 2
Defect classification error using the random forest algorithm.

Classification	False positive	False negative
SS	9.6%	7.8%
OID	16%	7%
Regression	6.6%	1.8%

Table 3
Error comparison for different algorithms used for defect prediction models.

Model & Scenario	RMSE	MAE
Code commit model		
Linear regression	161	76
Random forest	107	45
XG Boost	**96**	**42**
Feature complete model		
Linear regression	**12**	**32**
Random forest	12	41
XGBoost	14	46
Release model		
Linear regression	20	64
Random forest	**12**	**32**
XGBoost	13	35

Table 4
Defect prediction errors for different models.

Model	Defect prediction error
Code commit	14.3%
Feature complete	2.8%
Release	11.7%

Hammouri et al., 2018). Defect prediction is also very important for quality management in the industry. It is useful for resource planning and evaluating test effectiveness during software development and testing activities. It is also essential to predict defect prone components so that additional testing and quality improvement for these components can be done before the release. Like many other companies, Cisco Systems used defect density (Rahmani and Khazanchi, 2010) to predict the number of defects expected during software development activities. The defect density is calculated by dividing the number of known defects by the software size measured by Lines of Code (LOC). As shown in Prasad et al. (2015), the number of defects in software depends on many factors other than LOC. Therefore, defect density-based prediction has a higher error rate. Our experience has shown that the defect density-based prediction could result in greater than 25% prediction error.

In order to develop more accurate defect prediction, we used machine learning (ML) to developed models at three quality enforcement points: Code Commit, Feature Complete, and Release (Fig. 2). The defect prediction model used at code commit uses Lines of Code (LOC) to predict the number of defects likely to be found during functional testing. Note that LOC is the only information available at code commit. This prediction is used to plan testing activities before the Feature Complete milestone. The defect prediction model used at Feature Complete uses LOC along with additional quality metrics such as defect arrival and disposal rate, bug slip-through between testing activities, etc. Similarly, the defect prediction model used at the Release includes all attributes used in the Feature Complete model and additional quality metrics measured at the release milestone.

The training and validation datasets for these models consist of ~800 observations for 8 releases spanning over last 3 years. We performed basic data cleaning activities to remove attributes with null values, duplicates, and outliers. After data cleansing, the number of attributes was reduced to 119. We performed multicollinearity analysis using Stepwise VIF (Variance Inflation Factor) to remove attributes that are highly correlated (Akinwande et al., 2015). After eliminating attributes with VIF > 10, we got a dataset with only 35 attributes, all of which are used in the model building.

We used three different algorithms for developing defect prediction models and selected the best model using standard model evaluation criteria (Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE)) (D'Ambros et al., 2012). The error comparison for models is presented in Table 3. XG Boost was found to be the most effective for the Code Commit model. The Linear Regression and Random Forest algorithms were the most effective for the Feature Complete and Release models, respectively. We estimated the defect prediction error using the best ML algorithm for each model. As shown in Table 4, the defect prediction errors are less than 12% for both the Feature Complete and the Release models. The prediction error is higher (14.3%) for the code commit model that uses only LOC as the model attribute. These error levels are much lower than 25%, which was the error in prediction using defect density.

4.6. Culture and leadership

An organization that has "a culture of quality" exhibits four key elements: process participation, responsibility, credibility, and empowerment (Jacob, 2017). Throughout the evolution of software engineering, experts have shown again and again that the most important criteria for success are the people in an organization. At Cisco, we find that focusing less on the process and methods, and more on the people and culture, and communicating effectively and transparently, are the keys to success. We know from (Basili et al., 2014) and many other publications that people factors are far more correlated to project effort and schedule than any architectural, geographical, or process related measure. COCOMO II model calibrations have exhibited a 10:1 effect of personnel capability, experience, and continuity, and a severely negative effect from personnel churn (Boehm et al., 2000). Strategically, this culture is promoted through better

connectivity, visibility, insights, and collaboration using connected data and process automation.

