A Modeling Framework for Analyzing Process Architecture Transformations in the Software-Enabled Enterprise

by

Zia Babar

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Faculty of Information
University of Toronto

© Copyright by Zia Babar 2020

A Modeling Framework for Analyzing Process Architecture Transformations in the Software-Enabled Enterprise

Zia Babar

Doctor of Philosophy

Faculty of Information
University of Toronto

2020

Abstract

Software is becoming a fundamental enabler of all kinds of evolving enterprise capabilities and opportunities. For the enterprise to take advantage of software-based technologies, there will be redesigns of processes that are responsible for the development of software, and the business processes where these software are used. There exists extensive work on business process modeling and analysis however, it is not adequate to study and optimize processes in isolation as enterprise objectives are attained by multiple interrelated processes as a collection. We refer to a collection of processes and their interrelationships as a process architecture.

By repositioning decision points within a process architecture, an enterprise can take better advantage of software flexibility and data-driven capabilities. In this thesis research, we model and analyze process architecture reconfigurations and study possible alternative configurations of these process architectures, with emphasis on adaptability and flexibility. Through reconfigurable and flexible process architectures, enterprises can respond to changing situations by selecting suitable process architecture alternatives that best meets enterprise business objectives.

We introduce the hiBPM framework to support the study of process architecture reconfigurations. A hiBPM model describes processes, and their relationships and interactions needed to accomplish enterprise objectives. hiBPM emphasizes the existence of various decision-making points and offers expressiveness to allow relevant architectural properties to be analyzed, and to contrast amongst alternative process architecture configuration options. Through hiBPM, enterprise architects can explore alternatives to come up with process architecture that enable enterprise flexibility while factoring in other non-functional criteria, such as the time or cost involved.

This thesis research used design science research to determine hiBPM framework design artifacts, and case study research for the evaluation and validation of the developed conceptual modeling framework. The case studies performed contributed to the validation of the hiBPM framework by highlighting the varying degrees of plan and design completeness suitable to different contexts and situations within the enterprise under uncertain conditions, elaborating on the different types of processes in the process architecture and their relationships, and determining additional processes that were necessary for meeting of enterprise functional and non-functional requirements.

Acknowledgments

The attainment of my doctoral degree could not have been possible without the support and guidance of many individuals.

I would like to recognize Professor Eric Yu for enabling me to get to this stage. I originally reached out to him almost eight years ago asking if I could study under his tutelage. That one email started a new chapter in my life and allowed me to pursue certain life-long aspirations of mine. Prof. Yu spent innumerable hours reviewing my work and providing feedback on my research writings. I consider myself fortunate to have been his student and I look forward to exploring future research avenues with him.

I am further grateful to have Professor Kelly Lyons and Professor John Mylopoulos on my PhD committee. I was able to discuss my work with them on a regular basis and my research benefitted from their experience and insights. I also appreciate Professor Chun Wei Choo and Professor Alex Borgida for their detailed review, feedback and comments on my thesis.

I have been fortunate to be part of a very strong and supportive cohort of researchers, belonging to both the Faculty of Information and the Department of Computer Science. I wish to thank each and every one of them for their assistance and comradery. Attaining a PhD is often a lonely journey, and it's nice to have a supportive community with whom one can share one's joys and sorrows. I particularly wish to thank Christie Oh who lifted my spirits up at key moments, and Vik Pant, with whom I have had numerous cerebral discussions that allowed me to grow further as an individual, as a professional and as a researcher.

I wish to thank my parents who have made numerous sacrifices to ensure that their only son could pursue a path of his choosing. I also wish to thank my in-laws for the support that they have provided over the many years. My two sons, Zain and Zair, have been exceedingly patient having seen me work on my PhD for as long as they remember, and it is now time to spend some quality time with them. Lastly, I am thankful to my wife, Fauzia, who has always believed in me and has always been willing to undergo many personal sacrifices to support me.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Background

Software is a fundamental enabler of all kinds of evolving enterprise capabilities and opportunities [1] and is expected to play a more significant role across multiple industries and enterprises. The use of emerging and disruptive software technologies results in fundamental changes in these "software-enabled" enterprises, such as how the enterprise engages with its customers or the changes that occur to the business model [2][3]. By software-enabled enterprises, we refer to enterprises that rely on software-based technologies (such as Mobile, Cloud Computing, IoT, Artificial Intelligence, Blockchain etc.) to help attain specific enterprise objectives, such as that of agility and customer-centricity. Through adopting software in new ways, enterprises can have more rapid cycles of tools and artifacts development, reducing the time-to-market for new products and features, providing greater customer-centricity, rapidly introducing new features based on evolving customer needs, quickly resolving operational and support issues, and showing improved responsiveness to changing environmental situations.

For the enterprise to take advantage of software-based technologies, there will be redesigns of processes that are responsible for the evolution and adoption of software, and the business processes where these software are used. Such a change in the enterprise can lead to different possibilities, such as the inventions of new software technologies and innovation in how the software is developed or how it is used. However, in enterprises undergoing continuous change, business process design activities cannot be done once and be assumed to be valid over significant periods but instead need to be periodically reviewed and reconsidered. Parts of the enterprise engaged in developing and deploying software also go through many transformations to meet differing requirements for the speed of delivery of software. Specific drivers of change need to be monitored and evaluated, with alternative designs of business and software processes selected for ongoing implementation and execution.

Conversely, evolving requirements needed for the enterprise to adapt and respond to changing circumstances have resulted in software technological innovations as well. Software systems are becoming increasingly complex and thus difficult to develop, deliver and support a range of

enterprise functional and non-functional objectives [4]. Here the term software is meant to include both the software applications used by organizations to attain various enterprise functions and the specialized processes responsible for the development and ongoing evolution of its software. We refer to these specialized processes as software processes. The method of software development and delivery needs to be studied and improved upon, not only for the effective and efficient development and delivery of software systems, products and services but also to ensure that these processes can enable the requisite enterprise transformations.

There exists extensive work on business process modeling and analysis that helps with analyzing the basic functioning of any enterprise but these business processes are generally studied and modelled in isolation without considering their inter-process relationships and structures. It is not sufficient to study and optimize individual and isolated business processes for enterprise change as many processes collectively contribute to the attainment of enterprise objectives. Such a collection of interrelated processes is known as a process architecture and has previously been studied in scholarly literature [50][51]. However, existing approaches to modeling process architecture neither consider the regular and ongoing changes that help an enterprise continue to meet its objectives nor do they factor in uncertainty of design. There is a need for an approach that enables enterprise architects and process architects to model and analyze the ongoing reconfigurations that can happen in the process architecture that exists in the enterprise while considering complexities of business and software process design requirements.

1.2 Problem Statement

Adaptable business processes and software systems are becoming essential for modern enterprises that need to undergo change, with change cycles are happening with increasing frequency. There must be a focus on understanding adaptability and flexibility in business process design in addition to studying the design of enterprises for business process execution [5][6], and the supporting software, that enables enterprise change. Enterprise architects and system designers face choices as to when and where to deploy what kinds of flexibilities in the enterprise when designing business processes. Having reconfigurable and flexible business processes enables enterprises to

respond to changing situations by selecting suitable process design alternatives that help best meet enterprise business objectives.

Many processes collectively contribute to the attainment of enterprise objectives. Different types of process within the process architecture may take place over different timescales and have different frequencies of occurrence, with some activities executing very infrequently (e.g. strategic planning activities) whereas other activities may be more frequently executed (e.g. operational activities). Enterprises can decide to have processes that produce detailed plans which help deal with unpredictability at runtime process execution or may decide to invest in better-designed software tools that are used in the automation of processes. There is often insufficient information to create fully design these processes and often it is better to delay the designing of processes or planning for their execution until additional information is available. There would be relationships between these processes, with the output of certain processes feeding into other processes. Further, the relationships between these processes may themselves change to support enterprise transformations.

It is no longer enough to study and optimize individual and isolated business processes to accommodate the need for enterprise change. Any design reconfiguration of the process architecture should consider all processes that contribute to an enterprise objective. This provides additional possibilities for redesigning that goes beyond what the analysis of a single type of process could offer, and permits ignoring the traditional boundaries that demarcate the different types of processes (such as business, software , strategic, operational or otherwise) that typically exist within the enterprise. Such a design of the process architecture cannot be done in an ad hoc manner as the full spectrum of alternative design configurations would neither be evident nor considered. Instead, there needs to be a structured and systematic method that supports reasoning and insight, enables process architecture transformations and improvements, and promotes data-based design decision making while ensuring alignment with enterprise business objectives.

Contemporary enterprise architecture and process architecture modeling techniques generally assume a reasonably settled and stable set of requirements. They do not cater to periodic, variable and ongoing change, including the ability to decide between multiple alternative enterprise

configurations [7]. These techniques are thus, inadequate to deal with the type of enterprise transformation exercises described above. While such a focus on enterprise transformation can be studied from multiple perspectives, this PhD research project determines the design implications for a multitude of business and software processes with regards to enterprise goals for transformation and change. Such changes in enterprises can be considered at an elementary level, by allowing focus on shifts between process architectural configurations.

