SN Computer Science (2022) 3:38
https://doi.org/10.1007/s42979-021-00899-9

ORIGINAL RESEARCH

Software Product System Model: A Customer-Value Oriented, Adaptable, DevOps-Based Product Model

Haluk Altunel[1,2] · Bilge Say[3]

Received: 30 March 2021 / Accepted: 21 September 2021 / Published online: 2 November 2021

Abstract
DevOps pipelines have brought notable advantages, such as fast and frequent software delivery to software production paradigms, but dynamically dealing with quality attributes desired by the customer employing a DevOps pipeline remains a challenge. This work aims to define the design of a systems thinking inspired model, called Software Product System Model (SPSM), applying a customer-value oriented, holistic approach for implementing quality requirements, and its application and evaluation in a large software house. The main features include dynamic control of quality gates, the parameters of which are driven by customer requirements and feedback from surveys. All of the inputs are collected in a product backlog and fed forward to the quality gates over the DevOps pipeline. SPSM was successfully deployed in a large software house extending a DevOps pipeline with an accompanying improvement of customer-value oriented key performance indicators for projects. In a 2-year-long case study, security and code quality were the main quality attributes, with the metrics on security vulnerabilities and unit test coverage. At the end of the 2020, the DevOps pipeline within SPSM provided a 69.50% decrease of security vulnerabilities of all software products, and a 29.43% increase in unit test coverage for the whole code base for increasing code quality. At the end of 2020, the project completion ratio was measured to be 99.50% and the Schedule Performance Index (SPI) was measured to be 99.78% as the average of 762 projects delivered. The flexibility of SPSM allowed the software house to adapt to changing customer expectations. A checklist is provided for the replicability of the model application.

Keywords DevOps · Software Product System Model · Systems thinking · Software attributes · Customer value · Software metrics · Software product management

Introduction

DevOps has become a popular paradigm and introduced a major paradigm change in the software development world over the last decade [1, 2]. Software product development may include DevOps pipelines within a product management perspective. Within the product management boundaries, customer-value oriented software quality attributes such as reliability, security, robustness, etc. should also be adequately integrated into software product systems. Software development business models have been evolving from tailor-made, customer-specific projects to software products [3]. Like any other product, software products also have life-cycles with their own stages of development, introduction, growth, maturity, and decline [4]. Software product management covers the pre-development phase, where the product is defined with its vision, strategy and roadmap. Software development

This article is part of the topical collection "New Paradigms of Software Production and Deployment" guest edited by Alfredo Capozucca, Jean-Michel Bruel, Manuel Mazzara and Bertrand Meyer.

✉ Haluk Altunel
altunel@bilkent.edu.tr

Bilge Say
bilge.say@atilim.edu.tr

1 Softtech Inc., Ankara, Turkey

2 Department of Computer Engineering, Bilkent University, Ankara, Turkey

3 Department of Software Engineering, Atilim University, Ankara, Turkey

is well known for its agile and waterfall methodologies. Deploying a software into a production environment is not the end, rather it is the birth of the product. After that point, post-development life starts. That means the new born product needs care to survive and grow like an infant. For healthy growth, it needs to be monitored carefully and to be fed with new features, such as new product versions. Moreover, collecting and using product performance and customer satisfaction data become even more important for durable products across all of life-cycle stages. The holistic product view of the software is the first of the driving forces of the DevOps model and its application to be presented in this paper.

Due to complex engineering problems within DevOps, the mainstream research is concentrated on tool chains, architecture and organizational issues between developers and operation experts. Many applications and tools have been introduced into the industry for implementing and orchestrating software development pipelines [5, 6]. Although DevOps has been supported by a variety of tool chains for continuous integration and delivery, most of them focus on maximizing the efficiency and speed of software delivery. This approach assumes the faster the production pipeline is, the better it is for the whole system in terms of time-to-market. However, production speed cannot be the main focus alone for every industry and every product. As stated above, other attributes of software as a product can become dominant, such as usability, robustness, correctness, and security. Moreover, the importance of the attributes can change with time. Therefore, a static DevOps pipeline designed for faster production may not provide flexibility for other attributes required by the software products. The second driving force for the model emergence is the need for adaptability based on customer-driven product values.

In this study, a cross-disciplinary interaction between systems thinking and software engineering disciplines is put into use to define and design what is called the Software Product System Model (SPSM). The model suggests extending the boundary of the system to cover all life-cycle stages of software products including pre-development, development and post-development phases. The model proposes to optimize the whole system based on the product attribute that is determined by the durability and success of the product. Customer-value is the key here to determine the leading attribute. The model is applied over an extended period in a large software development company. We will first review related problems and approaches in the literature, then introduce the conceptual design of the features of the model and finally, present its application and evaluation in a large software company, ending with a discussion of the validity and implications of the approach.

Problem Identification and Related Work

The motivation behind the adaptation of systems thinking principles into developing a DevOps based product development model is to produce a product line that dynamically manages and monitors the requirements for software products through automation and the visibility provided by the pre-development, development and post-development tool chain while enjoying the more obvious gains brought along with continuous integration and shorter development cycles by DevOps. Before introducing the design of the suggested model, current approaches to DevOps from a perspective of quality will be briefly evaluated. Then the application of systems approach in software product development will be briefly reviewed.

Current Perspectives Relating DevOps to Software Quality

Although the DevOps mindset and its evolution seems to have emphasized quality, reliability and correctness of software in its core definitions [1], empirical studies focusing on the challenges of DevOps practices with respect to systemic views of software, that involves the entire ecosystem of the software as a product in a holistic manner, has been relatively rare [7]. This observation is also affected by the fact that DevOps Maturity Models of process improvement emphasizing continuous quality improvement and behavior monitoring with feedback and optimization mechanisms have been proposed but not yet fully implemented [8]. An ISO/IEC standard for Agile and DevOps Principles and Practices under Software and Systems Engineering is still under development (ISO/IEC AWI TR 24586). However, there is an increase in the number of recent works of research that strive to empirically understand, measure and improve the full impact of DevOps on software product quality.

