Modeling Software Process Configurations for Enterprise Adaptability

Zia Babar

Faculty of Information, University of Toronto
zia.babar@mail.utoronto.ca

Abstract. Modern enterprises are expected to continuously evolve and adapt to uncertain environmental conditions and evolving customer trends. Adaptability in software processes enable enterprises to respond to changing situations by selecting software process configurations that help best meet enterprise-level business goals. Conventional methods of modeling and designing software processes are limited in their ability to visualize these software process configurations, reason about them and select an appropriate configuration which meet functional and non-functional requirements while considering enterprise-level perspectives. As part of our PhD project, we propose a requirements-based software process adaptability framework that considers software process adaptability, first at a process-centric and then at an agent-centric level. Key constructs for this framework are discussed and illustrated by using the DevOps approach as an example.

Keywords: Enterprise Modeling, Software Processes, Software Process Variability, Agent and Goal Modeling, Adaptive Enterprises.

1 Introduction

1.1 Background

Modern enterprises are expected to continuously evolve and adapt to uncertain environmental conditions and increased competition from new market players, including those from non-traditional sectors [1]. Customers are increasingly expecting personalized services while emerging technologies are causing the digital transformation of enterprises. To this end, more and more enterprises are relying on software to aid in the delivery of customer-centric products and services in progressively more turbulent and dynamic environments [2]. As a result, software processes (SPs) are becoming an integral part of enterprises' strategic and operational processes. Recent years have seen the emergence of various software development approaches and practices. The current rapid adoption of DevOps [3] creates an opportunity to re-examine the ability of enterprises to quickly deploy new software product features, through to the end-user, by having frequent product release cycles. DevOps enables the achievement of enterprise business objectives through (a) the automation of process tasks in the software development lifecycle, (b) solicitation of customer feedback and usage of software metrics to continuously refine and improve the software development process, and (c) reducing department silos by promoting a culture of collaboration and sharing of information across teams [2][4]. The above are attained through diverse considerations, ranging from systems design and software tool support, business and process configuration, to social and organizational behavior matters.

1.2 Problem Statement

Each enterprise has unique characteristics and business goals; software development processes can vary significantly between enterprises as these processes are configured considering the unique variations and nature of software products and projects, and the behavioral peculiarities of each organization [5]. An appropriate configuration of SP needs to be selected based on defined enterprise-level functional requirements (FRs) and non-functional requirements (NFRs). The SP needs to be adaptable so as to be able to handle evolving enterprise situational needs, particularly those that result from emerging digital technologies (such as social media, mobile technologies, big data analytics, and cloud computing) within an enterprise setting. NFRs for SPs include adaptability of the development process, speed of adaptation, shortening the deployment cycle etc.

Adaptability requires the consideration of social- and enterprise-level perspectives, while addressing practices such as continuous software engineering, using concepts of multi-level adaptive systems, and linking software process design considerations with organizational stakeholder interests and enterprise-level business goals. While there is significant literature which studies the variations and commonalities between SP families, these studies do not sufficiently cover the high-level abstractions for representing adaptation constructs of SPs and mostly deliberate at a software process adoption and implementation level. Furthermore, conventional methods of modeling SP reconfigurations are limited in their ability to consider the multiple enterprise-level viewpoints for each alternative SP configuration and have also not been applied to the range of considerations that are present in DevOps.

1.3 Research Objectives

The objective of this research can be succinctly described as "to define and develop a requirements-based SP adaptability framework which enables enterprises to ensure ongoing and sustained delivery of products and services under varying circumstances while adhering to enterprise-level FRs and NFRs." An enterprise has fundamental SP adaptation tendencies, particularly those pertaining to emerging digital technologies; the nature and nuances of these need to be understood, determined and categorized, as well as the link established between SP reconfigurations and enterprise business goals. Based on this understanding, SP adaptability realization techniques need to be developed and abstracted for representing SP adaptability with respect to relevant constructs, concepts, relationships, information flows, dimensions etc.

