Modeling DevOps Deployment Choices Using Process Architecture Design Dimensions

Zia Babar, Alexei Lapouchnian, Eric Yu

Modeling DevOps Deployment Choices using Process Architecture Design Dimensions

Zia Babar[1], Alexei Lapouchnian[2], Eric Yu[1,2]

[1]Faculty of Information, University of Toronto
[2]Department of Computer Science, University of Toronto
zia.babar@mail.utoronto.ca, alexei@cs.toronto.edu,
eric.yu@utoronto.ca

Abstract. DevOps is a software development approach that enables enterprises to rapidly deliver software product features through process automation, greater inter-team collaboration and increased efficiency introduced through monitoring and measuring activities. No two enterprise-adopted DevOps approaches would be similar as each enterprise has unique characteristics and requirements. At present, there is no structured method in enterprise architecture modeling that would enable enterprises to devise a DevOps approach suitable for their requirements while considering possible process reconfigurations. Any DevOps implementation can have variations at different points across development and operational processes and enterprises need to be able to systematically map these variation points and understand the trade-offs involved in selecting one alternative over another. In this paper, we use our previously proposed Business Process Architecture modeling technique to express and analyze DevOps alternatives and help enterprises select customized DevOps processes that match their contexts and requirements.

Keywords: Enterprise Modeling, Software Processes, Business Process Modeling, DevOps, Goal Modeling, Adaptive Enterprise.

1 Introduction

Enterprises are expected to continuously respond to ongoing changes and evolving environmental factors. Increasing competition and emergence of new market players from non-traditional sectors require enterprises to react and adapt to change more quickly than ever before [1,2]. To this end, more and more enterprises are relying on software for the development and delivery of appropriate products and services. As a result, software processes are becoming an integral part of enterprise processes. Just like business processes (BPs), software development processes can vary significantly from organization to organization due to unique enterprise characteristics; these processes can be reconfigured in multiple ways to take account of enterprise variations and behavioral peculiarities so as to fulfill high-level enterprise requirements. However, current methods of modeling software process reconfigurations are limited in their ability to consider multiple enterprise perspectives and help choose among alter-

nate configurations. In this paper, we elaborate on the software process reconfigurations that are possible in the DevOps approach for the purpose of describing a Business Process Architecture (BPA) modeling technique, which allows the depiction and analysis of BP reconfigurations along multiple dimensions.

The term "DevOps" is a combination of two words "Development" and "Operations" and has been described and referred to as a phenomenon, a philosophy, a mindset, a set of techniques, a methodology, etc. DevOps is not a software tool or methodology per se, but rather an approach for rapidly and frequently delivering new software product features and service innovation. A recent Gartner news release predicted that "DevOps will evolve from a niche strategy employed by large cloud providers to a mainstream strategy employed by 25 percent of Global 2000 organizations" [3]. Broadly speaking, DevOps attempts to introduce rapid delivery of product features, services and bug fixes to end-users through frequent release cycles, each containing a small feature set. Rapid delivery enables an enterprise to reduce the time-to-market for new products and features, provides greater customer centricity by introducing new features based on evolving customer needs, quickly resolves operational and support issues, and shows greater responsiveness to changing (internal and external) environment situations. DevOps enables the above by [4,5,6]:

- Automating activities in the overall software development process through the introduction of software tools and custom development of scripts, thus shortening the time required for new feature development and bug fixes through reduction of manual effort. This enables software teams to deliver more frequent releases to customers and the user base.
- Using feedback loops for continuously improving software development processes and development of product features through the monitoring and measurement of various software process and technical metrics. These metrics are then interpreted and utilized for overall process improvement.
- Promoting a culture of collaboration and information sharing between multiple teams. The traditional approach of having organization silos with defined boundaries and handover points is discouraged, and team members are expected to collectively collaborate towards the attainment of enterprise objectives.

The above characteristics are not unique to DevOps, and indeed, are generally applicable to enterprises with respect to enterprise agility and enterprise digital transformation [7]. Looking at software processes can provide insights into a broader context, such as the development and evolution of new products and services, many of which are digitally enabled. A BPA needs to be understood through a combination of these ideas and concepts, particularly in light of enterprise requirement for greater responsiveness and adaptability, with DevOps being a suitable example for such a study.

This paper is organized as follows. In Section 2, we introduce a DevOps-based motivating example that allows us to discuss the core concepts of this paper. In Section 3, we model a typical DevOps implementation using the BPA modeling technique and indicate possible areas and dimensions of software process variability. In Section 4, we refer to the related work, while in Section 5 we outline future directions of this research. Section 6 concludes this paper.

2 Motivating Example

DevOps is an interesting challenge for enterprise modeling for a number of reasons. As described above, DevOps involves diverse considerations from the viewpoints of process design, systems and tools development and deployment, and social and organizational issues. Continuous Integration (CI) and Continuous Deployment (CD) of product functionality and infrastructure setup are outcomes of DevOps [6]. The general area of continuous software engineering, CI and CD has been covered in both academic and industry literature with numerous published case studies [8]. Through CD, "companies could benefit from even shorter feedback loops, more frequent customer feedback, and the ability to more accurately validate whether the functionality that is developed corresponds to customer needs and behaviors" [8]. Studying all facets of the DevOps approach is thus best done through the enterprise modeling lens enabling a multi-perspective understanding of the various considerations.