A recent systematic literature review, aligned with the model of ISO/IEC 25010 Systems and Software Quality Requirements and Evaluation standard, covering 31 articles in a span of 10 years on DevOps product quality emphasizes the challenges to several aspects of product quality [9]. Additional benefits to reliability and maintainability brought along by mechanisms such as deployment or test automation are marked by challenges to security, all considered as part of quality attributes in the empirical studies evaluated. Although technical guides and proposals for improving specific quality aspects of DevOps such as dependability and security are available [10–12], research in software industry involving empirical design and evaluation of overall quality processes within DevOps are still needed. A recent multi-case study of five companies using agile and DevOps practices emphasizes the perceived difficulties by practitioners, of striking the right balance between speed of deploying new functionality versus quality concerns and possible conflicts raised in monitoring with suitable metrics [7]. In a case study, Handl et al. [10] propose and evaluate an extension to the QUAMOCO quality meta-model for integration of dynamic nonfunctional requirements at individual feature levels as required by DevOps practices. A recent survey on DevOps practitioners reveal that lessening complexity and smooth integration with existing tools in the DevOps pipeline is an enabler for performance engineering for quality [13]. The motivation of the present study is to contribute to this body of knowledge by describing a conceptual framework of a systems-based approach to quality in DevOps and its implementation as an industrial case study.

Holistic System Approach in Software Development

Systems approach to software development have been inspired by seminal works, such as Weinberg [14] (originally published in 1975) and Senge [15] (originally published in 1990). Research exploring the system view in current paradigms of software development exist, such as agile methods within Weinberg's system view [16] or changes brought about by lean approaches to system elements [17]. The current study is inspired by a wish to extend the systematic explorations of Senge's work to enhancing Software Product System Model with DevOps. Senge's emphasis on learning as an organization with the joint responsibility of all parties involved as well as system thinking approach that integrates the interdependent parts comply well with the DevOps philosophy.

Another holistic implementation perspective of DevOps practices is introduced by Google as Site Reliability Engineering (SRE) [18]. SRE includes principles and practices for developing and operating large scale distributed software systems. It is an extension of the current DevOps best practices and SRE is realized with Site Reliability Engineers. It includes governance practices, such as training, meetings and metrics for managerial purposes. Although SRE has been utilized in large scale Google products, the product management perspective is not the primary focus. Moreover, reliability may not be the main concern for every customer. Our model is differentiated by the flexible optimization focus based on customer expectations that may change by time.

The present study reports the design, implementation and evaluation of a DevOps pipeline model in a large software house for creating customer-value for complex and dynamic quality issues with a system thinking perspective. The applicability of system thinking principles have been stated in DevOps literature before. We will cite a few ranging from DevOps principles, to DevOps metrics and to DevOps leadership. Kim et al. [19] introduce "The Three Ways" of DevOps: the systems thinking in the flow rising from seeing the whole complex system as creating business value; shortening and amplifying the feedback loops particularly between development and operations and a continuous learning and experimentation cycle, where risk taking and faults are dealt with positively within the culture of learning. Forsgren and Kersten [20] detail and justify how using systems based metrics jointly with survey based metrics can help attain a holistic systems view. Maroukian and Gulliver [21] cite "holistic systems thinking" as one of the prominent leadership skills for DevOps in an analysis of thirty interviews with an international sample of DevOps practitioners. To overcome the quality problems stated in the previous section, a system thinking approach [22] is also adopted in this study for designing the DevOps pipeline used in a large software house with an eye for product management.

Software Product Management

Software product management is a multidisciplinary area lying at the intersection of business, engineering and user experience [23]. Product management is not restricted to development, rather it covers the entirety of product life-cycle management. Product life-cycle management is valid for software products as well, and projects planned for the product have to consider the life-cycle stage of the product [4].

Companies focusing on software products target product-based growth by managing software assets from a business point of view. This is why software product management has become one of the most challenging and rewarding domain in software industry [24]. Since software products need planning new versions and frequent releases of them during products life cycle, DevOps is an important backbone of such software systems [25]. An attempt to integrate a product life cycle view with DevOps has, for example, been done in a business process architecture framework in [26]; our work is to present such a product-oriented conceptual framework and its application in an enterprise setting from a systems point of view.

Software Product System Model as a Conceptual Framework

In this section, the conceptual and architectural elements of a systems thinking inspired, DevOps based model is outlined. The model focuses on software product management from a customer-value perspective. There are three main factors that differentiate the model from other models of software production. The first is the software product management

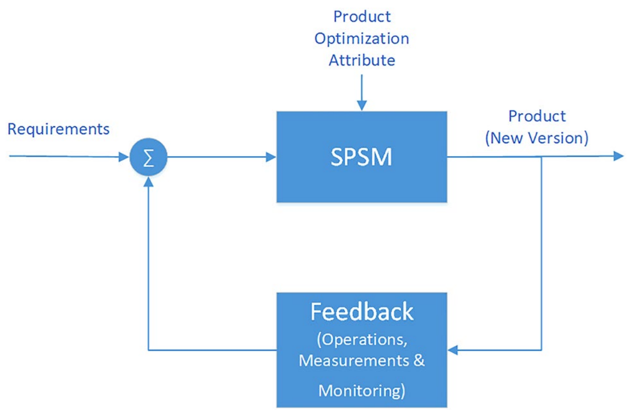

Product Optimization Attribute
Requirements
Σ
SPSM
Product (New Version)
Feedback (Operations, Measurements & Monitoring)

Fig. 1 SPSM as a systems process

focus. Since products have strategies and roadmaps covering middle and long-term goals as well as short-term deliveries, product-focused software production is more concentrated on what should be implemented. The follow-up question of how should it be implemented and delivered is addressed in development phase and in the DevOps pipeline. The second factor is the systems thinking perspective that helps us to combine product management and software development into a single system. This leads to an extended re-definition of the system boundaries. The third is its adaptability to the changing focus of customers. Customers use software for their own business requirements that change frequently. These changes should be met by the software products they use. Static products that do not have the necessary flexibility to meet the needs of customers cannot survive for a long period of time. Thus, adaptability is the most important factor of our framework.