1.3 Research Objectives

The objectives of this PhD research project can be expanded into three distinct areas as follows;

1. To identify a set of characteristics of enterprises undergoing change due to software technology innovations and determine a set of requirements for a conceptual modeling framework that can model and analyze the reconfigurations of enterprise process architectures.
2. To use these requirements to design a conceptual modeling framework that identifies the upstream factors (i.e., the "whys") that should be considered in the design of business and software processes that can be traced to enterprise business objectives.
3. To consider the downstream effect (i.e., the "hows") on software systems design and software usage during the design of business processes, including acknowledging the interplays between software design and the design of business processes that use these software.

1.4 Research Approach

Several research methods were used as part of this PhD research project. These are discussed in this section.

1.4.1 Systematic Literature Review

A systematic and structured literature review was performed to identify relevant research papers that would aid in the understanding and determining of the primary characteristics in software-enabled enterprise transformation.

While there are many ways to conduct a systematic literature review, such as [8][9][10], we adopt the eight-step approach proposed in [11] that deals specifically with performing standalone

systematic literature review as applied to information systems research. These eight steps are briefly described below.

- **Step 1:** The intended goal of the review is identified which provides the necessary background, a clear explanation, and the justification for conducting a systematic review.
- **Step 2:** A protocol for conducting the review is developed that is to be followed by the reviewer. In case of more than one reviewer, all reviewers would be trained on the protocol.
- **Step 3:** A criterion for searching the literature is defined, including justifying its the search criteria's comprehensiveness.
- **Step 4:** A practical screen is done to eliminate articles (a) if they do not apply to the purpose of the review, or (b) to limit the number of selected articles for manageability reasons.
- **Step 5:** A criterion is defined with only those articles being selected for the next review step if they are deemed to of sufficient quality, with all other articles being discarded.
- **Step 6:** The articles in the final selected list are then reviewed and studied in greater detail to extract data for the next step in the synthesis of the findings.
- **Step 7:** The extracted data across all the articles is then carefully generalized and combined to develop a set of hypotheses. This is the cumulative outcome of the review process.
- **Step 8:** The entire process needs to be recorded so that it can be independently repeated by other scholars who wish to perform a similar review processes of their own.

1.4.2 Design Science Research

Design Science Research (DSR) is a method well-suited for information systems research due to the inclusion of social and organizational aspects as part of the research methodology [12]. Design science is a research paradigm that attempts to create new design artifacts in order to determine, understand, define and analyze a problem domain in an organization while solving identified issues within [12]. Design science research is more agreeable towards the development of technology-based or technology-derived artifacts, such as conceptual modeling frameworks and software tools, along with their empirical evaluation in an actual study environment.

Design science research is characterized by a sequence of activities for theorizing, justifying, building and evaluating these design artifacts [14]. It has its "roots in engineering and the sciences of the artificial" [12] and is intended to extend human and organizational knowledge by solving organizational problems through the creation of new and innovative artifacts [12]. From an epistemological standpoint, design science research, due to its requirement for producing technical artifacts, can be considered to be generally positivist [12][15] with some anti-positivist inclinations when it comes to evaluation of artifacts. Positivism attempts to "explain and predict what happens in the social world by searching for regularities, causal relationships between its constituent elements" [16]. While design science research engagements may be iterative to ensure research relevance and ongoing artifact refining, it is not explicitly mandated. The resultant artifacts are expected to undergo a final evaluation to ensure research rigour [13].

The guidelines-based design science research approach proposed in [12] was used in this PhD project for the development and validation of design artifacts pertaining to the proposed conceptual modeling framework, as the desired research outcome stated previously strongly correlates to the design artifacts introduced in design science research. These guidelines are as follows,

- **Design as an artifact:** The purpose of design science research is to produce a "viable artifact in the form of a construct, a model, or an instantiation", with artifacts being "innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished" [12]. In our research project, the artifact is a conceptual modeling framework that comprises of a set of modeling constructs and accompanying methods that allow for the prescribing and determining of the usage of the constructs.
- **Problem relevance:** Through design science research, the developed artifacts should be relevant to solving the problem, by helping users of the artifacts move from a "goal state" from the "current state" of the system [12]. This objective is attained through the acquisition of knowledge that would help develop an understanding for developing technology-based solutions to solve a business problem. In our case, we provide a conceptual modeling technique that tackles the problem that was introduced earlier in this chapter.

- **Design evaluation:** The design artifacts go through a process of research evaluation to demonstrate rigour and relevance [17]. This evaluation is done by verifying the utility and quality of the artifact by applying it in a business environment that defines the requirements and conditions of the evaluation process. As indicated in [12], many different forms of design evaluation methods exist, each with their applicability and strengths. We selected the case study research method for the verification of our conceptual modeling framework.
- **Research contributions:** The outcome of the design science research should have a tangible contribution to one or more disciplines. The contribution itself may take the form of either the developed design artifact, the foundations of the design itself, or accompanying methodologies in which the design is developed or used. In our case, the primary contribution of our research is the design artifact itself. The contributions of this research are discussed in greater detail elsewhere in this chapter, and the concluding chapter of this thesis.
- **Research rigour:** Ensuring research rigour is an essential part of design science research and involves a "selection of appropriate techniques to develop or construct a theory or artifact and the selection of appropriate means to justify the theory or evaluate the artifact" [12]. As part of this research project, research activities for ensuring rigour were established and practiced during the design evaluation; these are discussed in greater detail in subsequent sections of this chapter.
- **Design as a search process:** Design science research follows an iterative mechanism of progressively attaining the goal through starting with a simplified explanation or decomposing the problem into manageable areas. We adopted this approach during the designing and developing of the artifacts by following an iterative and incremental approach, starting with simplistic domain examples and slowing building the modeling framework, including the various constructs. These were then verified under a range of conditions, with the framework becoming more practical and applicable.
- **Research communication:** The artifact needs to be presented to an audience of researchers and practitioners from both a technology-oriented and a management-oriented background. This ensures that the target audience includes individuals who can see how the artifact is constructed, and how it is to be utilized within a business context. Considering this, our research is primarily

targeted towards the enterprise modeling, business and software process modeling, and information systems design audience. Venues of research communication and publication were chosen accordingly with the findings published in scholarly literature after each research stage.

1.4.3 Case Study

The design science research approach provides an instructional set of guidelines on how to proceed with this research project. We supplemented this overarching approach with case studies for the evaluation, validation and ongoing refinement of the developed conceptual modeling framework [18][19]. The case study research method can be used from both a positivist and an interpretivist philosophical theory [20]. Specifically, we adopted a positivist case study research method as this is the dominant paradigm in information systems case research and was relevant in our situation for validating our early hypothesis [15]. For case study research performed with a positivist approach, there has to be a systematic process of defining the research criteria, collecting data, analyzing results, while ensuring replicability and generalizability. Theoretical constructs are defined and hypothesized, which are then empirically evaluated and measured [20][21]. Repetition of research activities across multiple case studies allows for the generalizability of the case study research findings [20][21][22].

This research went through cycles of explanation building and ongoing refinement [23] where initial theoretical conceptualizations were used as a guide in the development of the design artifacts and its accompanying methods. These were then applied to several real-world case studies. This allowed for a more detailed assessment and validation than those offered just by published case studies. Several industry partners were approached to understand their situational needs and constraints for applying the preliminary developed framework to these real-world situations and evaluated against these enterprise settings. Through multiple case studies, we were able to both generalize findings and observations across a range of circumstances while refuting theories and ideas which may only be valid in limited circumstances and localized settings [24]. Multiple case studies would also ensure greater validity, generalizability, applicability and rigour of the research exercise while limiting researcher bias and interpretability of data collection and analysis. Iterative refinements were made to the framework as applicable.

These organizations were selected based on relevance and appropriateness to the research study. The cases were in both technology-based organizations (i.e., organizations that specialize in the development of software products and services as their primary business) and non-technology based organizations (i.e., organizations which use digital technologies and software as a means to offer products and services to their customers). The scope and unit of analysis for the case study differed from instance to instance, nevertheless the case study was part of a research project that had business value to the enterprise.

Table 1: Attributes used to assess information systems positivist case studies (Source: [25])

	Attribute
Area 1 - Research Design	Clear research questions
	A priori specification of constructs
	Multiple-case design
	Context of the case study
	Different roles for multiple investigators
Area 2: Data Collection	Elucidation of the data collection process
	Multiple data collection methods
	Data triangulation
	Case study protocol
	Case study database
Area 3: Data Analysis	Elucidation of the data analysis process
	Field notes
	Logical chain of evidence
	Explanation building
	Project reviews

To ensure research rigour in both case studies, we followed the recommendations provided in [25] for positivist case study research. Here, a set of attributes is provided that span three main areas that every positivist case study should have present in order to ensure the rigour of research. We selected this approach as our case studies were often of long duration. Having such a structure helped ensure that specific attributes remained in focus throughout the case study. Further, there were often overlap during the case studies and the findings from one case study were corrected,

extended and refined in the other case study. We selected a subset of attributes from [25] as they pertained to our research; these are listed in Table 1.

The three main areas, including the attributes that were relevant to our case studies, are discussed further below. The specific activities within and across the areas are presented in a sequential manner; however, they were performed iteratively as per the guidelines provided in design science research.