The determined abstraction will be visually depicted through modeling notations. Additionally, these aspects need to be represented as a meta-model formalization. Existing modeling languages from the areas of system dynamics, systems and component modeling, business and software process modeling, agent and goal modeling etc. will be considered and extended as required. SP adaptability requirements need to be established and verified for requirements satisfaction. The SP adaptability framework aims to provide a way of characterizing the as-is and to-be states as alternate SP configurations. An existing as-is state may be deemed to not be satisfying adaptability requirements and thus necessitate the selection of an alternate to-be state. This research will develop software tool support for different parts of the framework such as the visualization and drawing of adaptability requirements, analysis of adaptability models and simulation of model evaluation.

2 Related Work

Tactics exist for the tailoring [6] and improvement [7] of standard SPs to meet the specific needs of an enterprise; this is accomplished through some form of adaptation of certain activities or SP parameters based on an assessment of environmental factors, product and project goals, and other organizational aspects. The Capability Maturity Model Integration (CMMI) models provide guidance for process development and maturity for different organizational areas, including software processes [8]. Software Product Lines (SPL) can reduce development cost for product families through the determination and reliance on variation points [9]; delaying the placement of these variation points along the software development cycle can provide certain technical and business benefits (e.g., increased code reuse) across multiple products at the expense of other goals (e.g. simpler architecture). A family of software processes can have task commonalities and variabilities; these could be integrated to produce Software *Process* Lines (SPrL) [10] which would help reduce the effort of managing many individual SPs that exist in any enterprise.

Situational Method Engineering (SME) can be used to create development methods for specific purposes by selecting and combining method fragments previously stored in method repositories [11]. Social actor modeling frameworks (such as *i** [12]) focus on the social dimension of any domain by allowing the incorporation of actor intentionality, motives and goals during domain analysis. This allows for the evaluation of alternatives based on the satisfaction of actor goals which can assist in the selection of software process configurations based on enterprise NFRs. At an enterprise level, Business Product Architectures (BPAs) have been previously discussed in [13], including the nature of relationships and the dependency types that exist between business processes, and the "binding" of variation points on alternative selection.

The above is not an exhaustive listing of related work but serves to demonstrate the range of study that can be considered related to the general "software process adaptability" area. This research aims to deal with software process adaptability at an enterprise level by considering system-, process-, and social-level factors while advancing current methods for software process modeling and design.

3 Initial Results and Expected Outcomes

Key constructs from the general area of SPs need to be abstracted to express the adaptability nature of SPs. The constructs for SP adaptability will be depicted through a modeling notation with multiple enterprise modeling and architecture perspectives being leveraged for considering these constructs. DevOps is used as an illustration for SP adaptability as the multiple enterprise-level perspectives (such as systems, process, social, and organizational) in the DevOps approach provide an interesting challenge for enterprise modeling. Further, the DevOps approach permits the study of SP adaptability at a process-centric and at an agent-centric level. Supplementing process-centric models with agent-centric models allows for the inclusion of stakeholder interests during reasoning and analysis resulting in more suitable adaptability designs.

3.1 Process-Centric Framework

The process-centric framework builds on the work previously introduced in [14] where the dimensions of Business Process Architecture (BPA) were discussed. *Process elements* (PEs) can be thought of as being a unit of any SP with a PE producing a measurable output based on a set of control and data inputs. Placing of PEs can be done along multiple BPA *dimensions*; the dimensions being temporal, recurrence, plan-execute and design-use. SP configurations (e.g., different variations of DevOps) can be represented through the placement of PEs along an SP. Similar placements across two configurations are referred to as commonalities while differences between configurations are considered as variabilities with "*choice-points*" indicating alternative options of PE placements. The placement of a PE along any one of these dimensions, including their movement along these dimensions, is done after considering suitable enterprise-level NFR trade-offs. *Boundary* conditions exist in any DevOps configuration, with PEs being placed within a boundary based on their relative characteristics. These concepts also allow us to handle different software engineering concepts, such as technical debt [15]. Movement of PEs within a boundary may not result in increased technical debt however moving a PE beyond the boundary may either increase or decrease technical debt significantly. Thus, alternative DevOps reconfigurations are obtained by considering technical debt (or other appropriate) trade-offs of PE placements across boundaries.