In the way we envision the system, the system boundary is extended to cover all the steps of software product development, including product management, requirements engineering, design, development, integration, testing, and deployment. We named this systems-thinking based approach Software Product System Model or SPSM for short. SPSM covers the end-to-end software development life-cycle from a product perspective, as shown in Fig. 1. In this model, optimization problems are considered in the context of the whole system rather than dealing with them on a local scale. Hence as part of SPSM, DevOps is designed in a dynamic mode for the solution of optimization problems.

SPSM is designed for customer-value orientation in the core. Its roots are coming from the large-scale agile approach, product-based transformation and DevOps transformation. The proposed model uses the continuous improvement concept of System Kaizen which requires system-level addressing of continuous improvement of problems over a long period [27]. Metrics are specified for the improvement of model.

Customer-value oriented, DevOps based SPSM is shown with its details in Fig. 2. The main roles are categorized as product manager, product development & operations team, product support team, and customers & users. The product manager can be a single person or group of people who are responsible for larger products. This role is mainly responsible for the success of the product. Setting the product vision and strategy, defining and updating the product roadmap and prioritizing the product backlog are the pre-development activities led by the product manager. Monitoring and approving the product version after the development phase are their two main responsibilities during development. The product manager is active during the post-development period as well, focusing on customer satisfaction and the performance of the product with data collected based on the metrics. The data is fed back, as it is in systems thinking, to revise the product strategy and roadmap when required. The product life-cycle stage is another important parameter that should be taken into account when managing the product.

The product development & operations team is the multidisciplinary team composed of different roles depending on the size and industry of the software product. For each product there may be a single dedicated team or a group of teams, such as a tribe or a scrum of scrums depending on the methodology. For convenience we will call this role simply a "team" from this point on, independent of the structure and size of the development team. This team should be capable of analyzing the requirements of the software in detail, designing the software architecture, implementing the software, and testing the software. The team must be competent in DevOps practices to manage continuous integration and continuous deployment. Technical skills based on building the DevOps pipeline, such as Infrastructure as Code, Cloud Competency, Kubernetes knowledge and similar, depends on the DevOps pipeline implementation and they are not within the scope of our framework. This team is responsible for monitoring the development and post-development of the software product. The performance metrics are defined based on the product optimization attributes and are collected throughout the products life-time.

The product support team, whose members are separate from the product development & operations team except for the product manager and the developers working on the production issues, is responsible for the incidents occurring during the post-development phase. These incidents can be handled based on the Information Technology Infrastructure Library (ITIL) framework [28]. ITIL defines incidents as the unplanned interruption or reduction in the quality of a software service that is in a production environment. The data gathered during the post-development phase are based on customer satisfaction and product performance based metrics.

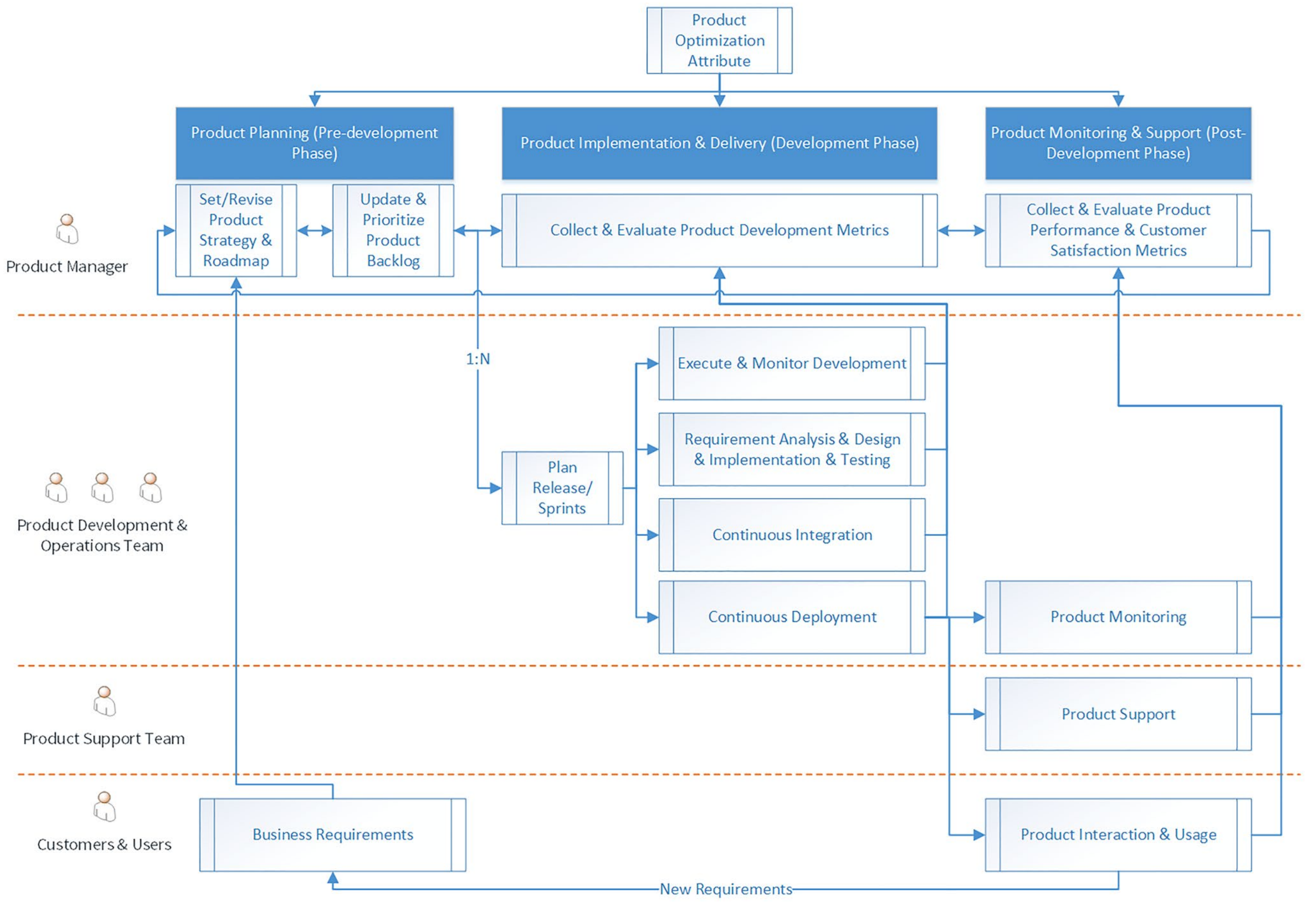

Product Optimization Attribute
Product Planning (Pre-development Phase)
Product Implementation & Delivery (Development Phase)
Product Monitoring & Support (Post-Development Phase)
Product Manager
Set/Revise Product Strategy & Roadmap
Update & Prioritize Product Backlog
Collect & Evaluate Product Development Metrics
Collect & Evaluate Product Performance & Customer Satisfaction Metrics
1:N
Execute & Monitor Development
Requirement Analysis & Design & Implementation & Testing
Plan Release/ Sprints
Continuous Integration
Continuous Deployment
Product Monitoring
Product Development & Operations Team
Product Support Team
Product Support
Customers & Users
Business Requirements
Product Interaction & Usage
New Requirements