Analyzing and deciding between various DevOps process reconfigurations can be done by considering enterprise objectives and benefits, which can be interpreted as functional requirements (FRs) and non-functional requirements (NFRs) from a process design perspective. The use of NFRs (represented by softgoals) in the requirements engineering discipline to evaluate and decide between variations and reconfigurations is well established [9]. Some of the NFRs, as present in a typical DevOps adoption, would be:

- **Agility and Adaptability:** Rapidly adapting to changing circumstances such as evolving customer behavior, regulatory environment, emerging technologies, etc.
- **Responsiveness:** Quickly responding to user feedback and change requests in the form of new product features and bug fixes.
- **Speed and Frequency:** Delivering new product features and bug fixes faster as well as having a high deployment frequency.
- **Efficiency:** Improvement in software process execution by automating key process segments and increasing collaboration between team members for greater information flow.
- **Customizability:** Being able to customize the behavior of the software development lifecycle based on changing contextual and situational needs.

Fig. 1 shows a simple BPMN [10] process model indicating the primary participants and the major activities in a typical DevOps-inspired software process. We have developed this context by referencing published literature from multiple sources, such as [8,11,12,13,14], with the intention of highlighting how the various process activities in DevOps can be better configured to serve a variety of enterprise FRs and NFRs.

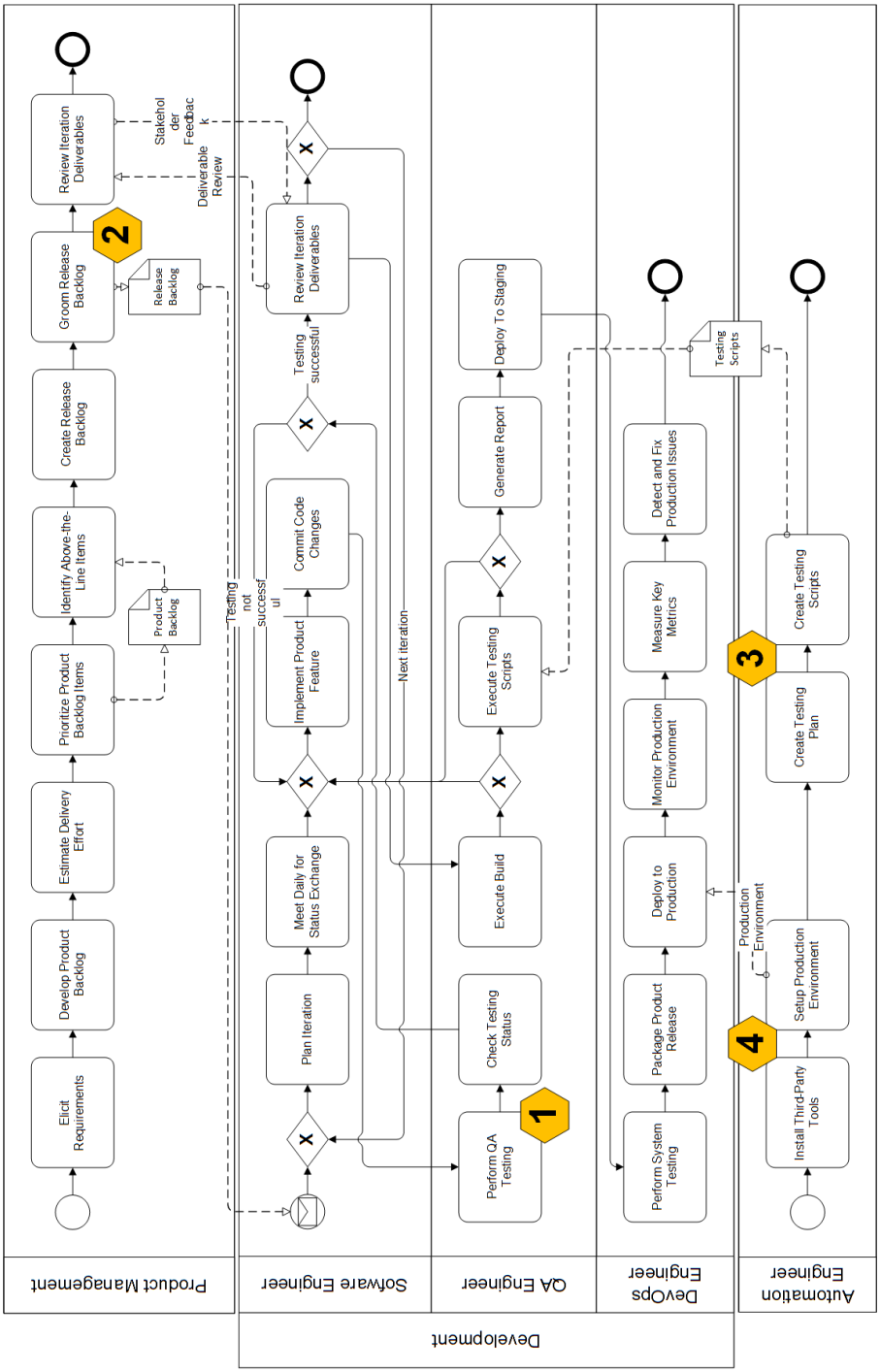

Fig. 1. A simple BPMN model representing a typical DevOps approach

In DevOps, the development of product features can be done using different development methodologies while adhering to different practices and policies specific to an enterprise adoption; in this context we assume the use of the Scrum project management methodology [14]. However, this general DevOps context is not intended to be an exhaustive depiction of variations in DevOps adoption in an enterprise setting, but rather is meant to illustrate variability in software process configurations. We consider four scenarios of variable behavior in this contextual setting, which correspond to the numbered annotations in the BPMN diagram:

1. **QA Testing:** Any developed feature has to be functional tested before it goes through the CD process. This testing can be carried out by QA engineers in at least two ways: they can retrieve the committed code from the code repository and test it on a test environment, or alternatively, they can collaborate with the software engineer to quickly validate the functionality before the codebase is committed to the code repository.
2. **Release Planning:** The enterprise is assumed to have periodic and fixed release cycles of appropriate duration. A release planning activity is carried out at release initiation that results in a *release backlog*; this artifact is then used to plan out individual sprint iterations. Two of the possible alternatives are 1) the release backlog is produced once and remains static throughout the release duration and 2) the release backlog is revisited at the beginning of every sprint and "groomed" (i.e. reordered and re-estimated) based on on-going change in circumstances and priorities.
3. **Automated Testing:** In order to reduce product delivery durations, some product testing can be automated by developing test plans that are then scripted for execution as part of the CI process. The test scripts can be developed once and reused for subsequent CI activities or they can be developed every time to serve specific testing needs based on the product feature being tested.
4. **Tool Usage for Automation:** DevOps is characterized by the usage of third-party tools for CI and CD, server configuration, infrastructure provisioning, deployment management, etc. These tools are configured for use repeatedly without requiring the knowledge of their inner working. This is depicted in Fig. 1 as a separate Automation Engineer pool to visually differentiate it from the on-going DevOps Engineer activities.

The BPMN model in Fig. 1 allows a visual understanding of the sequencing of process activities and the flow of information between them. However, BPMN process modeling is lacking in terms of the selection and evaluation of alternative DevOps configurations. In any enterprise, there would exist multiple process levels, with processes at one level feeding into those at an upper level. The multiple levels of process-driven dynamics and the relationships between the process levels are not apparent in the BPMN model nor are boundaries between these process levels obvious. Multiple BPs may come together to provide some feature functionality (for example, the development of test plans and their execution are part of two separate BPs), but the nature of their relationship is not explicit in the model. While process activities can be shown, along with the changes in their sequencing, the implications of any activity reordering cannot be determined. Similarly, enterprises rely on sense-

and-respond loops to continuously improve their operational processes [15]. While the BPMN model in Fig. 1 does show such feedback loops, the full range of attributes associated with them (for example, the multitude of timescales present in the loop or the execution frequency of the sensing and responding parts) are not evident.

3 Modeling Process Reconfigurations

The BPA modeling framework was introduced in [16,17] for assisting with the modeling of BPs, their relationships, and the flexibility afforded by various BPA configurations. We use this framework to evaluate various DevOps reconfigurations and to choose among them. Fundamental concepts in the BPA framework are that of *Process Element* (PE), *Variation Point* (VP), *Stage* and *Phase* [16].

- A PE is defined as "an activity that produces some output or outcome. It may also include the act of making decisions".
- A VP is referred to "the point in a process where multiple options exist. Variation points may appear anywhere in a process".
- PEs are grouped together in process Stages if they are executed together as part of the same execution cycle. A *stage boundary* exists between two stages and PEs can be moved across stage boundaries as required while considering different trade-offs.
- A stage may contain one or more Phases, which are sections of a stage that are the "portions of a process such that placing a PE under consideration anywhere within a phase produces the same result…However, moving PEs across *phase boundaries* may affect the quality of decisions and the outcome of actions".

A PE can be repositioned along four dimensions in any process architecture. These four dimensions include, (1) *the temporal dimension* – positioning a PE either before or after other PEs (with respect to sequence of execution), (2) *the recurrence dimension* – positioning a PE in a stage that is executed more frequently or less frequently compared to other stages, (3) *the plan-execution dimension* – positioning a PE in a stage that either is responsible for planning or responsible for the execution of that plan, (4) *the design-use dimension* – positioning a PE in a stage that either is responsible for designing a tool, capability or artifact, or responsible for using the output of that design stage. These dimensions are discussed in more detail in the subsequent sub-sections.

Fig. 2 shows a BPA model for the DevOps approach with multitudes of process elements, stages, phases and the relationships among them. The model visualizes the key aspects of software development and operational support processes that are commonly present in the DevOps approach starting from the Product Management stage to the Operational Support stage. For the sake of comprehension and understandability, we conceptually divide the model into multiple sections and consider them individually with regards to the overall DevOps approach as follows:

- **Product Management:** Product FRs and NFRs are elicited and gathered from a variety of sources (such as User Input and Business Need) and consolidated together. This is then used to develop a Product Backlog, which is frequently groomed for estimating and prioritizing individual Product Backlog Items (PBIs). The grooming exercise is a periodic process that runs at a higher recurrence than the requirements elicitation activity, which is denoted by the recurrence relationship between the two stages.
- **Development:** The model depicts the Scrum project management methodology with the various rituals and iterations shown as part of the Release Planning and Sprint Cycle stages. Evidence of recurrence is apparent in the usage of the Product Backlog over multiple Release Planning iterations. The Perform QA Testing process element can be used to demonstrate the temporal dimension as the testing can be done either before the code is committed to the source repository or after. Both options have different consequences as shall be seen in the section 3.1.
- **Automated Testing:** The DevOps approach promotes the usage of tools and scripts for automating the testing of product features. For this, test plans and test scripts are created and are then used to automate the testing effort, whereas test plans are implemented through test scripts. The test plans and test scripts are created in the Testing Plan stage and the test scripts are executed through the Execute Test Scripts PE (part of the Continuous Integration stage); these are illustrative of the plan-execute dimension.
- **Ongoing Deployment:** As with testing, the deployment of the developed product feature is ongoing, immediate and automated while factoring in the variable and multiple environments that the product would have to run on. The software deployment is automated through deployment scripts that are executed by various deployment tools; these scripts are developed by the DevOps engineers and executed as part of the Continuous Deployment stage that gets triggered on the successful completion of the Automated Testing and Continuous Integration stages.
- **Operational Support:** A major contribution of DevOps to software development is the breaking down of silos between the development and operational teams, thus fostering a culture of collaboration. The BPA models do not show process participants, so the collaborative aspect of DevOps is not visible. However, the Operational Support stage (along with the monitoring and measurement of operational metrics) is visually apparent, including the incorporation of software metrics into the product backlog (through a feedback loop) for ongoing software process lifecycle improvement.

The positioning of certain PEs in the DevOps approach are described in subsequent sub-sections along with the criteria for deciding among the options. Enterprises may want to analyze alternate positioning of PEs based on their FRs and NFRs. For this purpose, goal modeling can be used for representing the variations and helping select the appropriate alternative. NFRs are represented as softgoals and alternate methods of achieving a goal are represented as OR decompositions. Selection of a suitable alternative is made based on the positive and/or negative contribution(s) that the alternative would have on the NFRs (softgoals). The four scenarios described in the

previous section (which also correspond to each of the four process architecture dimensions) are presented, with goal models shown alongside the BPA model snippets. In all goal model examples, the root goal can be achieved through two alternate sub-goals. The choices are limited to just two for brevity and space reasons. A real-world situation could contain many possible choices, as well as many competing and complementary NFRs. Also, the goals are shown at a PE level and decomposed down to just one level. In the general case, the goal model would start from enterprise-level goals, with multiple levels of goal refinement and alternatives until PE level sub-goals are reached [9].