As with any modeling technique, adaptability constructs are connected to each other through *relationships* for representing sequencing, dependencies, information transfer, compositionality, triggers etc. Existing notions of relationships and interactions from multiple enterprise modeling techniques will be considered with the objective of extending them by considering the influence of emerging digital technologies and enterprise adaptability demands. For example, DevOps could undergo ongoing adaptation and refinement through the monitoring of software metrics collected through big data analytics which are propagated through feedback and feedforward loops. Adaptation from one DevOps configuration to another can be initiated through *triggers*. Triggers can be "fired" through data-driven sensing mechanisms and re-directed through to PEs as inputs. These triggers would then allow the PE to interpret this new sensory input and act with respect to the selection of suitable alternatives that satisfy the enterprise NFRs based on the changed situation. Fig. 1 shows a BPA DevOps implementation model with some of the aforementioned constructs.

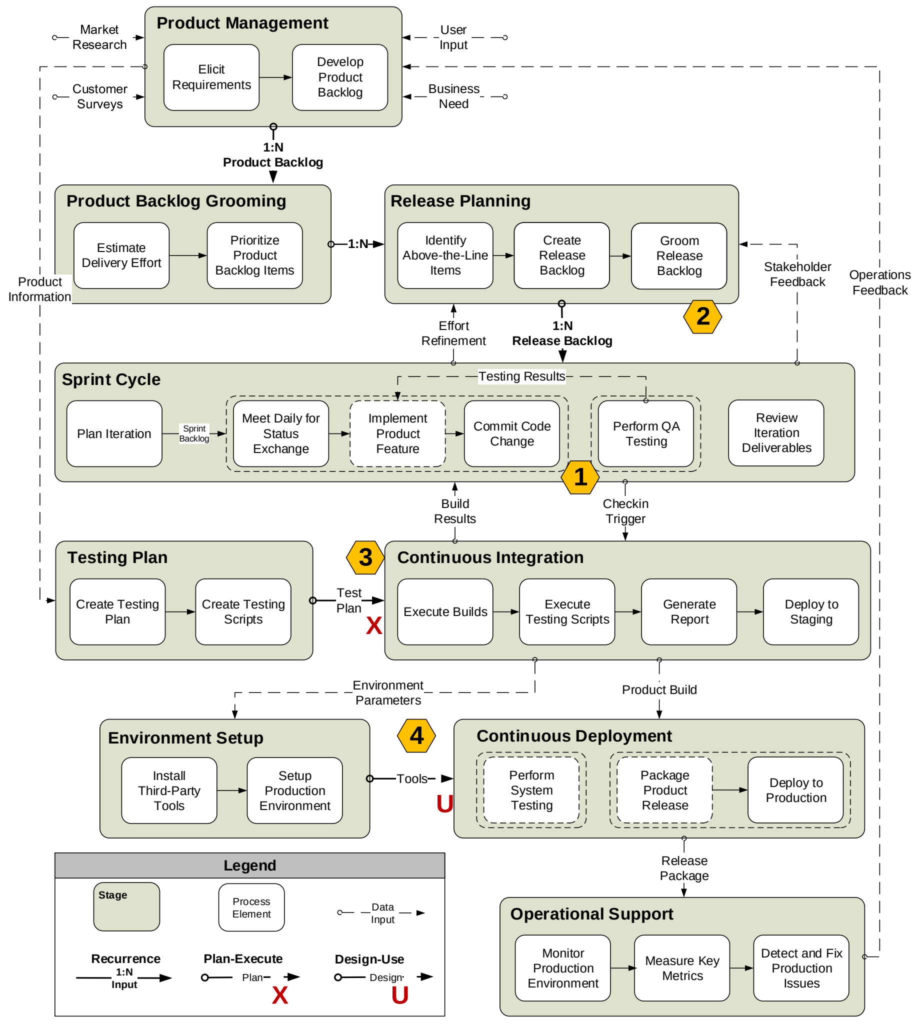

Fig. 1. Business Process Architecture (BPA) for a DevOps approach (Source: [16])