Fig. 2 Process flow in SPSM

Customers and users are the last group of people, the group that needs and uses the software product. Their interaction with the software product provides valuable information to the product manager for how to update the product roadmap and backlog. They can also discover new requirements during their use of the product.

SPSM is composed of three main phases: pre-development, development and post-development. The holistic approach taken from system thinking to define the system boundaries, the from-customer-to-customer system, is introduced into the SPSM. The from-customer-to-customer system is crucial for definition and solving the system optimization problems. The attributes should be defined for the whole system and the system optimization problems should be defined based on the customer's expectations and solved within the system boundaries as a software product. If SPSM is divided into sub-systems, such as backlog management, pipeline management, and maintenance; then, each sub-system will be optimized for local parameters, such as optimum backlog management, optimum DevOps pipeline management or optimum maintenance management. The collection of local optimums will be different from the whole system optimum; therefore, in SPSM, we focus on the whole system optimum [14].

In SPSM, the customer defines the business requirements with their business impact. These requirements are collected by the product manager and used in the product roadmap and product backlog. Items from the product backlog are pulled by the product development & operations team based on the priorities and used for the phase/iteration planning. The team can use either a waterfall or an agile methodology based on their choice for the software development principles. SPSM supports mainstream agile methodologies as Scrum, Kanban and Scrumban [29]. If the team adopts a waterfall approach, then the planning phase of the project is based on the items in the product backlog. If the team adopts an agile approach, e.g., Scrum, then release and iteration planning is done. In either case, the output of the planning session is the list of items chosen from the product backlog and the new version plan of the product.

When the team starts the next phase or iteration, the items chosen for the new version are moved to the Product Board. The Product Board should include at least the main software development life-cycle steps such as analysis, design, development, functional system test, user acceptance test, and the final step named as done corresponding for the compliance with the definition of done. After the user acceptance test is completed, the next step is the deployment of the new version of the product into production. Depending on the granularity required for tracking, other steps can be added. The steps are connected to the DevOps pipeline. The new version of the product is planned in the DevOps pipeline. All software development activities are based on this version. Software components and their own versions are linked with the product version. This way, a holistic configuration management is executed for the product covering all of the artifacts and their related footprints in SPSM. Thus, any product version can be linked with the software component versions and product documentation versions as well as the Backlog items. Code repositories are managed based on this versioning approach.

The DevOps pipeline must provide at least the development, integration and user acceptance test environments. More environments such as staging and performance testing can also be included based on the nature of the product. There can be quality gates between these environments to satisfy the pre-requisite of the next environment. When the user acceptance test is finished, there should be a quality gate before deployment into the production. In this gate, the new version of the product is approved for release into the production. This gate should include control elements to check the version for its readiness for production. The control elements are related to the SPSM attributes, and should have threshold levels to open the gate. For example, unit test coverage ratio can be a control element, and if the threshold level is set to 70%, then each new version will be will only pass the gate once they satisfy this level. Control elements can be defined on the DevOps pipeline and values can be collected automatically with the pipeline tools. Throughout the DevOps pipeline, in the development phase, the metrics are defined and collected based on the system attributes and related control elements. In the post-development phase, the product performance metrics are collected in a manner similar to the development phase. All the collected data throughout these two phases are based on the product optimization attributes. Therefore, SPSM supports a dynamic DevOps pipeline based on the chosen attributes. The prerequisites between the pipeline environments and the quality gate at the end are defined dynamically. For example, if the customer is concerned about the quality of the product, test related metrics can be added into the control elements, such as unit test coverage. If the customer wants more reliability, the unit test coverage ratio can be increased to 90%. If cyber-attacks are getting frequent, then security becomes the leading attribute and vulnerability measurement via proxy metrics will be the new focus. Here the term vulnerability is used for deficits in the specifications, development, or configuration of a software that, if it occurs, violates the security policy of the software [30]. Moreover, vulnerability-related code metrics will be added into the control elements. If the customer has time-to-market concerns, then production speed becomes the leading attribute and cycle-time will be measured and optimized for the whole SPSM.

After releasing the new product version and deploying it into production, customer satisfaction level is collected with regular surveys. The surveys are organized as questionnaires with electronic survey tools available in the marketplace. Answers for the questions are collected numerically with the help of a Likert-type scale. The last part of the survey is used for the computation of the net promoter score [31]. The outcomes are then fed into the product backlog. The change in the customer satisfaction level after new versions should be analyzed for any correlation with any of the SPSM metrics. The product quality results coming from questionnaires are then compared with the DevOps pipeline metrics. If a survey indicates a decrease in the product quality from customer point of view, then pipeline metrics for quality should be analyzed according to the discovered correlation. If a metric shows a meaningful correlation with survey results, then that metric will have higher contribution to the pipeline quality gates.

Besides customer survey metrics, the product is also monitored in production and data is collected. Typically, in the production environment availability related metrics are monitored. However, in SPSM, the leading system attribute needs feedback from the production environment as well. If quality is the leading attribute, then a change in the severity and density of incidents after the release of the new versions will provide valuable feedback into SPSM. The correlation between incident data and SPSM quality metrics will help SPSM to adjust the control elements at the gate. Moreover, if any corrective or preventive action is required, the corrective or preventive action should be sent to the product backlog with measurements. The operations required for the product in the production environment are traced and send into the product backlog. This data driven feedback mechanism will let the product improve continuously.