- **Area 1 – Research Design:** Attributes associated with the designing of the case study are covered in the first area. Here, there is attention to developing the research questions, the foundations of the study, and the criteria for selecting the case studies. In our case studies, we emphasized the defining of a clear research problem. We started the case study with a set of constructs that were then tested in each of the case studies. We were going with multiple-case studies to ensure the generalizability and applicability of our design artifact. We also identified the context of the study early on in each of the case studies to ensure that there was an understood frame of reference, in which the study was grounded. Finally, some of our case studies had multiple research investigators, so clear differentiation of responsibilities and research areas was specified.
- **Area 2 – Data Collection:** The second area is concerned with the overall quality and process of data collection in the case study. Here the attributes pertained to the methods used for data collection, particularly their application for enhancing the reliability of data collection. For this, we followed several attributes that apply in this area. We ensured that there was an elucidation of the data collection process through the process of gathering documentation and artifacts and having a questionnaire. As multiple researchers were involved in some cases, and the case studies were performed over a long duration, we needed to ensure reliability and minimize bias. For this, we developed a case study protocol (as required) and a case study database where raw materials (including the iterative conceptual models) were stored.
- **Area 3 – Data Analysis:** Attributes in the third area pertain to the analysis of the data collected in previous research activities, including the application and use of suitable techniques that lend to that analysis. Again, in our case, we describe the activities that were adopted for the

analysis of the data. We used field notes for capturing the verbal discussion for later analysis. The analysis itself was performed using both techniques of logical chain of evidence and explanation building. In logical chain of evidence, we moved from the initial research questions to ultimate case study conclusions using the evidence collected. In explanation building, we used the evidence collected to revise and refine our initial hypothesis, with the process iteratively repeated. Finally, in both our case studies, a final project review exercise was performed where the case report was presented for review to the project participants to ensure research credibility and accuracy and a questionnaire was filled out by a member of the organization.

1.4.4 Research Approach Suitability

Despite the suitability of both design science research and case studies research methods in our research project, some general challenges are mentioned below,

- **Study Environment:** Performing case studies in organizations adds a layer of complexity to the modeling framework development and needs to be carefully managed. A researcher's primary objective would be in the research outcome whereas the participant's objective may be self-preservation and organizational benefit. In design science research, there is a heavy focus on resultant artifacts which are not necessarily a priority for the participating organization or may not entirely be applicable in that organization's context.
- **Artifact Generation:** Several factors need to be considered during the generation of the design artifacts. Artifact generation for emerging and experimental areas requires some iterations to get them right which may not be acceptable for participants who are interested in immediate resolution of issues. Further, the participant may be unwilling to invest in the effort required for either iterative or overall artifact evaluation [13].
- **Rigour and Relevance Balance:** Establishing and balancing research rigour and relevance in design science research, and case studies, is a challenge as ensuring rigorous research in a particular context may erode its applicability and relevance to a broader domain. Establishing research rigour in the minds of the organization's managerial audience may be difficult due to a rejection of the methods adopted or differences in focus or priorities.

- **Artifact Generalization:** A careful balance needs to be maintained while developing new artifacts using design science research as generalizing these artifacts may result in them being too abstract while having too detailed constructs, models, methods and tools may limit their applicability to other organizations or domains. A set of guidelines are provided in [12] for ensuring overall rigorous design science-based research by emphasizing design evaluation, but this needs to be balanced against the generalizability of the design artifacts which may reduce their overall relevance.
- **Study Limitations:** There were some limitations with regards to the research projects with the industry partners, particularly around the nature and availability of research data and team members. The case studies were time-bounded and a predetermined problem was presented that required understanding and analysis. Relevant stakeholders were made part of the research team and were available to have team discussions and analyzing alternative design choices, and to identify suitable alternatives that would solve the identified problem. In some case studies, additional data was provided in the form of written documents and architectural diagrams to supplement the team discussions.

A document was drafted in the initial phase of all case studies that defined the scope, responsibilities and outcome of the research project. This activity was done to manage better the challenges mentioned above.

1.5 Research Contributions

This research yields both theoretic and practical contributions through advancing current conceptual modeling techniques for understanding, evaluating and analyzing software-induced enterprise transformation activities.

Our *first* contribution is to provide a set of characteristics that are common to software-enabled enterprise transformation, and the limitations of existing modeling approaches when applied to analyze the process reconfigurations in these enterprise. Indeed, while modeling enterprises in their complexity, and considering alternative ways of analyzing and reconfiguring is often central

to these modeling approaches, there are still relatively few propositions that consider the ongoing changes that need to be made in the overall enterprise business processes.

Continuing from this, our *second* contribution is a conceptual modeling framework that comprises a visual modeling language, and accompanying methods, that depict an architecture of business and software processes. The framework emphasizes studying the processes while abstracting aware from process design details and highlights the nature of their relationships rather than the individual process themselves. Existing techniques from goal-oriented requirements engineering were adapted to allow for analysis between alternative configurations of enterprise processes.

This framework is a significant step forward, and it allows for the *structured* contemplation of enterprise transformation, brought about by changes in business processes configuration and disruptive software-based technologies; we refer to these as software-enabled enterprise transformation. Basic process architecture reconfiguration types are included in this framework, along with methods for their implementation in an enterprise context by focussing on involved processes and software artifacts. Further, methods for assessing, reasoning and selecting suitable alternatives are provided while considering trade-offs with regards to enterprise goals. This research builds on the preliminary findings from [26] and [27].

1.6 Illustrative Example

To illustrate the problem and our approach, we chose an example from the financial services domain that offers sufficient richness to allow introducing various aspects of our modeling framework. The financial services industry has been undergoing innovation and change, more recently precipitated by opportunities offered by advancements in software technologies [28] and the competitive threats from new entrants to the incumbents. These new entrants heavily rely on software technologies such as mobile, cloud computing, artificial intelligence (amongst others), along with innovations in the design of software processes to provide services to their customers at lower costs and with rapid cycles of delivery. The incumbents are larger banks that have been around for centuries but are weighed down by existing investments in infrastructure and organization rigidities. The nature of challenges that this industry faces can be patterned across many other industry sectors, such as retail, transportation, and others.

Let us consider a banking institution that is undergoing change. The widespread adoption of software technologies in enterprises is enabling a change in the business model, improving customer experiences, and optimizing operational processes [29]. Here, the enterprise change is being influenced by both the internal adoption of software technologies and the general pervasiveness of software technologies in the environment that they operate. These shifts in how the organization is operated results in digital-first business models that are a significant departure from their previous business model [3]. Here, the focus of transformation should not be on individual systems or processes but should instead be viewed holistically and at an enterprise level. The incumbent banks are expected to adapt continuously to successfully survive in such evolving and uncertain conditions [30]. We discuss three scenarios below.

Scenario 1: Fintech startups and technology companies in the financial services space are offering competing products and services which are eroding the profitability margins of traditional banks [28]. As a result, banks are digitizing their core products like credit cards, loans and payments to better compete with these entrants. A seamless "omnichannel" experience to customers across all service delivery channels is becoming essential, which requires careful coordination of the processes across the enterprise, even if these processes belong to disparate parts of the enterprise.

The nature of change in the organization may be along various perspectives such as strategic vs. operational, transformational vs. transactional, discontinuous vs. continuous, revolutionary vs. evolutionary etc., involving diverse areas such as people, processes, and technology [31]. It is not enough to consider individual processes in isolation for optimization; rather, the collective set of processes would have to be considered and redesigned. But can the interactions between several business and software processes be represented to signify their associations and relationships? The business processes are coupled with the software processes, and thus need to be reviewed collectively for improvement. Software processes could evolve to develop software tools which then are used in business processes, or plans could be developed as part of strategic planning processes that are then used to influence the execution of the software processes.

The dynamic nature of the enterprise invalidates the notion of complete planning before execution. Banks can identify further areas of improvement in the design of their processes, and the creation

of software design artifacts, through big data analytics, which are then optimized and improved upon [32]. Can activities or decisions currently performed in a planning stage be moved to an execution stage, and what are the placement trade-offs? How would changes in the business processes reflect on the need to redesign software to support them? If business processes are to be automated or made simpler, would it require off-loading specific activities within the business processes to be executed by software systems? These are the many questions that enterprise architects struggle to answer.

Scenario 2: As banks increasingly adopt cloud-based infrastructure, they rely on recent innovations in software development processes (such as DevOps) to rapidly develop and deliver software to ensure customer centricity and responsiveness. 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 [33][34][35]. 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 (a) 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, (b) 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, and (c) promoting a culture of collaboration and information sharing between multiple teams.

Analyzing the possible configuration of such a development process for enterprise requires considering many possible alternatives as there is no one prescribed solution. Implementing DevOps can vary from enterprise-to-enterprise and needs to be carefully considered while considering the enterprise functional and non-functional objectives, the existing software processes, and the expected business outcome. Is greater automation of processes preferred or

should human intervention be part of the process execution? How can we represent and reconfigure software processes to introduce faster delivery and release cycles? These are some questions that the enterprise and process architects in the banks have to consider when reviewing a DevOps approach.