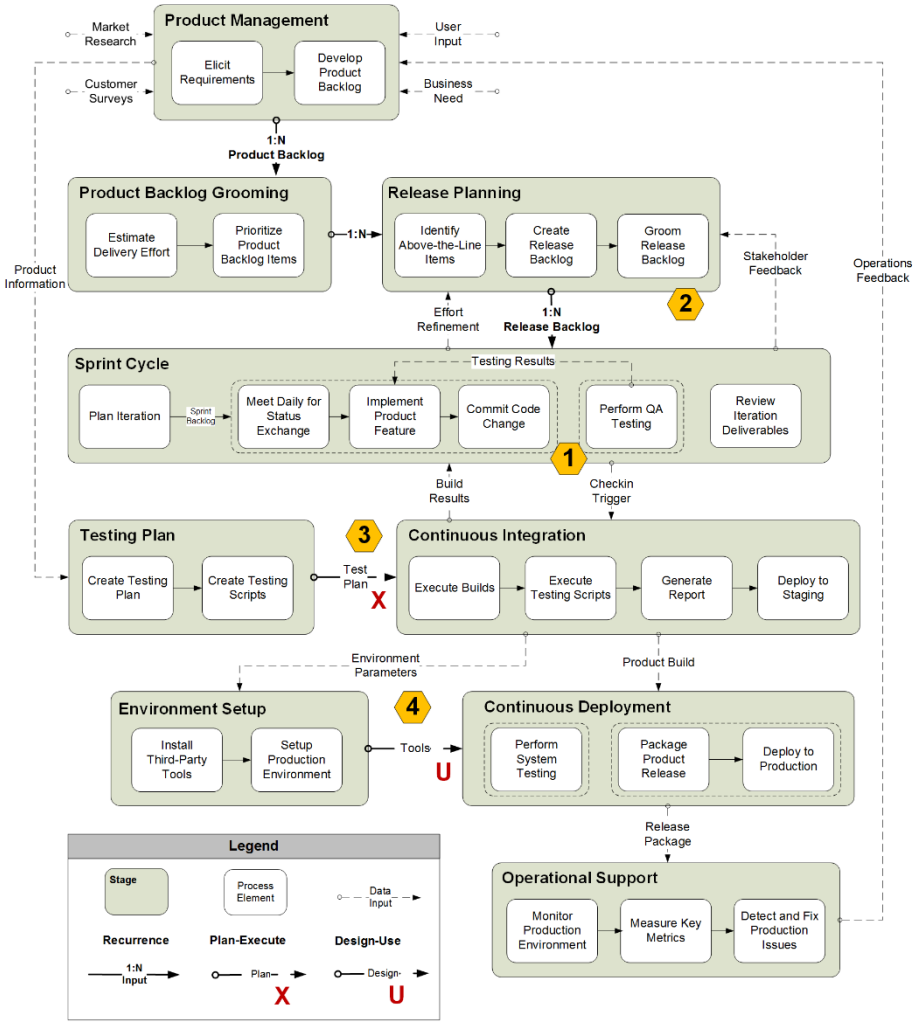

Fig. 2. Business Process Architecture (BPA) for a DevOps approach

3.1 The Temporal Dimension

The particular temporal placement of a PE can bring about certain benefits. A PE can either be advanced (and be executed) before other PEs or postponed after those PEs. Postponing a PE provides the benefit of executing it with the latest context and information available, thus reducing the risk and uncertainty that are inherent in any BP. The alternative is to advance the PE relative to other PEs, which reduces the complexity and cost as less effort is required to process the limited contextual information available at that instant. Uncertainty is also reduced. Therefore, the placement of any PE should be carefully considered with regards to various NFRs, subject to inherent temporal constraints among the PEs. The testing of a product feature by a QA engineer illustrates the trade-offs between advancing and postponing a PE (Fig. 3). The QA engineer can verify the developed feature (Perform QA Testing) after the software engineer checks in the code to the code repository (Commit Code Changes) or before the code is checked in by working directly with the software engineer. As shown by the goal model, the latter approach has the benefit of being collaborative in nature and encouraging both the software engineer and QA engineer to work together to solve the problem quickly. The former approach is more methodological and allows for the proper (and independent) validation of the feature and the tracking of testing issues. The appropriate order of the Perform QA Testing PE is determined based on the organization's prioritization between the softgoals.

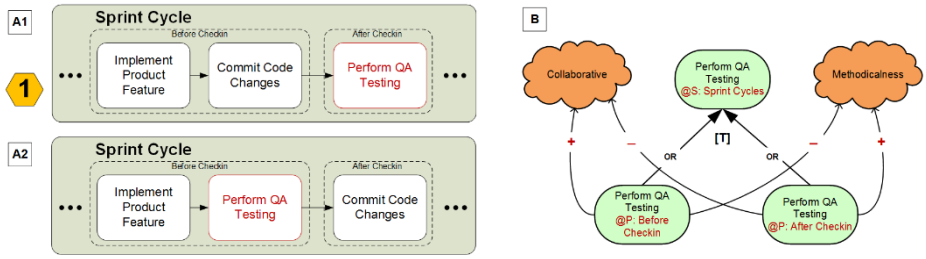

Fig. 3. QA testing alternatives (A1) As a separate phase from product feature implementation, (A2) As part of the product feature implementation phase. (B) Analyzing the temporal placement of QA testing process element based on NFRs.

3.2 The Recurrence Dimension

A recurrence relationship exists between the two stages of a process when the output of one stage can be used repeatedly (and without change) by the subsequent stage. A PE can be moved from a stage with a lower recurrence to one with a higher recurrence (and vice versa). Such a movement of the PE can change the non-functional properties of the BP in various ways. For example, reducing the PE recurrence saves cost as the same PE does not have to be executed repeatedly. Conversely, increasing the PE recurrence can assist with flexibility and adaptability as the PE is executed based on updated and current information.

In the DevOps approach, a product can be developed by having periodic and multiple product releases with many development sprints (within each release) required for attaining the release objectives (Fig. 4). Depending on the situation, an enterprise can create (Create Release Backlog) and groom a release backlog (Groom Release Backlog) once, which is then used for subsequent sprint planning. Alternatively, the enterprise can reassess the release objectives every time it starts a new sprint [13]. The former is a more methodological approach and ensures that the enterprise is aligned to what the release deliverable is going to be, whereas the latter enables the enterprise to adapt to changing priorities by constantly reviewing the release delivery items. The enterprise can decide to go with either approach based on NFRs such as methodicalness, stability, cost, adaptability, flexibility, etc., by moving the Groom Release Backlog between the Sprint Cycle and Release Planning stages.

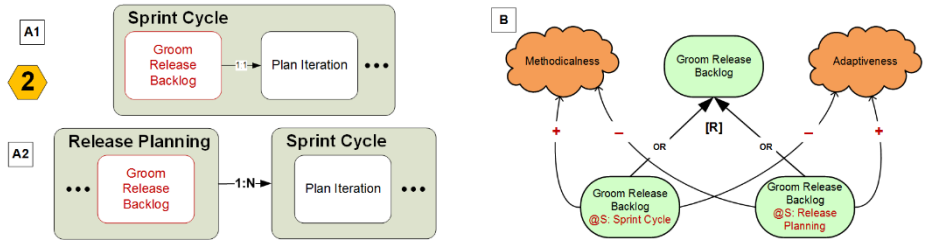

Fig. 4. Release backlog grooming alternatives (A1) As part of the Sprint Cycle stage with no recurrence, (A2) Moved to the Release Planning stage with a multi-recurrence dimension between both stages. (B) Analyzing the recurrence arrangement of the release backlog grooming and sprint planning stages based on NFRs.

3.3 The Plan-Execute Dimension

A BP can be considered to have two distinct segments, where one segment is responsible for creating a plan, which the other segment would then execute one or many times. Here, a plan-execute relationship exists between the two segments of the process. In the BPA modeling technique, each segment is modeled as a stage, with the stage producing the plan being the *planning stage* and the stage executing it being the *execution stage*. PEs can also be moved from an execution stage to a planning stage (and vice versa) based on the goal-driven analysis of their contribution to the relevant NFRs. Such movements create variations in the plan-execute behavior and allow either increased pre-planning (by moving a PE to the planning stage) or shifting more responsibility to the execution side (by moving a PE to the execution stage).