Execution of some PEs result in the production of *artifacts*. Artifacts can adopt different forms; they may be physical objects, a digitized entity or even be informational in nature. A PE along the plan-execute dimension can produce a planning artifact or an executable artifact. Similarly, a PE along the design-use dimension can produce a design artifact or an artifact that is used for some purpose. Examples of artifacts in the DevOps approach include testing plans, testing scripts, testing tools, environment setup configurations, product planning backlogs, software product features etc. Some of these artifacts (such as testing plans or testing tools) are internal to the SP process and are intended to be used for the production of the final delivered artifact (such as product features or product releases). In other cases, the progression along the SP may result in artifacts to manifest into another form. For example, a testing case design artifact can manifest itself as a testing case implementation (testing script) artifact. Fig 2. shows a goal-oriented approach for evaluation between two different process re-configurations through the use of enterprise NFRs (represented by softgoals) [17].

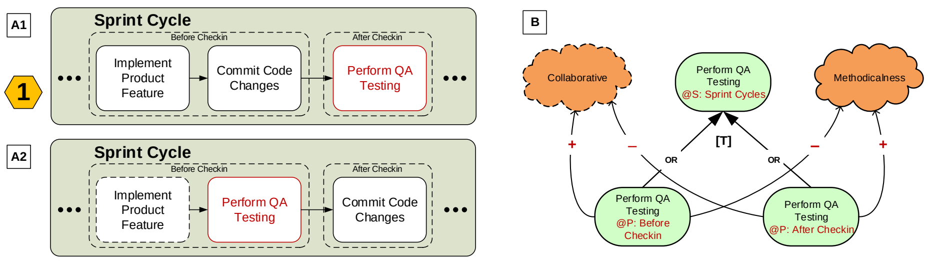

Fig. 2. QA testing alternatives (A1) As a separate phase from product feature implementation, (A2) As part of the product feature implementation phase. (B) Analyzing the temporal placement of QA testing process element based on NFRs. (Source: [16])

3.2 Agent-Centric Framework

The agent-centric framework extends the process-centric framework by allowing the inclusion of social and agent relationships. SPs exist in a complex and collaborative environment which is influenced by a multitude of human and non-human (system) actors. These actors are intentional in nature and can be considered to be executors and influencers of the PEs. Considering actors as having intentionality permits the use of agent-oriented modeling notions (such as [12]), and allows the assignment of goals and soft-goals to these actors. Actor-assigned goals provide motivation for choosing one particular DevOps configuration over another; with the alternative configuration selection being done through the use of goal satisfaction evaluation methods. For example, a particular set of PE placements (and, by extension, a DevOps configuration) would be preferred in a situation where rapid deployment is an actor objective as opposed to a situation where a more structured release based deployment is required.

The agent-centric framework will have similar constructs to that of the process-centric framework. For example, agents too have boundaries of influence however these may or may not align with process-centric boundaries. Agent could have their tasks or goals move within or beyond their boundaries of influence; methods needs to be devised that allow the determination of these boundaries and the common attributes of elements that reside within an actor boundary. Similar to the boundary construct, the relationships and artifacts constructs exist in the agent-centric framework as well, albeit with some conceptual differences from those in the process-centric framework. At present we are studying how both these modeling frameworks can be aligned with each other.

4 Methodology

Design science research has been gaining wide acceptance in Information Systems. The guidelines-based approach introduced in [18] will be used in this PhD project for the development and validation of design artifacts.