Scenario 3: The development of mobile enterprise systems is also an indication of two separate and distinct enterprise segments with different characteristics and timescales. The front-end mobile app development is characterized by quick development and deployment cycles with customers providing immediate feedback through the app store rating, whereas back-end enterprise systems have longer, more cautious development cycles [36]. Thus, a new business feature affecting both mobile front-end and enterprise backend systems would be managed, developed and delivered differently based on the different enterprise levels and timescales.

The goal for both back-end enterprise systems and front-end mobile applications are the same, to provide customers with software features that service the customer's request. However, both sides have significant environment autonomy with defined integration points for the successful completion of this business objective [36]. The processes would need to be designed in a manner where both areas can maintain distinct processes, software tools and environments that are more conducive to their particular needs and user requirements but still align with overall business objectives. Moreover, the business and software processes can encompass various timescales and the implications of the process architecture reconfiguration needs to be considered when moving activities across different timescales. Some of the questions that enterprise architects need to consider are if some activities or decisions be deferred closer to the customer or be part of the back-end enterprise? What would be more suitable approach for improved delivery cadence for the considering such an enterprise solution and how to select the most appropriate one?

1.7 List of Publications and Presentations

Below are all the publications that are based on this research.

1.7.1 Refereed proceeding

Conference Papers

1. Babar, Z., Yu, E.: Integration of Software Applications into Evolving Enterprise Business Processes. In Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS), Vol 2, pp. 778-786 (2020)
2. Babar, Z., Yu, E., Carbajales, S., Chan, A.: Managing and Simplifying Cognitive Business Operations Using Process Architecture Models. In International Conference on Advanced Information Systems Engineering (CAiSE), pp. 643-658, Springer Cham (2019)
3. Babar, Z., Lapouchnian, A., Yu, E., Chan, A., Carbajales, S.: Modeling and Analyzing Process Architecture for Context-Driven Adaptation: Designing Cognitively-Enhanced Business Processes for Enterprises. In Proceedings of the 22nd International Enterprise Distributed Object Computing Conference (EDOC), pp. 58-67, IEEE (2018)
4. Lapouchnian, A., Babar, Z., Yu, E.: Designing User Engagement for Cognitively-Enhanced Processes. In Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering (CASCON), pp. 227-233 (2017)
5. Lapouchnian, A., Babar, Z., Yu, E., Chan, A., Carbajales, S.: Designing Process Architectures for User Engagement with Enterprise Cognitive Systems. In IFIP Working Conference on The Practice of Enterprise Modeling (PoEM), pp. 141-155, Springer Cham (2017)
6. Babar, Z., Lapouchnian, A., Yu, E.: Modeling DevOps Deployment Choices Using Process Architecture Design Dimensions. In The Practice of Enterprise Modeling (PoEM), pp. 322-337, Springer Publishing (2015)

Workshop Papers

1. Babar, Z., Yu, E.: Digital Transformation – Implications for Enterprise Modeling and Analysis. In Trends In Enterprise Architecture Workshop (TEAR), Springer International Publishing (2019)
2. Babar, Z.: A Conceptual Modeling Framework for Software-Enabled Enterprise Transformation. In International Conference on Advanced Information Systems Engineering Doctoral Consortium (CAiSE DC), pp. 11-22 (2019)

3. Babar, Z., Lapouchnian, A., Yu, E.: Modeling Evolving Human User Engagements with Cognitive Advisory Agents using the i* Framework. In iStar Workshop Colocated with CAiSE. (2018)
4. Babar, Z., Lapouchnian, A., Yu, E.: Chatbot Design - Reasoning about design options using i* and process architecture. In iStar Workshop Proceedings, pp. 73-78 (2017)
5. Babar, Z.: Modeling Software Process Configurations for Enterprise Adaptability. In The Practice of Enterprise Modeling Doctoral Consortium (PoEM DC), pp. 125-132 (2015)
6. Babar, Z., Yu, E.: Enterprise Architecture in the Age of Digital Transformation. In Trends In Enterprise Architecture Workshop (TEAR), pp. 438-443, Springer International Publishing (2015)

1.7.2 Non-refereed posters and presentations

1. Babar, Z., Yu. E.: Integration of Data Analytics into Evolving Enterprises Business Processes: Overcoming Enterprise Modeling Challenges, Presentation at Exploring Modeling Methods for Systems Analysis and Development (EMMSAD) conference (2019)
2. Babar, Z., Yu, E., Carbajales, S., Chan. A.: Leveraging Solution Catalogues for Designing Enterprise Cognitive Systems. Poster at CASCON x EVOKE (2019)
3. Babar, Z., Yu, E., Carbajales, S., Chan. A.: Managing and Simplifying Cognitive Business Operations using Process Architecture Models. Poster at IBM CASTLE, Markham, Toronto (2019)
4. Babar, Z., Lapouchnian, A., Yu, E., Chan. A., Mourra. J., Carbajales, S., Pacholski, P., Ngo, K. T.: cogBPM: A Cognitive-Enhanced Modeling and Analysis Framework for Enterprise Business Processes. Poster at CASCON'17, Markham, Toronto (2018)
5. Babar, Z., Lapouchnian, A., Yu, E., Chan. A., Mourra. J., Carbajales, S., Pacholski, P., Ngo, K. T.: cogBPM: A Cognitive-Enhanced Modeling and Analysis Framework for Enterprise Business Processes. Poster at IBM Day at University of Toronto, Toronto (2018)

6. Babar, Z., Lapouchnian, A., Yu, E., Chan. A., Carbajales, S.: Enhancing BPM for Cognitive Operations. Poster at IBM CASTLE 2018, Markham, Toronto (2018)
7. Babar, Z.: Conceptual Modeling Framework for Supporting Software-Enabled Enterprise Transformation. Poster at CASCON, Markham, Toronto (2016)
8. Yu, E., Babar, Z., Nalchigar, S., Danesh, M., Lapouchnian, A.: Cognitive Business Operations Made Easy. Poster at IBM CASTLE, Markham, Toronto (2017)
9. Babar, Z., Yu, E.: Towards an Enterprise Adaptiveness Framework for Digital Transformation. Poster at iSchool PhD Research Days, Toronto, ON, Canada (2015)
10. Babar, Z., Yu, E.: Re-Architecting Enterprises Based on Adaptiveness Requirements. Poster at CSER, Markham, ON, Canada (2014)

1.8 Thesis Structure

The rest of the thesis is structured as follows. Chapter 2 provides a literature review and domain knowledge relevant to this thesis. Chapter 3 discusses the set of characterizations of enterprises undergoing change and transformation and provides requirements for the conceptual modeling framework. Chapter 4 shows how this framework would be used to model and analyze the banking domain example. Chapter 5 discusses the modeling constructs in more detail with Chapter 6 showing how to analyze change in a domain. Chapter 7 and Chapter 8 apply the modeling framework to two real-world case studies for evaluation purposes. In Chapter 9, we conclude this thesis.

2 Literature Review

This chapter presents the related work for this research project.

2.1 Adaptive Enterprises

Enterprises are expected to respond to ongoing changes and evolving environmental factors continuously. Increasing competition and the emergence of new market players from non-traditional sectors require enterprises to react and adapt to change more quickly than ever before [2]. Various types of internally and externally initiated changes need to be determined early, analyzed and be responded to. Such "adaptive" enterprises have many characteristics [7], and should be able to anticipate change in the environment, and respond to it using build-in provisions within the enterprise; these could entail pre-constructing flexible technology infrastructure, having configurability capabilities in software systems, supporting modifiable organization structure, and having a workforce trained to be adaptable. Such provisions require investment and time, and appropriate trade-offs should be considered. In this section, we review the literature on adaptive enterprises and the support provided in enterprise architecture frameworks to analyze such enterprises.

2.1.1 Adaptive Enterprises

Enabling "adaptiveness" behaviour in adaptive enterprises can be explained and understood through the defining of specific enterprise characteristics that cover multiple perspectives, such as process, social and systems [7]. Haeckel [37] provides a differentiation between *make-and-sell* enterprises and *sense-and-respond* enterprises where sense-and-respond adaptive enterprises monitor their environment for changes and accordingly respond to them [37]. In such enterprises, the rigid organization hierarchy is replaced by one that consists of capabilities, where capabilities are modular subsystems (managed by individuals and roles) that are responsible and accountable for outcomes. Moving to such an adaptive enterprise requires (a) setting the context by identifying stakeholders of the enterprise and their objectives, (b) structuring the enterprise as a collection of sub-systems, each with its roles and responsibilities, and (c) determining adaptation loops that

would gather data from the sensing and interpreting portion of the enterprise and pass them to the deciding and acting part of the enterprise.

Introduction of agility in enterprises has been discussed previously [38]; however, these studies generally do not consider the bidirectional adaptation influences between business and technology (particularly those enabled by software systems and business processes). Wilkinson [2] provides a staged-based approach to designing an adaptive enterprise, starting with bringing existing IT assets to a stable stage, then leveraging best practices and automation through technology to get to an efficient stage, and finally removing organizational silos and introducing service-oriented computing infrastructure to get to an adaptive stage.