Typically, testing plans are created (Create Testing Plan) for enabling automated testing. They are then coded up (Create Testing Scripts) in the form of testing scripts by the DevOps engineer and repeatedly executed (Execute Testing Scripts). As shown in Fig. 5, there are two possibilities with respect to the creation of the testing scripts. One is to create the testing scripts for every instance of automated testing so that the scripts are customized to the particular feature being tested (Create Testing Scripts PE is part of the Continuous Integration stage), whereas the other is to have a consistent

and standard set of testing scripts that would allow testing coverage irrespective of particular product features being developed (Create Testing Scripts PE is part of the Testing Plan stage). The trade-offs would be between customized behavior and efficiency; on the one hand, the repeated creation of testing scripts would allow specific and customized testing, while in the other case, the development lifecycle automation would be higher. Enterprises would have to choose the appropriate configuration based on their situational and contextual needs.

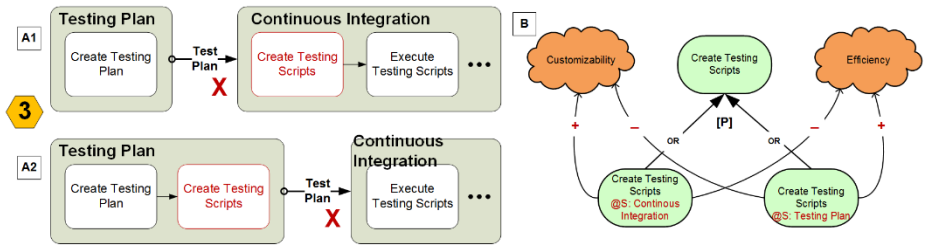

Fig. 5. Test scripts creation placement alternatives (A1) As part of the continuous integration stage with increase customizability of testing, (A2) As part of the testing plan stage leading to greater efficiency and reuse. (B) Analyzing the placement of test script creation along the plan-execute dimension while considering trade-offs for NFRs.

3.4 The Design-Use Dimension

A BP can result in the creation of a tool, capability or artifact that can be repeatedly used. Just like the plan-execute dimension, such BPs can be considered as having two distinct stages, with one stage being responsible for designing the artifact and the other stage for using that artifact repeatedly. Thus, a design and use relationship exists between these segments of the process. In the BPA modeling framework, the stage producing the artifact is called the *design stage* and the stage using the artifact is called the *use stage*. The use stage uses the artifact repeatedly without necessarily being aware of the inner working of that artifact. PEs can also be moved from a design stage to a use stage (and vice versa), with such a repositioning either leading to an increased design/artifact sophistication/automation or to trading the design effort for run-time usage control/customizability.

The DevOps approach emphasizes greater automation of the software development lifecycle through the use of tools. A number of third-party tools are available (e.g., Jenkins[1] for CI, Chef[2] for deployment management, Github[3] for source repository and Splunk[4] for application monitoring etc.), which provide such automation of process activities. These tools are configured (designed) for use in any particular DevOps implementation and thus enable a move from manual methods of product deployment

1 https://jenkins-ci.org/
2 https://www.chef.io/
3 https://github.com/
4 http://www.splunk.com/

(shown by the Manual Deployment stage in Fig. 6) to more automated and CD cycles (shown by the Environment Setup design stage and the Continuous Deployment use stage). However, the introduction of any artifact in the design-use dimension should be evaluated against the NFRs (as shown by the goal model).

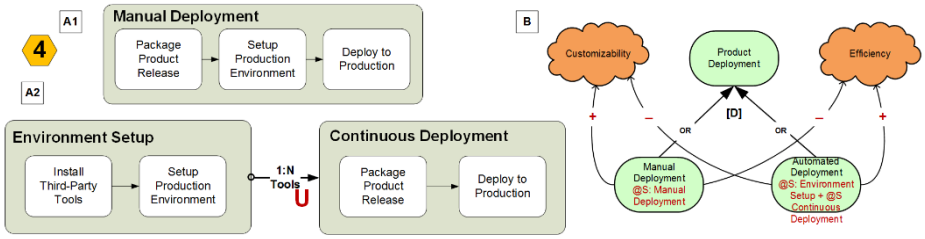

Fig. 6. Deployment of product release alternatives (A1) Manual deployment without the use of automated tools, (A2) Through the design and use of configured third-party tools. (B) Analyzing the need for having a design-use dimension for product release deployment.

4 Related Work

CI and CD are well understood concepts in continuous software engineering where the objective is to deliver ongoing software product improvement and enhancements in less time and greater frequency through improved process automation and introduction of suitable software tools [8]. Generally, any organization would have multitudes of software processes to handle different project development situations; appropriate software processes are selected based on situational needs and business context [18]. Software process tailoring refers to the customization of standard software planning, development and operational processes [19] in situations where organizations need an enterprise-level assessment on how environmental factors, product and project goals, and other organizational aspects influence software process configurations. Commonalities and variabilities exist between these software processes and, as such, these software processes can be tailored to meet specific enterprise business and operational goals and objectives using different techniques for decision making [20,21].

Several software process modeling techniques exist and are primarily based on process modeling languages (such as BPMN), Unified Modeling Language (UML) or Software & Systems Engineering Metamodel (SPEM) [22]. Apart from a few (such as [23,24]), most of the software process modeling frameworks do not provide support for modeling variability in software process configurations and the ability to reason about them while taking enterprise- and process-level NFRs into consideration.