In addition, we used several conventional methods like internal wiki pages, social media groups, executive summits, technical presentations, and launch & learn sessions, etc. to increase communication or participation at all levels of Cisco. One of the major factors in the success of this transformation so far is the alignment of quality objectives with the strategic goals of the company. The other important factors are the participation and support of Cisco's executive leadership team to promote and drive the culture of quality. It would have been impossible to achieve the transformation at this scale without the support and active participation of our leadership team.

5. Implementation and results

The first phase of this transformation includes the definition of the framework, metrics standardization, development of a reporting and analytics dashboard, process standardization, and ML/AI model development. The second phase implements ML/AI based prescriptive models and process automation, which enhances the activities started in the first phase. As presented in this section, the first phase of the implementation shows close to a 20% reduction in customer found defects and ∼45% improvement in release maturity time. Details of these results are presented below.

The results presented in this section are for the Enterprise Networking (EN) group, which was first to adopt the quality transformation framework. The EN is a hybrid organization that supports 55 platforms, includes 2,500+ engineers across four geographies, and receives over 300 daily changesets. Close to 20% of the teams in the EN group use the Agile development methodology while the others use the Waterfall and continuous integration development model. We examined three different outcomes to evaluate the effectiveness of the proposed framework: (a) defect incoming profile during internal testing (also known as the defect mountain chart, Fig. 6), (b) annual failure rate (Fig. 7), and (c) time to maturity in the field (Fig. 8).

One of the main objectives of this framework is to find more defects early in the internal testing cycles so that teams can fix the defects early. Fig. 6 shows the weekly defect discovery rate for three releases from the beginning of the release until many months after the release. As expected, the defect discovery profile for the pre-release period is similar to the Rayleigh distribution. For an effective testing process, the peak of the curve or the "defect mountain" should be as far to the left as possible. As we see from Fig. 6, because of the emphasis on automation code coverage and test execution, the peak of the defect mountain has moved by three months to the left after implementation of the framework in Release 2. This is a significant improvement in a release that takes 8 months from beginning (i.e., requirement definition) to the end (i.e., customer delivery). We continue to see a further shift to the left by a smaller amount in the third release.

Another important metric to measure customer quality experience is the number of customer found defects (CFDs) normalized with respect to the number of customer installations. We call this the CFD per install base. The graph in Fig. 7 shows the number of CFDs found each month divided by the new deployments in that month with respect to the FCS (First Customer Shipment) or the release date. As the graph shows, the CFDs per install base decreases exponentially over time. The new releases show a lower value for a given month after FCS, which indicates quality improvement release-over-release. We implemented the new quality framework after the first release and saw a significant improvement in the second release. As expected, the rate of improvement is becoming smaller in the third release.

We also investigated release maturity as another way to validate the effectiveness of the quality transformation. The maturity

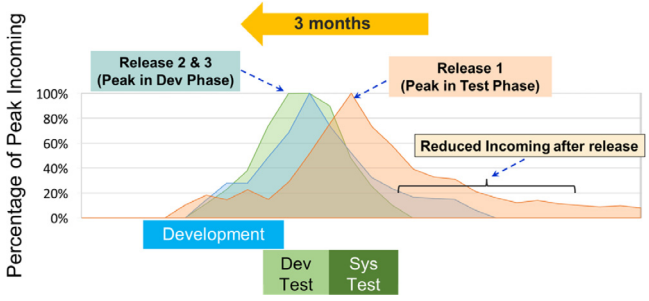

Fig. 6. Defect mountain movement release-over-release.

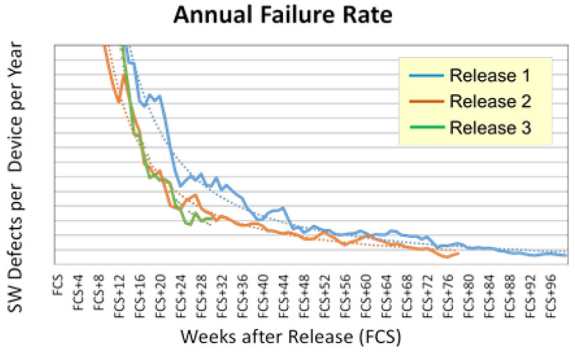

Fig. 7. The trend of customer found defects per installation.