2.1.2 Enterprises Architecture

The Enterprise Architecture discipline provides the necessary mandate to study the design of adaptive enterprises by emphasizing the inclusion of the business context and environment to the design of enterprise technology and software systems. Hoogervorst [39] presents an argument that enterprise architecture should not just focus on the technology aspect of the enterprise, but should also encompass other architectures as well, such as business architecture, organizational architecture, and information architecture. Each of these represents a particular manner in designing the enterprise, but collectively with the same architectural goals of agility and the ability to change. Unified Enterprise Modeling Language (UEML) allows for the "discovery" and integration of multiple enterprise perspectives however are limited in their ability to reason about the propagation of influences caused by reconfigurations in any one area [40][41]. Enterprise Service Oriented Architecture (SOA) offers a narrower and technology-centric approach towards enterprise agility and adaptability by allowing for business process modifications through an enterprise service-based architecture [42][43].

ArchiMate [44] is an enterprise architecture framework that has been extended in previous years to incorporate goals and rationales (part of the Motivation extension in ArchiMate 2.0) and strategy and physical layers (part of extensions in ArchiMate 3.0). These extensions are indicative of the need for multi-perspective considerations to study adaptive enterprises. ArchiMate has multiple architectural layers (business, application and technology) with the lower service layer

contributing to the higher service layers; the lower layer provides the "primitives" or building blocks that the higher layer arranges into services. Two relationships that cross these layered boundaries are the serving relationship that "serves" to the upper layer functions, whereas the realization relationship indicates a realizing of data objects and application components [44]. Further, ArchiMate considers cooperation amongst business processes by looking at *causal* relationships between processes. Here, processes are mapped onto business functions with the realization of services through these processes [44]. TOGAF ADM allows for migration from an as-is to a to-be state through iterative multi-phased cycles for attaining strategic business needs [45] but does not necessarily support analysis of incremental and ongoing small-scale transitions. For example, the Architecture Development Method (ADM) in TOGAF supports enterprise architectural change in response to a business need, but ADM is an iterative exercise with many phases, explicitly designed to transition the enterprise from an as-is state to a to-be state [45]. However, it does not cater to periodic and variable changes, including the ability to decide between multiple alternative enterprise configurations at run-time.

2.1.3 Summary

While enterprise architects are well equipped to model and reason about enterprise architectures using the frameworks described above, there are limitations when it comes to modeling and analyzing about the nature of multi-perspective (i.e., systems-, enterprise-, and process-level factors) changes that are introduced to enterprises due to ongoing technology and software innovations. E.g., in ArchiMate, there are distinct layers between business and technology, and it is not apparent how processes can be migrated from one layer to another, or the consequences of such a migration. There is a limited notion of partiality in the relationships, so a serving relationship cannot be shown as partially fulfilling the requirement.

A certain level of uncertainty exists in the enterprise as the enterprise continually reacts to changing circumstances. Despite this, sociotechnical rigidities and barriers to change exist that resist change in the enterprise. The frameworks do not offer techniques that allow enterprises in dynamic and uncertain environments to design suitable enterprise processes while being repeatedly informed through feedback loops and considering the multiple types and levels of processes with

their complex relationships, including the integration of software systems. Some enterprise architecture frameworks (like ADM in TOGAF) do allow for ongoing change but do not cover the full range of the transformative requirements for the enterprise, such as those presented in Chapter 3. Therefore, these frameworks should have modeling constructs that support diversity and variability along multiple dimensions of adaptiveness, including the ability to decide among different architectural configurations.

The research in this PhD project differs from the aforementioned related work as it deems fundamental transformation requirements of enterprises from multiple perspectives (such as process-based and goal-based) while reasoning about interplays and influences amongst (as well as within) these perspectives. These transformation requirements are presented in Chapter 3. Such an analysis can lead to continuing process redesigns due to process restructuring and altered requirements for software tools and artifacts.

2.2 Business Process Management

Business processes help understand the basic functioning of any enterprise. The designing and architecting of business processes include notions such as specifying the relationships, dependency types, structure, composition and associations that exist between them [50], including additional process relationships, such as the sequencing of information flows, triggers, specialization, reference, and composition [51]. There exist multiple "core elements" that need to be present to be able to design and reconfigure processes due to evolving organizational needs [52]. In this section, we review some conventional approaches to modeling and analyzing business processes.

2.2.1 Business Process Modeling

A popular business process modeling notation is BPMN [53]. While activities can be shown, along with changes in their sequencing, the implications of any activity reordering cannot be determined. BPMN models do show feedback loops, but the full range of attributes associated with them (for example, the multitude of timescales present in the loop or the recurrence of the sensing and responding parts) are not evident. Process participants are used to represent abstract roles in BPMN, but these roles cannot be used to indicate intentionality or motive. These processes have a

sense of duration as described from start and end events, and in-between process constructs. BPMN and other traditional business process modeling languages rely on an imperative approach where the process model represents (in great detail) the process state of the system and all permitted actions. However, capturing such detailed specifications of the system-under-study is challenging, particularly as the underlying processes may be ever-changing. Declarative process modeling notation (such as BPMN-D) allows the capturing of constraints on activity flows [54], i.e., any flow is permitted as long as the restrictions are upheld.

Other approaches in business process management have focused on the role of "artifacts" (uniquely identifiable and self-contained flow entities) within process design and execution. This is necessary as without considering artifacts, it would be difficult to consolidate processes to see how they can attain a common goal. Business participants often are too focused on the execution of processes without understanding the reasons for the execution, thus having an understanding of the information context is necessary in order to design business processes properly [55]. Elsewhere, specific business tasks are considered to be encapsulated functions that act on these business artifacts [56]. Artifact-Centric Operational Modeling (ACOM) is an approach that emphasizes identifying artifacts that traverse the complete process, and aids in systems development and specification, rather than purely lending itself to business analysis [57][58]. These artifacts "capture the contexture of a business and operational models describe how a business goal is achieved by acting upon the business artifact" [59]. An alternative conceptualization of a business artifact (and its lifecycle in business processes) is provided in [60] where an artifact "is a concrete, identifiable, self-describing chunk of information that can be used by a business person to actually run a business" and is "taken to be the only explicit information contained in the business; that is, the set of business records represents the information content of the business."

Business processes are modelled from the perspective of information entities or data flows in the information-centric approach. An information-centric process model of a process scope may contain multiple information entities, with information entities being the data that are used by business functions, and the input and output of different business services [61]. Declarative data-centric approaches for business process design are useful as they provide an understanding of

various business artifacts that are managed within the process flow, including the operations performed [62]. The details captured in such declarative workflow specifications make it easier if there is a need to come up with physical implementation. Business processes can be considered to cross many "layers" in the organization, these being the operations layer, execution layer, and implementation layer [63]. A sense-and-respond organization would have data flows (through business processes) across these layers that would allow it to communicate data from the sensing part to the deciding part. Finally, the impact of cognitive computing on business process management is covered in [64] where multiple types and levels of business processes are discussed; these include transaction-intensive, judgement-intensive, and design & strategy support processes. These processes result from the incorporation of cognitive capabilities within an enterprise and how cognitive processes enablement can be attained.

2.2.2 Business Process Redesign

Business processes need to be periodically redesigned. This is often a response to the changing state of the organization itself, or as a response to the changes in the external environment. While these redesigns can be done in an unstructured and creative manner, there are systematic means of analyzing and proposing alternative designs. Dumas et al. [50] suggest two methods of redesigning business processes, heuristic-based and product-based design. Seven elements are considered to enable structured contemplation better and to redesign a process from a heuristic standpoint. These elements are customers, business process operation, business process behaviour, organization, information, technology, and external environment. The product-based design method considers a different perspective where the central focus is on designing a particular enterprise product or service while ignoring the existing design of that process. This frees the process architect from the constraints of existing process design and permits a design that is the most efficient for the creation of that product or service. Another approach to business process redesign is provided in [65] as a framework where the emphasis is on the mechanics of the process. This framework highlights several elements that should be central to the redesign of a process, including best practices and commonly accepted ideas.

Another relevant domain is business process variability modeling which focuses on representing customizable business process models through *variation points* used to describe and bind variability at design-time [66]. Introducing variability at design-time means that all instances of the business process execution follow the same design configuration. Conversely, flexibility can be added in business processes at run-time where customization decisions are made that affect different instances of the process execution, but not the process model itself [67]. Variation points are specific locations in the business process where decisions are made (at *binding time*) for selecting alternative process configurations that help accomplish particular enterprise objectives. In dealing with business process flexibility, [68] proposes four dimensions of change that help determine the relationships between process fragments and the late selection for these fragments. Overall, these approaches deliberate about variability only at the process level (i.e., within a single process) and do not support reasoning about and within business processes nor do they guide ongoing enterprise transformations.

The concept of process families is discussed in [69], where several processes, with minor differences between them, are considered as variations of one meta-process. Such a way of viewing collections of processes as a process family allows for generalization and contemplation of design choices around automation, cost, simplicity, management etc. However, this work does not sufficiently differentiate between design-time process variants of run-time variability in process execution. A method of determining process variants and design-time, and reconfiguring the process at run-time is provided in [70] through a five-step method which involves eliciting and describing variability, determining and analyzing the context, linking the non-functional requirements to different process variants, and finally reconfiguring the process at execution time.