Enterprises attempt to reduce development effort and increase the range of product features offered through software product lines (SPLs). SPLs can be used to support multiple software products through the development of common software architecture(s) and code components. SPLs rely on variation points to support software product variability [25]. Delaying decisions during the development cycles of these SPLs provides the benefit of allowing the optimization of technical and business goals (e.g., increased code reuse) across multiple products, possibly at the expense of other goals

(e.g., simpler architecture). Extending the idea of SPLs to processes results in the notion of Software *Process* Lines (SPrL) [26], which is based on a similar premise: similarities and differences between a set of software processes could be scoped for determining customized software process configurations as per unique software project conditions. In [27], the idea of (software) Process Line Architectures (PLA) is introduced. A PLA is described as "a process structure which reflects the commonality and variability in a collection of processes that make up a process line from the perspective of overall optimization". Like a BPA, a PLA also represents the existence of VPs in (software) processes. However, it does not support the placement of a process element along the four dimensions as described in this paper.

Previous research on BPAs largely focused on the nature of the relationships among their BPs. Various relationship types were proposed (e.g., [28,29]), such as sequence, reference, composition, etc. Unlike most BPA approaches, we focus on systematically analyzing multiple BPA alternatives along the four variability dimensions with the aim at finding the one that best matches the properties of the domain.

Another relevant domain is BP variability modeling that focuses on representing customizable BP models and deriving custom variants from them (see [30] for an overview), with the key element being a VP, which is used to represent and bind variability. Overall, these approaches deliberate about variability only at the process level (within a single process) and do not support reasoning about BPAs. In dealing with BP flexibility, Weber et. al [31] propose four dimensions of change, including the one focusing on the recurrence of activity execution. While somewhat similar to our approach, it neglects trade-offs among the various options and does not cover flexibility in BPAs. Feature models [32] are sometimes employed as a useful abstraction to help guide BP customization (i.e., selecting or deriving a BP variant from a customizable process model). While a viable option, feature models (unlike goal models used here) lack the ability to represent selection criteria and support trade-off analysis among configuration alternatives.

5 Future Work

In future iterations of this work we plan to study the following:

- Many enterprises are becoming critically dependent on software and software processes to create and deliver value to their stakeholders in the form of products and services. Successfully introducing software process reconfigurations in response to changing business models or strategic direction may impact the ongoing delivery of value, product and services [33]. We aim to link the impact of software process reconfigurations to business goals and value in order to exploit synergies and mitigate negative consequences.
- We wish to understand the possible forms of software process reconfigurations with the intention of identifying key points of process variations and the influencing factors that contribute towards these process reconfigurations. Requirements for software process reconfiguration are usually developed in response to shifting enterprise objectives, adaptability requirements and emerging digital technologies

in the enterprise context. The relationship between these requirements and their influence on variation points for software processes would need to be understood.

- Processes are executed by participants or actors in any enterprise. Changes in organizational structure and team dynamics would invariably influence process configurations (and vice versa). For example, any process reconfiguration would possibly shift the boundaries of actor influence with some actors gaining responsibility and other actors losing responsibility or power. Conversely, changing an actor's boundary of influence may also require the selection of an alternate process configuration to successfully attain the same set of goals. The association of operational process level concerns and social organizational considerations needs to be studied and developed by combining the BPA technique (for process representation) with a social actor modeling framework, such as *i** [34].
- Enterprises take advantage of software metrics to routinely and incrementally improve on software processes. While software metrics are well documented [35], illustrating and analyzing the integration and usage of these software metrics for ongoing software process improvements, through the use of enterprise modeling techniques, is not well covered. The BPA can be continuously refined through use of software metrics and data analytics in all stages of the feedback loop – i.e., sensing, interpreting, deciding and acting.

We are exploring methods and techniques from diverse areas, including software engineering, requirements engineering, system dynamics, and management literature, to contribute towards a framework for the management of enterprise software process variability. We are developing a meta-model and an ontology to understand the nature of software process variability and to extend existing enterprise modeling techniques to incorporate attributes and constructs for denoting variability and flexibility in software processes. Finally, we aim to validate such a proposed framework by conducting case studies for various types of enterprises.

6 Conclusions

Every enterprise relies on various BPs for proper functioning, which can take many forms and can include operational, transactional, strategic, recurring, design processes, etc. Having uniform and static processes is no longer an option for enterprises dealing with a multitude of dynamically changing situations that require periodic adjustment of process configurations [36,37]. A recent report from Gartner mentions that "by 2017, 70 percent of successful digital business models will rely on deliberately unstable processes designed to shift as customer needs shift" [38]. In this paper, we considered the possible dimensions of software process reconfigurability using the DevOps approach as a motivating example. Limitations of current process modeling languages, such as BPMN, in illustrating multiple aspects of process architecture were discussed, with the BPA modeling technique being used to describe four dimensions of PE positioning, namely, temporal, recurrence, plan-execute, and design-use, in a typical DevOps implementation. Goal models were used for evaluating alternate software process reconfigurations by assessing the satisfaction of enterprise NFRs.