The main components of SPSM in the DevOps pipeline is shown in Fig. 3. During software development, software quality is monitored via tools. There are a variety of tools in the market that can be integrated to the DevOps pipeline. Another component is software security which can be realized with the help of tools in the market and can also be integrated to the pipeline. Test monitoring is the third main component for tracking the system, integration and other tests. The fourth component is user acceptance as part of getting formal validation from customers or their representatives. The last component is labeled as product documentation which may vary sector to sector. These five components form the minimum set. Other components can be inserted depending on the customer value orientation of SPSM.

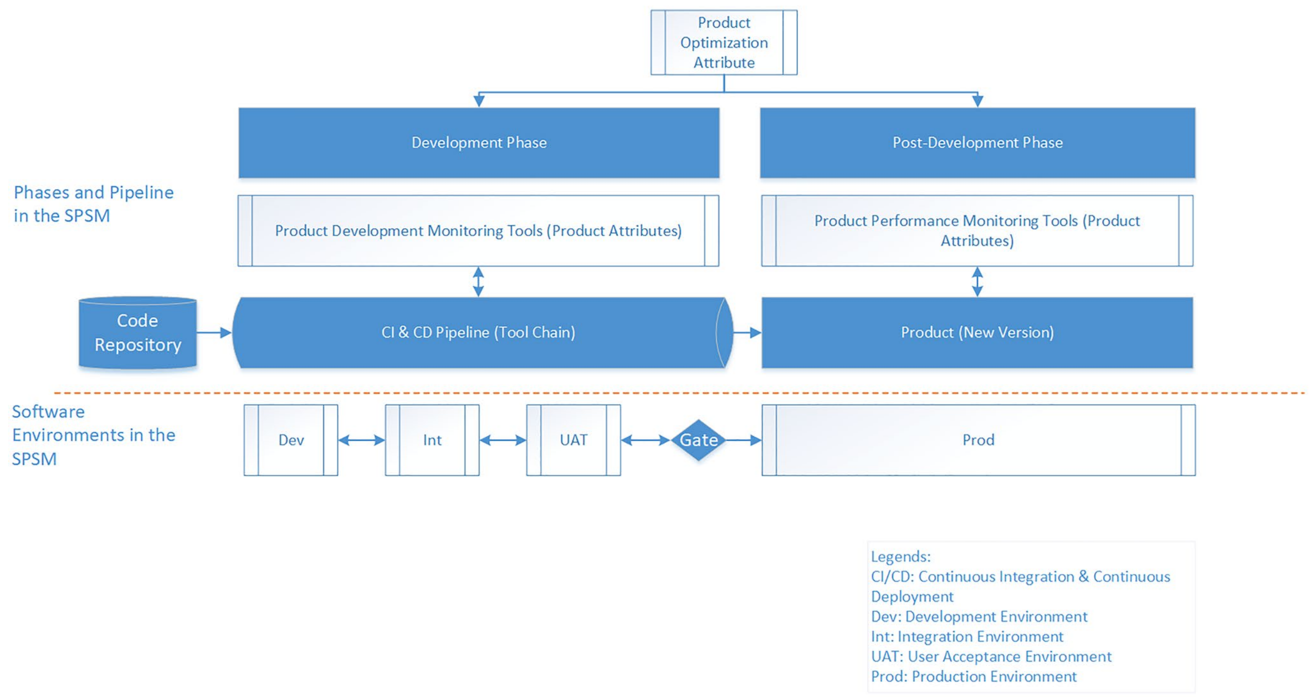

Product Optimization Attribute
Development Phase
Post-Development Phase
Phases and Pipeline in the SPSM
Product Development Monitoring Tools (Product Attributes)
Product Performance Monitoring Tools (Product Attributes)
Code Repository
CI & CD Pipeline (Tool Chain)
Product (New Version)
Software Environments in the SPSM
Dev
Int
UAT
Gate
Prod
Legends:
CI/CD: Continuous Integration & Continuous Deployment
Dev: Development Environment
Int: Integration Environment
UAT: User Acceptance Environment
Prod: Production Environment

Fig. 3 SPSM system architecture

Implementation and Deployment into the DevOps Cycle

The SPSM model was realized with pilot implementations during 2016–2017 and was put into widespread use in 2018 in a large-scale software company. The usage of SPSM was within a software company with 19 product groups and 56 product development and operations teams in the banking and finance industry and a work force of around 900 people working on the product development pipeline. The company was developing web-based enterprise applications as well as mobile applications in Android and iOS environments. The leading software language was Java, followed by C#.net. There were other languages and software development frameworks as well but their shares were small compared to the leading languages. Product development and operations teams were composed of a team lead, a product architect, back-end and front-end developers, business analysts, test experts, and a DevOps engineer. Each team was between 5 and 12 people. The average team size was 8 people with a mean age of 34.2 years. The teams were responsible for a single product or part of a large enterprise product. Teams were equipped with the technical skills that were required for the development and deployment of the software. In 2019, there was an enterprise level agile transformation, mainly scrum with its practices. The adoption of the agile methodologies finished in July 2019. Since that point, every team has been using scrum with product backlog and boards in Jira™. Each team also has a volunteer scrum master within the team focusing on scrum practices as well.

Each product had a product manager responsible for the product strategy, roadmap, backlog management as well as customer satisfaction. Product managers were also tasked with approving the product before deploying it into the production environment. Each product had a support expert or support team responsible for the solution of incident tickets occurring in the production. Security was considered a common knowledge base for all development & operations teams. Therefore, every team member was given basic software security training. Besides that, there was a separate security team which is located as a security center of excellence that answers security related questions from team members and advising and monitoring security related topics.