There have been attempts to model context in problem domains. The VIVACE framework is derived after a systematic literature review into handling process variability [71]. The framework allows for comparing existing process variability approaches and selecting an approach that best meets the requirements of a particular situation while also factoring in the application context. Context Modeling Language (CML) allows for capturing of various fact types with regards to related objective types [72]. Fact types are further differentiated into static facts, dynamic facts (such as profile facts, sensed facts, derived facts) and temporal facts. Other standard modeling

approaches like UML and ER have been used for context modeling. However, they are limited in their ability to capture specific distinctive characteristics of contextual information [72]. For example, UML class diagrams model user, personalization, and context metadata subschemas together in one model [73].

Context-driven process adaptation has recently been given much consideration. A formal representation of context, using a metamodel, for business processes for a domain is given in [74]. The metamodel is defined across three layers (context, process, and domain) that collectively support the representation of relevant contextual variables (associated with a process model within the domain under study). Such a metamodel helps with adapting business processes based on available context. A Process Management System, consisting of a model and a prototype, is provided in [75][76]. This additionally features a set of techniques for supporting the run-time adaptation of knowledge-intensive process instances in response to unanticipated exceptions. Situation calculus is used to model context with planning systems used to execute the automated adaptation of processes using encoded action plans. A context model-based approach and planning technique is proposed in [77] to tackle the problem of dynamic adaptation within a process-aware information system by characterizing unexpected situations as known as contextual elements. This helps automate the decision of process flow replanning while ensuring process strategy is still attained. Various characteristics of the problem domain to define the context in which systems are to operate are considered in [78] with a methodology proposed for exploring context variability while modeling and analyzing its effects on requirements goal models, whereas an illustration in managing and monitoring context to redesign business processes is provided in [79].

2.2.3 Business Process Architecture

Considering a multitude of business processes as a collective to understand their relationships, exchanges of data, and how services are realized has been previously studied in scholarly literature. The concept of Business Process Architecture, and the sequence and hierarchy of business processes, how and why to split process stages, etc. is covered in [50]. Eid-Sabbagh et al. [51] define additional ways of considering the relationship between processes by considering notions such as composition, specialization, trigger, and information flow. Business process architectures

are used to provide an abstract representation of multiple processes that exist in an enterprise. Dumas [50] distinguishes between three types of relationships that exist in a process architecture - sequences, decomposition and specialization. Process architectures can also be seen as a means for developing a more holistic view of the organization by associating business process modeling and enterprise architecture, while additionally abstracting processes into a higher level of granularity that provide increased visibility on the constituent parts of the integrated processes [6].

2.2.4 Summary

Business processes are generally studied and modelled in isolation without considering their inter-process relationships and structures. Multiple business processes may come together to provide some feature functionality, but the nature of their relationship is not explicit in modeling languages, such as BPMN. Pools are used to show a multiplicity of processes that operate independently of each other and the inter-process connections show sequence relationships between these processes and the exchange of messages and artifacts. Despite this, the multiple levels of process-driven dynamics and the relationships between the process levels are not apparent in a BPMN model, nor are boundaries between these process levels obvious. The relationship notations in BPMN are limited in the sense that they cannot convey the degree of configurability between the relationships of processes where one business process is producing a plan that is being used by another business process in its execution, or when software processes are building software artifacts that are used elsewhere. Variability in business process design is well covered by existing literature but these ignore the influence of surrounding processes on business process design.

The existing approaches for studying and modeling business process architecture provide a representation of a collection of business processes under study. They do not offer constructs or mechanisms for studying enterprise transformations or provide mechanisms for dealing with uncertainty in the design of the business process architecture. While similar to the idea of business process architecture, the approach in this research project differs by emphasizing alternative process constructs and various means of reconfiguring the process architecture for enabling fundamental transformations in the enterprise. We are focused on the need for ongoing change in the enterprise and use process architectures to model those changes and analyze possible variants

of process architecture configurations that can exist. We ignore the differentiation of processes as business processes or software processes. Rather the focus is on how multiple processes come together to achieve a common objective and how to reconfigure the process if certain associated non-functional objectives change.

2.3 Software Processes

Software processes are a collection of numerous activities involving many organizational units and individuals performing various roles for the generation of different kinds of software artifacts. Software processes have been defined as "the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product" [4] and "activities, methods, practices, and transformations that people use to develop and maintain software and the associated products" [206]. These definitions imply that the process of software development does not exist in isolation from the rest of the organization and is influenced by multiple organizational and technological factors.

Software processes are continuously evolving, with numerous innovations to software process design being introduced over time [42]. Enterprises continue to customize and tailor software processes to suit their local environments, different team cultures, software project priorities, organizational policies, software tools, and functional requirements. Several approaches to designing and modeling software processes have been proposed over the years to allow for process transformations and improvements, and support design decision making [4][80]; some of these are discussed in the subsequent sections.

2.3.1 Software Process Modeling

Several techniques and methods have been proposed for modeling software processes to help with the design of these processes, along with providing insights into their execution [42]. These software process models allow the depiction (through various means) of the activities that need to be executed to accomplish process objectives. The participants involved in the enactment of the activities can also be identified, including the roles that they play. Further, the nature and form of

software artifacts that are produced by process execution can be discovered alongside the software tools that contribute towards the development of these artifacts [42].

A categorization of activity-oriented and artifact-oriented software process models is provided in [80]. *Activity-oriented* models focus on the activities and methods that comprise a software process, whereas *artifact-oriented* models focus on the resultant artifact output produced through process execution. The activity-based approach focuses on the various activities, methods and tasks that collectively contribute towards defining the overall software process [80]. Breaking up the software process into finer-grained activities enables the assessment of the key actions that need to be performed (as well as understanding the requirements for that action) for developing the software product. This further allows the redesign of the software process by shifting around, substituting or even repeating activity segments based on varying enterprise needs. Individual activities could be considered for localized improvements and automation. Associations could be defined between the activity segments, which would indicate the nature of the relationships, the ordering present, and any control and information flow, all of which would give a more in-depth insight into the implementation of the software process.

Several software process modeling techniques exist which are activity-based. Situational Method Engineering (SME) can be used to create development (software process) methods for specific purposes by selecting and combining method fragments previously-stored in method repositories [81]. These method fragments can be either activities or product artifacts, and the method combination is done in a manner to meet the demands of the situation. Software & Systems Engineering Metamodel (SPEM) is a meta-process modeling technique for designing software process models particular to specific enterprise needs [82]. SPEM abstracts out key software activities. By ignoring specific process modeling languages, implementation and execution details, construction of various process models (and their ongoing evolution and redesign) is possible without being burdened by software process nuances. The Unified Process (UP) [83] and some of the techniques in the Unified Modeling Language [84] can also be considered an activity-based approach by their emphasis on activity-based development. This is not to say that activity-based approaches ignore the presence of software artifacts or tools. Instead, their primary focus is on the software activities with the artifacts that may or may not be well described or included [80].

In contrast to the activity-based approach, the artifact-based approach focuses on the artifacts which are to be produced as part of the software development process. Such an artifact-centric approach ensures that the software artifact is given primary importance, with the software process being designed around it [80]. The requirements, nature and structure of the artifacts thus need to be well understood with the processes contributing to its development. It would be prudent to define here what is meant by an artifact. An artifact "is seen as a structured abstraction of modeling elements used as input, output, or as an intermediate result of a process. An artifact has particular properties (structure, behaviour, etc.) and can be precisely described using standardized (semi-)formal modeling concepts. Upon such a description, one can incorporate different techniques and notions, define clear responsibilities and support a progress control for the production of artefacts" [85].

The advantage of the artifact-based approach is that the software process can be designed to be flexible, modifiable and less concrete as long as it manages to meet the functional and non-functional requirements of the artifact. The stakeholders need not be bothered about defining the software process in great depth and detail as long as there is an understanding of what artifacts would be produced by process segments; minor process segments and activities can be determined by the process participants themselves based on their conveniences and efficiencies. Modeling techniques from requirements engineering can also be used to define the problem space (for artifact requirements elicitation), which would allow opportunities for richer and deeper analysis and design of the software process [85]. Some modeling techniques under the UML umbrella can be considered to be artifact-based approaches to modeling software processes. Another example is that of V-Modell XT, where modeling process defines the resultant artifacts [85].

2.3.2 Software Process Variability and Adaptability

Designing software processes for enterprises is a complex activity and needs to be carefully done, considering each enterprise's unique characteristics. Software processes that have been successfully adopted in one enterprise may not necessarily serve the needs of another enterprise. Besides, multiple software processes may exist within an enterprise, each serving some specific project and having certain contextual considerations that cause them to be different from each

other. Determining commonalities and variabilities between these software processes would be beneficial in many ways. Sharing of design experiences would reduce the time spent in designing unique software processes while reducing the risk of less optimum process design, which may result in wasteful activities being performed or project budget being impacted [86]. Further, being able to manage minor variations between software processes enactments efficiently allows the contextualization and refinement of these processes at a certain granular level. For example, allowing individual teams to customize the software process based on their needs while staying within the broader enterprise mandated software process design. Customized software processes can be created and discarded as per project needs.