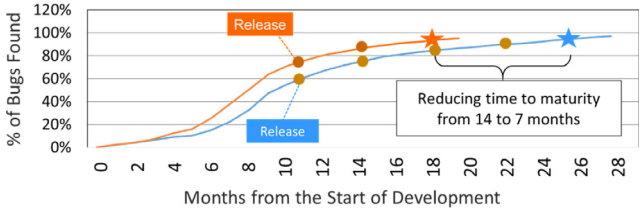

Fig. 8. Software maturity trend.

of a release can be measured by the reliability growth model (Kan, 2003). In this model, the cumulative CFDs follow an S-curve and converge to an asymptotic value representing the total number of defects that exist in the software. Fig. 8 shows the trend of cumulative CFDs for the first two releases. The third release has not been in the field long enough to show any noticeable trend. As we can see from Fig. 8, the first release took 14 months to reach a maturity level of 90% (i.e., 90% of possible defects found). The second release reached the same level of maturity within seven months, which is a 50% improvement.

The results observed so far validate the effectiveness of the transformation in improving product quality for large hybrid development organizations. We continue to monitor the outcome metrics from other groups to further validate and finetune our implementation.

6. Threats to validity

Although the results presented in Section 5 are promising, there are some factors that threaten the validity of our conclusions. This section summarizes both internal and external threats.

Internal validity is the ability of a study to establish a causal link between independent and dependent variables regardless of what the variables are believed to represent (Perry et al., 2000). We have proposed six elements for the transformation and three

outcome metrics to measure the effectiveness of the transformation. The main threat to internal validity in our study comes from the lack of mathematical correlation between the elements of the transformation and the outcome metrics. This is an inherent challenge in most of the studies on real-life transformation in large organizations where control experiments are not possible to establish any correlation. However, given the consistent and significant improvement seen in all outcome metrics, we associate these improvements to the changes introduced during the transformation. Based on our experience, it is unlikely to have such a large improvement in outcome due to other smaller activities that are continuously implemented in large organizations with thousands of engineers. We are continuing to monitor the results from other organizations within Cisco to validate the effectiveness of our transformation.

External validity refers to how the results of a study generalize (Perry et al., 2000). The results in our study include the Enterprise Networking (EN) group, which is one of the largest and diverse groups in Cisco. The EN group develops 55 platforms for switching, routing, wireless, Industrial IoT, and network management. It consists of more than 2,500 engineers across four major geographic locations with close to 20% of the teams using the Agile development methodology, and others using the Waterfall or continuous integration development model. While the results from the EN group give evidence of the ability to generalize, results from the implementation in other groups e.g., Service Provider, Data Center, etc. will increase our confidence in the general applicability of this transformation.

Another potential threat is the availability of tools and infrastructure to implement this transformation. Like other large software companies, Cisco uses many opensource and commercial tools in our IT infrastructure and developed several applications to connect information from these tools. Except for general operational challenges associated with application development, we did not encounter any major issues related to tools. It should be noted that every large-scale transformation or process change in any company (not just Cisco) is expected to encounter similar challenges related to tools and IT infrastructure. Therefore, these challenges need to be mitigated by the right choice of tools, IT infrastructure, and applications.

7. Summary and future work

We have presented a case study on the large-scale transformation of a legacy quality management system to a modern system enabled by machine learning/artificial intelligence, connected data, integrated operations, and big data technologies. The framework for this transformation is defined in terms of six distinct areas: metrics standardization, process standardization, measurement, reporting, quality analytics, and culture & leadership. Phase I of the transformation has been successfully implemented in Cisco Systems. The results for several key products validate the effectiveness of the transformation in terms of the incoming defect profiles, annual failure rates, and release maturity timeline.