The model is implemented with two main parts: the product-based production process and the DevOps pipeline. Continuous deployment is exercised for all products of the company. The DevOps pipeline is established on premise with centrally decided tools. Tools are connected and made ready for the usage of development & operations team before SPSM model is utilized. All codebase is managed on the premise's central repo in Nexus™. The process is built on Jira™, where customers could send and track their requests for the products. This process is connected to the DevOps pipeline which is built on the XebiaLabs™ product family for the coordination of releases and automated deployment, Team Foundation Server™ for configuration management and Jenkins™ for automated building in the security step. Due to the security requirements of the main customer which is one of the largest banks in Turkey, the whole of SPSM has been optimized to provide more secure software products. To manage the pipeline application security levels, the static code analysis tool Checkmarx™ has been integrated. Since the security checks of the software components took time, the tool is adjusted to trace the code base asynchronously on Jenkins. The security metrics are reported via pipeline dashboard automatically on a regular basis. The product teams can monitor their findings and fix them during their development period. Once a new code is developed, the continuous integration steps of DevOps pipeline added the code to the products code base. The code base is traced regularly and detected security vulnerabilities are prioritized by the tool as high, medium and low according to Open Web Application Security Project (OWASP) and reported in the pipeline dashboard [32]. Product teams are responsible to resolve them. In addition, a DevOps security control element is added into the gate which is located just before the new product version is approved for release into the production. If SPSM is focused on software security, then this control element can be activated. In this control element, high level vulnerabilities of each software product are compared with their previous versions, and if any increase is detected, then gate is closed. That means, the new version of the software product is automatically rejected to be deployed into the production until the high-level vulnerabilities are less or equivalent to the previous version. In other words, this control element in the gate forces product teams to focus on security as the main attribute in SPSM for this particular software production environment, whereas other attributes could become the main focus for another environment.

Other control elements are also defined according to Cobit standards which defines the regulations for information technology governance of the banks in Turkey [33]. These include the two control elements for verifying whether the tests are completed and user acceptance from the customer side is achieved. Besides that, extra control elements are defined for validating the product versioning and related documents. In these control elements, each product is checked for the main set of documents; analysis, architecture, and user manual being the minimum set of documents for each product.

The implementation phase within the company required the integration of tools for the control elements. For the internal quality of the software, Sonarqube™ is utilized with the measurements on unit test coverage and rule compliance index. This control element checks the level of these two metrics; however, it does not act as the show stopper for the pipeline. The product teams can monitor their unit test coverage automatically for each new deployment. If for another customer SPSM is focused on unit testing, then this control element can act as the main gate for the deployment for rejecting the product versions with lower unit test coverage than the customer defined value. This value may change depending on the customer and there is no industry standard [34].

The customer satisfaction level is measured by the surveys as introduced in the previous section. The surveys are composed of four parts and have a total of 12 questions with multiple choices reflecting the customer satisfaction level. Surveys are organized to measure in the form of system usability scale (SUS) [35]. The surveys have five choices with the help of a Likert-type scale. The questions cover the fulfillment of requirements, product quality, product support quality, and company reputation. Overall satisfaction level is measured by the survey responses to each question.

Evaluating the Use of SPSM

In this section, we evaluate the use SPSM within the company's DevOps cycle for 2 years with relevant metrics as well as discussing the present case study from the perspective of empirical validity.

Monitoring and Improving the Use of SPSM

During 2019, this pipeline was utilized and operated for all the products of the bank with the goal of improving their security level. The outcome of this utilization is a 60.39% decrease of security vulnerabilities of all software products delivered to the bank at the end of 2019. This decrease covers all the vulnerabilities of the code base of products that have been developed over many years. The code base is around 25 million lines with different coding languages with two main groups in Java and C#. Around 20% of the code base is greenfield, meaning that it is a newly developed product with no legacy code, while others are mainly additional features for live products that are in one of the introduction, growth, maturity, or decline stages.

The security focus as a quality attribute has had almost no negative effect on the timely and complete delivery of contracted software products. At the end of 2019, project completion ratio was 98.11% with a Schedule Performance Index (SPI) of 99.23% as the average of 684 projects delivered. Major decrease in security vulnerabilities as well as successful delivery of the products can be explained by effectiveness of the security focus in SPSM implementation in the company. Since security is the main goal and the control element is adaptable as explained in the previous section, all product teams should focus on vulnerabilities before pushing the code in the pipeline into the production.

In 2020, in addition to the security focus, the bank asked for an increase in the code quality. The reason behind this requirement was the motivation to decrease the number of incidents in the production environment. Unit test coverage was chosen as an indicator of the code quality. The challenge was to increase unit test coverage for the whole code repository. That means SPSM model needed to optimize the system for both security and the code quality while keeping the project completion ratio above 95%. As the first step a dashboard was added into SPSM to show the security and code quality metrics of each software package in real time. This way each product team could easily monitor their progress. In addition, a self-service scan capability was added to the dashboard that enabled product teams to start security or quality scans synchronously. This way teams can scan their own codes without waiting for the next deployment. At the end of the 2020, the DevOps pipeline provided a 69.50% decrease in security vulnerabilities of all software products, while a 29.43% increase in unit test coverage for the whole code base was realized. Baseline values taken at the beginning of the year were 91,348 vulnerabilities and a 20.35% unit test coverage. The number of new incidents per month was also monitored and at the end of 2020, a 17% decrement was realized when compared to the previous year based on the total annual 120,000 incidents taken as the baseline at the beginning of the year. By the end of 2020, project completion ratio was measured to be 99.50% and the Schedule Performance Index (SPI) as 99.78% as the average of 762 projects delivered. Projects that could not be finished within the same year were carried over to the next year and counted as failures. Even though the ratio was small, the main reason behind them was the inability to adapt to changing customer values and requirements.

At the end of 2019 product satisfaction surveys reached the value of 3.76, while at the end of 2020, the customer satisfaction level was 4.05. Additional unit test coverage has a positive correlation with the increment.

During the adoption period there were mainly three issues to cope with. The first one was the cultural change of the style of work. The formation of new teams around products and focusing on the same picture of customer value and product success was challenging for the development and operational teams that had worked separately until that point. This issue was resolved with the help of product managers who set the product goal based on customer value. The second issue was the migration of legacy code base to the new tool chain. Some parts of the old legacy code were not supported by the new tool chains, which required the legacy code to be converted to the new code base when necessary. The third issue was adaptability of teams to changing customer expectations while maintaining their pace. Even though SPSM and tool chains promote adaptability, mental agility takes some time to adapt to changing priorities.