- **Design as an artifact:** A *modeling language* (with the model being a set of *constructs*) is being developed for representing SP adaptability along with *methods* that allow for alternative selection based on enterprise goals. These design artifacts will be developed through a study of published case studies sourced from various academic papers. Software tools will provide the design *instantiation*.
- **Problem relevance:** Enterprises (large and small) are forced to continuously adapt and reconfigure their software processes in order to deliver products and services through short-cycle software releases so as to keep pace with evolving customer expectations. A conceptual framework is required that supports such enterprise requirements while considering system-, process, and social-level factors.
- **Design evaluation:** Analytical evaluation techniques will be used for ongoing refinement of the artifact(s) during its development process. Industry partners will be approached to understand their situational needs and constraints and the proposed design artifact will be tested and refined against these real-world situations.
- **Research contributions:** Current techniques for software modeling and design would be advanced by addressing practices such as continuous software engineering and linking software process design and adaptation considerations with organizational stakeholder interests and enterprise-level business goals.
- **Research rigor:** The need and acceptance of variability at a software product, software process and enterprise level is well understood and accepted. The proposed design artifacts extend these foundational areas and will be validated through theoretical and empirical evaluation methods.
- **Design as a search process:** The design search process will start from process-based considerations and be gradually expanded to include enterprise-level concerns. The effects of and on SP configurations by social agents and software product design would be considered as the design artifacts are developed.
- **Research communication:** The research is primarily targeted towards an enterprise modeling and information systems engineering audience and venues of communication will be chosen accordingly. The results will be published in scholarly literature on an on-going basis at the completion of each research stage.

5 Conclusions and Thesis Progress

In this paper we introduced the problem of software process adaptability and proposed a requirements-based approach for evaluating and analyzing this area. The constructs for SP adaptability are presently being understood to define their behavior and characteristics. We aim to form an initial description and definition of these constructs in the third year of our PhD studies, along with their representation as a meta-model and modeling notation. In our fourth year of studies, we aim to seek out industry partners for ongoing refinement and validation of the proposed solution.

Acknowledgements
I would like to recognize my PhD supervisor, Prof. Eric Yu, for his ongoing support as I progress through the doctoral program and establish a research career for myself.

6 References

1. Wilkinson, M.: Designing an “adaptive” enterprise architecture. BT Technology Journal, 24(4), pp. 81–92. (2006)
2. 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)
3. 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)
4. Lwakatare, L. E., Kuvaja, P., Oivo, M.: Dimensions of DevOps. In Agile Processes, in Software Engineering, and Extreme Programming, pp. 212–217. Springer (2015)
5. Bosch, J. (Ed.): Continuous Software Engineering. Springer (2014)
6. 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)
7. Zahran, S.: Software process improvement: practical guidelines for business success. Addison-Wesley Longman Ltd. (1998)
8. Chrissis, M. B., Konrad, M., Shrum, S.: CMMI Guidelines for Process Integration and Product Improvement. Addison-Wesley Longman Publishing Co., Inc. (2003)
9. 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)
10. Rombach, D.: Integrated software process and product lines. In Unifying the Software Process Spectrum, pp. 83–90, Springer Berlin-Heidelberg (2006)
11. Henderson-Sellers, B., Ralyté, J.: Situational Method Engineering: State-of-the-Art Review. Journal for Universal Computer Science, 16(3), pp. 424–478. (2010)
12. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press (2011)
13. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.,: Fundamentals of Business Process Management, Ch.2. Springer-Verlag, Berlin-Heidelberg (2013)
14. 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)
15. Kruchten, P., Nord, R. L., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Software, 29(6), pp. 18–21. (2012)
16. Babar, Z., Lapouchnian, A., Yu, E.: Modeling DevOps Deployment Choices using Process Architecture Design Dimensions. In PoEM. Springer. (2015). Accepted.
17. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-driven design and configuration management of business processes. In BPM, pp. 246–261. Springer (2007)
18. Hevner, A.R., March, S.T., Park, J., and Ram, S.: Design Science in Information Systems Research, MIS Quarterly, 28(1), pp. 75–105. (2004)