We have also presented results on ML-based model development for defect classification and prediction. The next step in our quality journey is to implement analytics models as part of the quality management system. We are continuing to monitor KPIs and feedback from cross-functional teams. A detailed analysis of the current framework will be undertaken to improve the metrics, processes, and infrastructure used in our quality transformation. In addition, we are currently working on expanding this framework to cloud products.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Akinwande, O., Dikko, H.G., Agboola, S., 2015. Variance inflation factor: As a condition for the inclusion of suppressor variable(s) in regression analysis. Open J. Stat. 754–767.

Almaliki, M., Ncube, C., Ali, R., 2015. Adaptive software-based Feedback Acquisition: A Persona-based design. In: 2015 IEEE 9th International Conference on Research Challenges in Information Science, RCIS, Athens, pp. 100–111.

Anthony, S.D., Trotter, A., Schwartz, E.I., 2019. The Top 20 Business Transformations of the Last Decade. Harvard Business Review, Sept 24, (https://hbr.org/2019/09/the-top-20-business-transformations-of-the-last-decade).

Basili, V., Trendowicz, A., Kowalczyk, M., et al., 2014. Aligning Organizations Through Measurement. Springer.

Boehm, B., Abts, C., Chulani, S., et al., 2000. Software Cost Estimation with COCOMO II. Prentice-Hall.

Boehm, B., Turner, R., 2004. Balancing Agility and Discipline. Addison Wesley.

Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C., 2009. Safe-Level- SMOTE: Safe-Level-Synthetic Minority Oversampling Technique for Handling the Class Imbalanced Problem. Springer Berlin Heidelberg, pp. 475–482.

Chemuturi, M., 2011. Mastering Software Quality Assurance – Best Practices, Tools and Techniques for Software Developers. J. Ross Publishing.

Chillarege, R., Bhandari, I.S., Chaar, J.K., et al., 1992. Orthogonal defect classification - A concept for in-process measurements. IEEE Trans. Softw. Eng. 18 (11).

Cognizant White Paper, 2010. Software Quality Transformation: Focus on Results, Not Process, Cognizant, Retrieved on 30 Dec 2019 from https://pdfs.semanticscholar.org/0d35/8a8abed71950cce24a8c38426e5e962fd690.pdf.

D'Ambros, M., Lanza, M., Robbes, R., 2012. Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empir. Softw. Eng. 17 (4–5), 531–577.

Ebert, C., Duarte, C.H.C., 2018. Digital transformation. IEEE Softw. 35 (4), July/August.

Forbes Insight, 2018. Workforce and culture can transform your organization, Forbes insight in association with scrum alliance.

Hammouri, A., Hammad, M., Alnabhan, M., Alsarayrah, F., 2018. Software bug prediction using machine learning approach. Int. J. Adv. Comput. Sci. Appl. (IJACSA) 9 (2), 78–83.

Handscomb, C., Jaenicke, A., Kaur, K., et al., 2018. How To Mess Up Your Agile Transformation in Seven Easy (Mis)Steps. McKinsey Report on Organization Practice, April.

Horch, J.W., 2003. Practical Guide o Software Quality Management, second ed. Artech House.

Jacob, D., 2017. Quality 4.0 Impact and Strategy Handbook – Getting Digitally Connected to Transform Quality Management. LNS Research.

Jones, C., 2000. Software Assessment, Benchmarks, and Best Practices. In: Addison-Wesley Information Technology Series.

Jones, C., Bonsignour, O., 2011. The Economics of Software Quality. Addison Wesley.

Kan, S.H., 2003. Metrics and Models in Software Quality Engineering, second ed. Addition-Wesley.

King, T., 2019. Transforming Testing: A Recipe for Quality Engineering. CIO Review, Retrieved on 31 Dec 2019 from https://softwaretesting.cioreview.com/cxoinsight/transforming-testing-a-recipe-for-quality-engineering-nid-23988-cid-112.html.