In Situational Method Engineering (SME), allowing the archiving and usage of software process fragments enables variability in software process design through the creation of customized software processes [81]. Software Process Tailoring (SPT) and Software Process Improvement (SPI) methods aim to mould a general software process to a particular project by "adding, removing or modifying the activities and the required inputs/outputs of a base process model to develop high-quality system/software efficiently" [87]. There is a plethora of approaches within process tailoring on how to achieve improvements through variability, customizability and reusability through the mixing or matching software process components for new process creation or the instantiating of customized process architectures using process architecture templates [87].

Extending the idea of Software Product Lines to processes results in the notion of Software *Process* Lines (SPrL) [87], 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. SPrL allows for the reasoning of alternative configurations by considering the placement, and their binding, of variation points, thus facilitating software process reuse. Another approach for considering variability in software processes is by abstracting out the specifics of the process implementation and designing the process at a meta-level. SPEM, being at a meta-process level, does not specify the particulars of the software process. For example, while it may indicate that a requirement elicitation stage is required, the exact approach to be used may be left out. SPEM also provides different variability constructs that can be extended (as per specific implementations) to give concreteness to a SPEM process design [82].

Adaptability is characterized by some form of feedback loop which provides opportunities for adaptation through the selection of one out of many variations. All agile development practices have in-built mechanisms for self-evaluation and retrospection; in fact, having a post-iteration review is one of the principles of the Agile Manifesto [88]. The Scrum methodology has a "sprint retrospection" ritual where the recently concluded sprint is reviewed and activities rated according to whether they should be started, should not be done, or whether certain activities should continue to be done [89]. Despite having sound practices of feedback, the reflection and retrospection rituals in agile are manually initiated, unsystematic and subjective. Due to the unstructured manner of feedback collection, interpretation and implementation, it is difficult to measure the effectiveness of these feedback sessions on the overall improvement of the agile software development process being practiced in the enterprise. The reflection and retrospection rituals in agile development practices are usually represented as simplistic block diagrams and non-standardized modeling notations. While these diagrams may provide primitive representations of the activities involved in adaptive software process design and are relatively easy to understand, they do not lend well to reasoning and analysis, particularly when trying to compare alternative software design configurations.

2.3.3 Software Process Modeling for Variability and Adaptability

Some techniques exist for formal or visual capturing of variability aspects of software processes [117]. As mentioned in the previous section, SPEM provides specialized variability constructs for illustrating variability in a software process. SPEM 2.0 consists of various elements such as *role*, *work product*, and *task* in addition to a *variation element* [82]. Variation elements allow the introduction of variation and extension functionality to other SPEM elements (i.e., role, work product and task) through *variability types*. There are four variability types, *contributes*, *replaces*, *extends*, *extends-replaces*. The *contributes* variation type allows the addition of the properties of a variability element to the base variability element. The *replaces* variation type allows the substitution of the properties of a variability element to the base variability element. The *extends* variability type allows the inheritance and extension of the properties of a base variability element to the variability element. A combination of the last two variation types is *extends-replaces*, where the variability element can replace specific defined properties of the base variability element while

extending others. SPEM 2.0 provides visual modeling notations for denoting each of the above cases. Here the “base variability element” refers to the component which is abstractly depicted in the SPEM model and the “variability element” is the element that contributes to particular concrete implementation.

Variability aspect of SPEM 2.0 is indicated in [118], however in a somewhat constrained manner as it (a) does not provide solutions for tailoring the software process (despite solutions for substituting, extending or adding a base process), (b) does not offer variability specific notations (but reuses the UML association relationship), and (c) does not provide guidance on the modifications to the work element resulting from the process variation. To overcome these stated limitations in SPEM v2.0, the authors present vSPEM where the concept of variation points and variants (borrowed from SPL) is introduced to SPEM and are defined as “places at which variations occur, and the elements may be different from one process to another. Variants are specific implementations of this variability, and each one of these variants makes the process unique” [119]. Both variation points and variants are abstract classes for SPEM elements, and the concrete class represents the different alternatives possible for that abstract SPEM element. Additionally, vSPEM provides variability specific notations to identify variability in software processes better and quickly.

UML activity diagrams too have been proposed to representing variability in software activities by introducing stereotypes (<<VarPoint>>, <<Variant>> and <<Variable>>) to standard UML activity diagram notations [120]. Another approach for modeling variability is through feature modeling. Feature models were introduced as being used to represent variability in SPLs; however, they can be similarly be used for software processes as well [117]. In this context, mandatory and optional software process elements are represented and marked as features with the standard feature modeling analysis techniques being applied to derive alternatives. Finally, from a requirements engineering perspective, agent-oriented and goal-oriented modeling techniques are used to supplement a software process modeling visualizing by showing the different goals and the alternative means of achieving the higher-level goal [121]. These goals map to process elements in the software process and the alternatives indicate the different configurations of those process elements that are permitted for attaining the process objectives.

There is limited modeling support for software process adaptability. System dynamics is a feedback-oriented approach for modeling complex continuous systems through mathematical and graphical modeling. System dynamics provide structured and established modeling techniques using graphical process depiction and equations, including diagrams such as stock and flow diagrams and causal loop diagrams [49]. System dynamics has been applied to a wide range of areas, including managerial decision making and organizational behaviour. The design of software processes has also benefitted by the application of system dynamics [90]. Since software processes can be considered to be processes with inputs, outputs and feedback control elements, thus the principles of system dynamics can be used for adapting the software process.

2.3.4 Summary

Software processes are increasingly seen as contributing towards organizational strategy [1] and becoming an integral part of operational processes [42]. Software process design cannot be done in isolation from the rest of the organization and should consider various organizational considerations, in addition to the nature and need of the software being developed and the team that is responsible for developing it. Indeed, software processes can be considered to be more "complex and unpredictable than typical production processes as they depend on people and circumstances" [207]. As a result, the design of software processes is a fairly difficult activity that requires considerable insight into multi-level objectives, i.e., at an organizational level, at a team level, at a software architecture level, at a customer level etc. [208].

As part of our PhD research, we ignore the traditional boundary between software processes, operational processes and business processes, and focus on the collection of processes that come together to attain some enterprise functionality. This allows us to consider the design of software processes in conjunction with other processes. Most of the software process design approaches described in the previous sections support better software development and production activities, including automating processes, by identifying processes, resulting artifacts (to be produced or used), and the process participants. However, they do not sufficiently provide tools and methods for reasoning about design alternatives (for both process and artifacts) while evaluating enterprise objectives. They do not provide mechanism for periodically studying software processes

reconfigurations to account for organizational variations while fulfilling high-level requirements. Ongoing changes in process design are handled in our research by providing process constructs that allow for various degrees of configurability to meeting ongoing shifts in soft-goal attainment.

2.4 Software Systems

It is no longer economically feasible or acceptable to have multiple disparate versions of software systems developed and maintained separately, each designed to function in a specific situation [46]. Further, organizations are shifting from developing single systems to a domain-specific family of systems, which further adds pressure for reuse between software systems. This requires modern software systems design to factor commonalities and variabilities between families of systems, in addition to the architectural design of individual applications. Evaluating and analyzing families of systems for commonalities and variabilities can be done by studying the problem space (for variability requirements) or by examining the solution space (for software architectural variability). The following sections consider modeling variability from both perspectives with a few techniques discussed for illustrative purposes.

2.4.1 Software Systems Variability and Adaptability

There has been a conscious effort to make software systems utilizable across a range of conditions and situations. This requires the designing of software systems to have variable behaviour based on differing requirements that are presented to them and enable the system to be extended, customized and configured for use in multiple contexts [46]. Thus the system is described as having *variability* in design, behaviour and execution, with variability helping with "delaying constraining of the system." System designers are thus able to offer a broader range of products and services (more economically) through the late selection of variants. Variability in software systems is influenced by two underlying forces [46],

- Moving the embedding of variability behaviour from hardware to software allows the introduction, modification and embedding of system variability at a far lower cost.
- Delaying the design decision points to a later stage in a system's lifecycle based on economic considerations.

Variation points (VPs) are specific locations in systems where decisions are made to select *variants* of the system component or system design [46][47]. Variants can be alternative system components or system design elements that help accomplish particular system design objectives. One of the possible variants is selected at *binding time* in order to commit an alternative at that variation point. The system does not need to be designed with all possible variants determined beforehand; in some instances, the variants may be later added to the system.

Systems are further expected to demonstrate *adaptability* characteristics where they are supposed to adapt in response to various external and internal stimuli. The adaptation may be self-initiated (i.e., *self-adaptive* systems) or may be initiated through some form of manual or external initiation. There is a distinction between system *adaptability* and *adaptiveness* [7] where a system is considered to be adaptable if it is "easy or amenable to change" whereas system adaptiveness refers to the "ability of an entity (organism or system) to change its behaviour to better survive or succeed in its environment." In order words, the adaptability of technological systems contributes towards the adaptiveness of systems which operate at a higher (business) abstraction level.