7 References

1. Wilkinson, M.: Designing an "adaptive" enterprise architecture. BT Technology Journal, 24(4), pp. 81–92 (2006)
2. The Economist: Organisational Agility: How Business can Survive and Thrive in Turbulent Times. A report from The Economist Intelligence Unit (2009)
3. Gartner Research: Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations, March 5, 2015, http://www.gartner.com/newsroom/id/2999017
4. Erich, F., Amrit, C., Daneva, M.: A mapping study on cooperation between information system development and operations. In Product-Focused Software Process Improvement, pp. 277–280. Springer (2014)
5. Bang, S. K., Chung, S., Choh, Y., Dupuis, M.: A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps. In Proceedings of the 2nd annual conference on Research in information technology, pp. 61–62. ACM (2013)
6. Lwakatare, L. E., Kuvaja, P., Oivo, M.: Dimensions of DevOps. In Agile Processes, in Software Engineering, and Extreme Programming, pp. 212–217. Springer (2015)
7. Smeds, J., Nybom, K., Porres, I.: DevOps: A Definition and Perceived Adoption Impediments. In Agile Processes, in Software Engineering, and Extreme Programming, pp. 166 – 177. Springer International Publishing (2015)
8. Bosch, J. (Ed.): Continuous Software Engineering. Springer (2014)
9. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In Business Process Management, pp. 246–261. Springer, Berlin Heidelberg (2007)
10. Business Process Model and Notation, v2.0, http://www.omg.org/spec/BPMN/2.0/PDF/
11. Ståhl, D., Bosch, J.: Modeling continuous integration practice differences in industry software development. Journal of Systems and Software, 87, pp. 48–59 (2014)
12. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Using scrum in distributed agile development: A multiple case study. In Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, pp. 195–204. IEEE (2009)
13. Fitzgerald, B., & Stol, K. J.: Continuous software engineering and beyond: trends and challenges. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, pp. 1–9. ACM (2014)
14. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall, (2002).
15. Haeckel, S.H.: Adaptive Enterprise: Creating and Leading Sense-And-Respond Organizations. Harvard Business Press (1999)
16. Lapouchnian, A., Yu, E., Sturm, A.: Re-designing process architectures towards a framework of design dimensions. In Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on, pp. 205–210. IEEE. Chicago (2015)
17. Lapouchnian, A., Yu, E., Sturm, A.: Towards Variability Design for Business Process Architecture. In 34th International Conference on Conceptual Modeling, (2015). Accepted.
18. Alegría, J. A. H., Bastarrica, M. C.: Building software process lines with CASPER. In Software and System Process (ICSSP), 2012 International Conference on, pp. 170–179. IEEE. (2012)
19. Pedreira, O., Piattini, M., Luaces, M. R., Brisaboa, N. R.: A systematic review of software process tailoring. ACM SIGSOFT Software Engineering Notes, 32(3), pp. 1–6, (2007)
20. Martıınez-Ruiz, T., Garcıa, F., Piattini, M., Munch, J.: Modelling software process variability: an empirical study. Software, IET, 5(2), pp. 172–187 (2011)

21. Martínez-Ruiz, T., García, F., Piattini, M.: Managing process diversity by applying rationale management in variant rich processes. In *Product-Focused Software Process Improvement*, pp. 128–142, Springer Berlin Heidelberg (2011)
22. García-Borgoñon, L., Barcelona, M. A., García-García, J. A., Alba, M., Escalona, M. J.: Software process modeling languages: A systematic literature review. Information and Software Technology, 56(2), pp. 103–116 (2014)
23. Cares, C., Mayol, E., Franch, X., Alvarez, E., Goal-driven agent-oriented software processes, in: Proceedings of the 32nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA, Cavtat/Dubrovnik, Croatia, pp. 336–343 (2006)
24. Washizaki, H.: Deriving project-specific processes from process line architecture with commonality and variability, in: Proceedings of the IEEE International Conference on Industrial Informatics (INDIN'06) , pp. 1301–1306, Singapore (2007)
25. Van Gurp, J., Bosch, J., Svahnberg, M.: On the notion of variability in software product lines. In Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on, pp. 45 –54, IEEE (2001)
26. Rombach, D.: Integrated software process and product lines. In Unifying the Software Process Spectrum, pp. 83–90, Springer Berlin-Heidelberg (2006)
27. Washizaki, H.: Building software process line architectures from bottom up. In Product-Focused Software Process Improvement (pp. 415–421). Springer Berlin-Heidelberg (2006)
28. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management, Ch.2. Springer-Verlag, Berlin-Heidelberg (2013)
29. Eid-Sabbagh, R., Dijkman, R., Weske, M.: Business process architecture: use and correctness. In Proc. 10th International Conference on Business Process Management (BPM'12), pp. 65–81, Springer-Verlag, Berlin- Heidelberg (2012)
30. La Rosa, M., Aalst, W.M.P. van der, Dumas, M., Milani, F.P.: Business process variability modeling: A survey. ACM Computing Surveys (2013)
31. Weber, B., Reichert, M., Rinderle-Ma, S: Change Patterns and Change Support Features – Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Engineering (3), pp. 438–466 (2008)
32. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, SEI, Carnegie Mellon University (1990)
33. Esfahani, H. C., Yu, E., & Annosi, M. C.: Strategically balanced process adoption. In Proceedings of the 2011 International Conference on Software and Systems Process, pp. 169 – 178. ACM (2011)
34. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press (2011)
35. Fenton, N., Bieman, J.: Software metrics: rigorous & practical approach. CRC Press (2014)
36. Yu, E., Deng, S., Sasmal, D.: Enterprise architecture for the adaptive enterprise – A vision paper. In Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation, pp. 146–161, Springer Berlin Heidelberg (2012)
37. Yu, E., Lapouchnian, A.: Architecting the enterprise to leverage a confluence of emerging technologies. In Proceedings of the 2013 CASCON, IBM Corp. (2013)
38. Spender, A.: Top 10 Strategic Technology Predictions for 2015 and Beyond, Gartner Research. February 18, 2015, http://www.gartner.com/smarterwithgartner/top-10-strategic-technology-predictions-for-2015-and-beyond/