Knaster, R., Leffingwell, D., 2019. SAFe Distilled – Applying the Scaled Agile Framework for Lean Enterprises. Addison-Wesley.

Leffingwell, D., 2007. Scaling Software Agility – Best Practices for Large Enterprise. Addison-Wesley.

Lunardon, N., Menardi, G., Torelli, N., 2014. ROSE: A package for binary imbalanced learning. R J. 6 (1).

Miller, G.J., 2013. Agile Problems, Challenges, & Failures. Paper Presented at PMI® Global Congress 2013. Project Management Institute, North America, New Orleans, LA, Newtown Square, PA.

Mistrik, I., Soley, R., Ali, N., et al. (Eds.), 2016. Software Quality Assurance in Large Scale and Complex Software-Intensive Systems. Elsevier.

Nauman, M., 2017. Shift Left Testing: Going Beyond Agile. In: STARWEST 2017 – Software Testing Conference, Anaheim, California, October.

Nir, R., 2020. The roles of Atlassian tools in quality management. Retrieved on 22 August 2020 from https://radbee.com/the-roles-of-atlassian-tools-in-quality-management/.

Perry, D., Porter, A., Votta, L., 2000. Empirical studies of software engineering: A roadmap. In: Proceedings of the Conference on the Future of Software Engineering. ACM New York, USA, pp. 345–355.

Pettey, C., 2018. Moving to a Software Subscription Model. Gartner Research, Retrieved on 4 April 2019 from https://www.gartner.com/smarterwithgartner/moving-to-a-software-subscription-model/.

Pradhan, S., Nanniyur, V., 2019. Back to basics – Redefining Quality Measurement for Hybrid Software Development Organizations. In: 2019 IEEE 30th International Symposium on Software Reliability Engineering, ISSRE, Berlin, Germany, pp. 402–412.

Prasad, M.C.M., Florence, L., Arya, A., 2015. A study on software metrics based software defect prediction using data mining and machine learning techniques. Int. J. Database Theory Appl. 8 (3), 179–190.

Pruitt, J., Grudin, J., 2003. Personas: Practice and Theory. In: Proceedings of the 2003 Conference on Designing for User Experiences, June 06–07, San Francisco, USA.

Quality Management Software, 2020. Capterra. Retrieved on 22 Aug 2020 from https://www.capterra.com/sem-compare/quality-management-software.

Rahmani, C., Khazanchi, D., 2010. A Study on Defect Density of Open Source Software. In: Proceedings of the 9th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2010, pp. 679–683.

Shobha, D.V., Sumith, R.B., Vignesh, T., 2018. Customer success using deep learning. Adv. Econ. Bus. 6 (5), 332–338.

Yackel, R., 2020. Best Practices for Shifting Testing Left to Deliver Higher Quality Software Faster. Gartner Research, Retrieved on 2 Jan 2020 from https://www.gartner.com/technology/media-products/newsletters/tricentis/1-5SAF30Q/index.html.

Satya Pradhan: Satya Pradhan is Sr. Quality Lead for Cisco's Enterprise Networking Business. He has led the development and implementation of the quality metrics program in Cisco. Pradhan has led many strategic initiatives on release process, software quality, analytics, engineering operations, and infrastructure development. Several of his pioneering ideas have been implemented and adopted across Cisco. He holds a master's degree in Computer Science and Ph.D. in aerospace engineering. He has more than 30 technical publications in software engineering, quality, robotics, and aerospace engineering, and a book on "Creative Thinking – A Problem Based Approach to Teach Creativity in STEM."

Venky Nanniyur: Venky Nanniyur is Vice President of Planning and Operations for Cisco's Enterprise Networking Business spanning 6,000+ engineers and contributing over $15 billion to Cisco's revenue. Nanniyur is responsible for driving operational efficiencies, managing quality and customer experience. He runs large-scale transformation initiatives across Enterprise Networking and continues to improve process engineering with DevOps and Agile. He has implemented and oversees a rigorous portfolio planning process to align investments to business priorities.