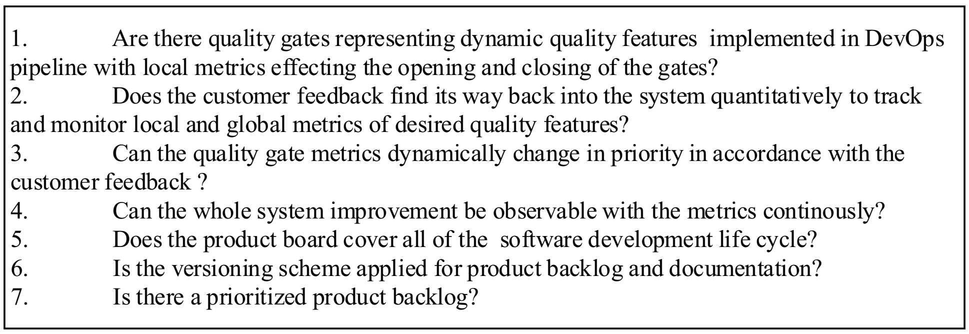
Threats to Validity

The current research can be loosely positioned within the Design Science Research paradigm [36], an industrial case study forming the main implementation and evaluation context for the suggested DevOps System Model, SPSM. In this section, we will address the main concerns regarding the validity and reliability of the study from an empirical perspective [37].

Construct validity is concerned with how much the theoretical model and the interpretation and operationalization of its application matches. What is assessed has to correctly correspond to the theoretical construct [37, 38]. The primary author of the present article was also the responsible director for process improvement in the company forming the current case study, thus SPSM's implementation and evaluation was realized over 2 years with synchronous methodological validation and verification directly matching the conceptual model.

Internal validity is about the strength of the causal and logical relationship between the variables of the research and the conclusions drawn [37]. The present case study is about a complex software development environment with a large code base, and a large number of developers. Clearly distinguishing what is caused by the properties of the SPSM deployment in contrast to other processes such as specifics of agile processes in the company can be hard to do. We acknowledge that the metrics chosen for the evaluation were consistent with dynamic monitoring and the triangulation of the data for the evaluation of the effectiveness of SPSM in line with the policies of the company, but they were not intended as an exclusive subset that would be relevant only to the performance of SPSM. Moreover, some possibly confounding factors, such as the effect of coronavirus pandemic, and hence, the transfer of the company to remote work during the latter half of the application of SPSM also has to be evaluated separately as future work.

We will treat external validity, the extent of generalizability for a successful deployment of SPSM in other industrial cases, and reliability, the transparency showing the correctness of measurements for evaluation and mechanisms for the replicability of the study, together, as the factors concerning both are common. There are no case study protocols and databases for the replicability of the study regarding the nature of the company-internal information that such empirical replicability and reliability tools would involve. However, the company has well established procedures and internal quality systems in accordance with ISO9001:2015 for metric collection, evaluation, and continuous improvement. Moreover, we drew up a general checklist, shown in



1. Are there quality gates representing dynamic quality features implemented in DevOps pipeline with local metrics effecting the opening and closing of the gates?
2. Does the customer feedback find its way back into the system quantitatively to track and monitor local and global metrics of desired quality features?
3. Can the quality gate metrics dynamically change in priority in accordance with the customer feedback ?
4. Can the whole system improvement be observable with the metrics continously?
5. Does the product board cover all of the software development life cycle?
6. Is the versioning scheme applied for product backlog and documentation?
7. Is there a prioritized product backlog?

Fig. 4 Checklist for SPSM application in DevOps Framework

Fig. 4, for other organizations who want to implement and evaluate SPSM in a DevOps context to mitigate risks in replicability and external validity.

Concluding Remarks: Implications of SPSM for DevOps

We have proposed a systems-based, customer-value oriented, dynamic System Model, SPSM, over a DevOps pipeline with quality gates and with feedback from both customers and local metrics. Our purpose is to enhance DevOps pipelines' interaction with software development processes without giving up DevOps' advantages in integration and production speed. The implementation and evaluation of the model over 2 years in a large software house improved the quality needs of the particular customer focus, with further improvement of key project performance indicators. SPSM has allowed for a better adoption of agility concerning customer expectations and changes in customer focus by allowing dynamic fine tuning with respect to product attributes.

SPSM as a System Model overlaid on a DevOps pipeline has also allowed local optimums on desired attributes to work better, being integrated to the whole system optimization by bringing product log, DevOps, documentation and maintenance all under the same framework and making causal links explicit. The adoption of SPSM with extra quality gates has not been detrimental to delivery rates; one particular reason for that in this particular implementation has been the firmness of the process owners in the non-flexibility of the quality gate control levels, even though such requests and even complaints on possibly overriding the gates in the name of urgency of a certain feature implementation have been presented. Such rebounds are likely to happen but management support in such an implementation is of importance so that developer teams have a chance to adjust to expectations and will know that they cannot be bypassed.

Future work could involve more empirically controlled application and evaluation of SPSM in other software company settings. Since the adoption and acceptance of the model characteristics by the developer teams are crucial, one way to go could be in the direction of making the adoption of the model for the product teams easier; possibly supporting how the quality emphasis in the current model could be enhanced through alternate means, such as gamification.

Author Contributions HA is the lead designer and the application coordinator for Software Product System Model (SPSM). BS helped put the design and deployment process into a research framework. Both contributed to the writing of the article.

Funding Not applicable.

Availability of Data and Materials (Data Transparency) Not applicable.

Code Availability (Software Application or Custom Code) Not applicable.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest. Dr. Haluk Altunel has been working for Softtech Inc.as Director of Strategy and Product Management.

References

1. Leite L, Rocha C, Kon F, Milojicic D, Meirelles P. A survey of DevOps concepts and challenges. ACM Comput Surv. 2019. https://doi.org/10.1145/3359981.
2. Ebert C, Gallardo G, Hernantes J, Serrano N. DevOps. IEEE Softw. 2016;33(3):94–100. https://doi.org/10.1109/ms.2016.68.
3. Kittlaus H, Samuel AF. Software product management. Berlin: Springer; 2017. https://doi.org/10.1007/978-3-642-55140-6.
4. Altunel H. Agile project management in product life cycle. Int J Inf Technol Proj Manag. 2017;8(2):50–63. https://doi.org/10.4018/ijitpm.2017040104.
5. Bolscher R, Daneva M. Designing software architecture to support continuous delivery and DevOps: a systematic literature review. In: Proceedings of the 14th international conference on software technologies (ICSOFT); 2019. pp. 27–39. https://doi.org/10.5220/0007837000270039.
6. Kersten M. A cambrian explosion of DevOps tools. IEEE Ann Hist Comput. 2018;35(2):14–7. https://doi.org/10.1109/ms.2018.1661330.

7. Lwakatare LE, Kilamo T, Karvonen T, Sauvola T, Heikkilä V, Itkonen J, Kuvaja P, Mikkonen T, Oivo M, Lassenius C. DevOps in practice: a multiple case study of five companies. Inf Softw Technol. 2019;114:217–30. https://doi.org/10.1016/j.infsof.2019.06.010.
8. Badshah, S, Khan AA, Khan B. Towards process improvement in DevOps: a systematic literature review. In: EASE'20: 24th International conference on evaluation and assessment in software engineering 2020, April 15–17, 2020. Trondheim, Norway. https://doi.org/10.1145/3383219.3383280.
9. Céspedes D, Angeleri P, Melendez K, Dávila A. Software product quality in DevOps contexts: a systematic literature review. In: Trends and applications in software engineering. CIMPS 2019. Adv Intell Syst Comput. 2020;3:51–64.
10. Hsu T. Hands-on security in DevOps. Birmingham: Packt Publishing; 2018.
11. Düllmann TF, Paule C, van Hoorn A. Exploiting devops practices for dependable and secure continuous delivery pipelines. In: 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE), 2018. pp. 27–30.
12. Haindl P, Plösch R, Körner C. An extension of the QUAMOCO quality model to specify and evaluate feature-dependent non-functional requirements. In: 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Kallithea-Chalkidiki, Greece, 2019. pp. 19–28. https://doi.org/10.1109/SEAA.2019.00012.
13. Bezemer P, Eismann S, Ferme V, Grohmann J, Heinrich R, Jamshidi P, Shang W, van Hoorn A, Villavicencio M, Walter J, Willnecker F. How is performance addressed in DevOps? In: Proceedings of the 2019 ACM/SPEC International conference on performance engineering (ICPE '19). New York, NY, USA, 2019. pp. 45–50. https://doi.org/10.1145/3297663.3309672.
14. Weinberg GM. An introduction to general systems thinking. Anniversary edition. New York: Dorset House Publishing; 2001.
15. Senge P. The fifth discipline: the art and practice of the learning organization. 2nd ed. New York: Doubleday Press; 2006.
16. Wendorff P. An essential distinction of agile software development processes based on systems thinking in software engineering management. In: Third international conference on eXtreme programming and agile processes in software engineering, Alghero, Sardinia, Italy, 2002.
17. Osterman C, Fundin A. A systems theory for lean describing natural connections in an XPS. TQM J. 2020;32(6):373–1393. https://doi.org/10.1108/TQM-12-2019-0284.
18. Murphy N, Beyer B, Jones C, Petoff J. Site Reliability Engineering: How Google runs production systems. Newton: O'Reilly; 2016.
19. Kim G, Humble J, Debois P, Willis J. The DevOps handbook: How to create world-class agility, reliability, and security in technology organizations, IT Revolution, 2016.
20. Forsgren N, Kersten M. DevOps metrics. Commun ACM. 2018;61(4):44–8. https://doi.org/10.1145/3159169.
21. Maroukian K, Gulliver S. Exploring the link between leadership and Devops practice and principle adoption. Adv Comput: Int J. 2020. https://doi.org/10.5121/acij.2020.11401.
22. Franklin GF, Fowell JD, Emami-Naeini A. Feedback control of dynamic systems. 8th ed. London: Pearson; 2018.
23. Anon J, González de Villaumbrosia C. The product book. Product School, 2017.
24. Wagenblatt T. Software product management fundamentals. In: Software Product Management. Management for Professionals. Cham: Springer; 2019. https://doi.org/10.1007/978-3-030-19871-8_1.
25. Feijter R, et al. Towards the adoption of DevOps in software product organizations: a maturity model approach. Technical Report Series UU-CS-2017–009, 2017.
26. Babar Z, Lapouchnian A, Yu E. Modeling DevOps deployment choices using process architecture design dimensions. The Practice of Enterprise Modeling. Berlin: Springer; 2015. p. 322–37. https://doi.org/10.1007/978-3-319-25897-3_21.
27. Berger A. Continuous improvement and kaizen: standardization and organizational designs. Integr Manuf Syst. 1997;8(2):110–7. https://doi.org/10.1108/09576069710165792.
28. ITIL, 4th Edition. https://www.axelos.com/welcome-to-itil-4. Accessed July 2021.
29. Agile Alliance Web Site. https://agilealliance.org. Accessed Mar, 2021.
30. Bulut FG, Altunel H, Tosun A. Predicting software vulnerabilities using topic modeling with issues. In 2019 4th International conference on computer science and engineering (UBMK). IEEE, 2019.
31. Reicheld FF, Sasser WE Jr. Zero defections. In: Lewis M, Stack N, editors. Operations management: critical perspectives on business and management, vol. 105. Milton Park: Routledge; 2003. p. 289–98.
32. Open Web Application Security Project. https://owasp.org/. Accessed Mar 2021.
33. COBIT: Control Objectives for Information Technologies, an ISACA framework. https://www.isaca.org/resources/cobit. Accessed Mar 2021.
34. Zhu H, Hall PAV, May JHR. Software unit test coverage and adequacy. ACM Comput Surv. 1997;29(4):366–427. https://doi.org/10.1145/267580.267590.
35. Drew MR, Brooke F, Baccus WL. What does the system usability scale (SUS) measure?. In: International conference of design, user experience, and usability. Cham: Springer; 2018. https://doi.org/10.1007/978-3-319-91797-9_25.
36. Hevner A, Chatterjee S. Design research in information systems. Boston: Springer; 2010.
37. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. Experimentation in software engineering. Berlin: Springer; 2012.
38. Yin R. Case study research and applications: design and methods. 6th ed. Los Angeles: SAGE; 2018.