System adaptability is inclusive of some form of feedback loop that allows a system to monitor input sensory data, analyze and evaluate this data and undertake corrective actions as a response [48]. Further, this adaptation is enabled through the selection of variants that provide alternative ways of achieving system objectives. Stock and flow diagrams allow for the visualization of the various software process components and activities, the associated software flows, and information linkages [49]. The elements in a stock and flow diagram are the *level*, *source and sink*, *rate*, *auxiliary*, and *information link*. The various software tasks, software artifacts, individuals and participants, activities performed can be considered to be *levels*. The *sources and sinks* would be things outside the boundary of the software process under consideration. The various software actions being performed in the system being modelled would be considered to be *rates*. *Auxiliaries* could be regarded as the different software metrics that exist in any software process environment, metrics such as percentage accomplished and code coverage. Finally, *information links*, progress, customer feedback, milestone and status information help with the flow of information.

Causal loop diagrams show cause-and-effect linkages and feedback loops in a system along with positive or negative influences that may exist [49]. Causal loop diagrams are advantageous over stock-and-flow diagrams in the sense that they can be quickly drawn and do not require detailed modeling and specification of levels and rates. In the case of adaptive enterprises, they can help show the various feedback that is received from the stakeholders and process participants. These can include software engineers, test engineers, product analysts and even the customers. Both causal loop and stock and flow diagrams can help with simulation modeling of business processes, which can further enable adaptive tendencies in the design of enterprises.

2.4.2 Requirements Modeling for Software Variability

Various requirements engineering (RE) techniques elicit requirements for variability between multiple software systems that occupy the same domain space. Agent-oriented requirements engineering (AORE) [91] and goal-oriented requirements engineering (GORE) [92] techniques provide a means for defining and assessing goals which are to be achieved by software systems. Goal and agent requirements models allow for depicting higher-level goals, which can iteratively and recursively be decomposed into multiple sub-goals with OR decompositions between these sub-goals. Each sub-goal represents a means-to-an-end, i.e., one of many alternative ways (as indicated through the OR decompositions) of attaining the higher-level goal. From a variability perspective, the higher-level goal can be considered to be a variation point with the sub-goals being the possible variants. A suitable alternative (variant) can be selected based on positive or negative influences that that variant has on different non-functional requirements, represented as softgoals in AORE and GORE [93]. The evaluation and selection of alternatives can be done using techniques proposed as part of the Non-Functional Requirements (NFR) framework [94], and through the qualitative or quantitative methods discussed in [95]. Examples of AORE and GORE techniques include i* [91], Tropos [96] and KAOS [97]. Requirements engineering techniques have been proposed to assist and aid in software systems adaptability by focusing on the problem-space [98]. These techniques are primarily focused on the design of software adaptive systems, and less so on the dynamics and complexities of the relationship between software systems, business and software processes, users and participants, and the alignment with enterprise business objectives.

Scenario-based approaches provide another way of modeling requirements variability in the problem space [99], with scenarios described as "projections of future system usage, thereby helping to identify requirements" and one scenario being "one sequence of events that is one possible pathway through a use case". A multi-view variation modeling technique for developing product families is proposed in [100] what considers alternative scenarios that "describe a spectrum of possible futures that affect the architecture." Multi-view here refers to the usage of five different modeling views - namely functional, conceptual, realization, application, and customer – for determining the impact of identified commonalities and variations along each one of those view dimensions. The views themselves are integrated through annotations or linkages amongst the modeling elements (of each view). Problems Frames (PF) have also been proposed for studying requirements variability. Problem frames is a *problem-oriented* conceptual framework for requirements analysis, which emphasizes a focus on the problem domain rather than the solution [101]. The context of the problem (captured using context diagrams) plays an essential role within problem frames, and changes in context may result in different system behaviour. The context in the problem space can be considered from a variability standpoint [102], and problem frames can be used for representing and reasoning about contextual variability by considering them as variant problems. Despite the support for variability assessment, scenario- and problem-oriented approaches do not support systematic methods for analyzing and modeling variability for the complete problem domain, whereas the goal- and agent-oriented approaches allow capturing of domain variability requirements by starting with a high-level goal and gradually and recursively decomposing into lower sub-goals.

2.4.3 Domain Modeling for Variability

Analyzing and modeling a domain can help in identifying commonalities and variabilities between software applications existing in that domain [103]. A model of the domain would help with abstract description, identification of relationships between key constructs, and determination of commonalities and variabilities between individual software applications [104]. Domain analysis generally comes under the broader domain engineering umbrella for economically developing Software Product Lines (SPL). Software product lines allow an enterprise to identify and define shared software components and artifacts in a product family, thus enabling the launch of multiple

disparate products in a shorter time and lower cost by taking advantage of the commonalities between them [105]. Various domain engineering approaches exist which can be used for analyzing and modeling variabilities in a domain. The Family-Oriented Abstraction, Specification, and Translation (FAST) approach consists of three sub-processes (domain qualification, domain engineering, and application engineering), which helps organizations determine a viable product line, develop suitable product line artifacts, and finally build software products using those shared artifacts [106]. This approach is in contrast to traditional software processes that focus on a single software project or product. FAST evaluates a product domain and produces a domain model, in the form of a commonality analysis document, "which is a record of the family's terminology, commonalities, and variabilities, and the key issues that arose during the analysis" [106].

Another approach, Product Line UML-Based Software Engineering (PLUS), extends UML (which helps in designing single systems) to consider multiple products by introducing some additional modeling notations and techniques [107]. PLUS supports three categories of modeling for domain analysis – requirements, analysis and design modeling – with many modeling techniques within each category. The Product Line Software Engineering (PuLSE) approach seeks to focus on the organizational context, rather than the general domain, when conceiving, developing and deploying product lines as the organization is deemed to be a strong influencing factor while designing product families [108]. The PuLSE methodology consists of three "elements" – deployment, technical, support – each further refined into phases or components as applicable. PuLSE contains both graphical models (for showing the flow of activities within phases) and a tabular map (for displaying a combination of software characteristics and the products that they map to) for modeling the application domain. Finally, the Domain Analysis and Reuse Environment (DARE) approach uses multiple sources – such as product code, technical documents, and domain experts – to determine domain variability through models for determining opportunities for software automation and reuse [109].

2.4.4 Feature Modeling for Software Variability

Software systems comprise of several software features that provide specific functionality. Features are an intuitive way of expressing and understanding a software's purpose and

characteristics. Variants of software differ with regards to the features that they offer, and thus, software variability can also be expressed with regards to the commonalities and variabilities of features across multiple products. Models which allow the expression and analysis of commonalities and variabilities of software features are referred to as feature models [110]. Feature-Oriented Domain Analysis (FODA) supports "the development of domain products that are generic and widely applicable within a domain" [111]. In FODA, variability of product design is obtained through three methods (a) aggregation/decomposition (multiple units instead of one monolithic unit), (b) generalization/specialization (a conceptual unit that can be instantiated into different forms), and (c) parameterization (unit adaptation into various forms based on parameters thus enabling variability). FODA feature models are a means to capture the "general capabilities of applications in a domain" through a graphical tree-like structure where each node represents a feature [111]. In this model, features can be decomposed recursively into sub-features, with linkages showing the relationship between features and variability being represented by labelling features as *alternative* or *optional*. Feature Oriented Reuse Method (FORM) extends FODA from the requirements engineering phase to the systems design phase by utilizing feature models for designing system architecture and code by adding appropriate constructs to feature models [112].

Various modeling techniques are proposed that use UML models in conjunction with feature models to design product families. Here feature models are used for tracing overall domain variability while UML models help with designing individual software systems. Reuse-Driven Software Engineering Business (RSEB) is an approach that utilizes use-case models for driving software reuse within a product family, with variability being captured through the explicit definition of variation points in use-case diagrams [113]. FeatuRSEB proposes the inclusion of feature models (from the FODA approach) into RSEB to complement the RSEB models and capture feature-based variability and commonality [114] as part of the overall domain analysis. *Feature* tags for associating elements in a UML component diagram is introduced in [115]. An abstract (feature-based) representation of variabilities in product lines (coming from the feature model) is mapped to architectural components spanning multiple software products (as illustrated by UML component diagrams). Unlike the previous approach, which introduced UML lightweight extensibility mechanisms (i.e., tags and stereotypes), [116] advocates for making changes to the

UML use-case meta-model by adding two new relationships, *option* and *alternative*, and one new model element, *variation point*. This evolved use-case meta-model is combined with feature modeling for overall domain analysis.

2.4.5 Summary

Software designers can offer a greater range of products and services by delaying design decision points to a later stage in a system's lifecycle based on economic considerations. This necessitates having an understanding of how the software system is used within the enterprise setting. Postponing design decision points to a later point is only possible if the necessary data for decision making is available. Similarly, building a software artifact later means that there should not be a need for its use in business process execution until that point. Hence, software design should consider the context in which the software is going to be used within business processes.

In this research, we do not specifically focus on the design of software systems but emphasize the building and use of these software artifacts within the broader process architecture. This allows for differentiating between processes that are responsible for the building of software, and processes where the software is used, thus indirectly indicating the software artifacts that need to be developed, including the location within the overall process architecture in which they are to be used. We also consider the possibility of having partial designs of software artifacts, where certain design decisions are left to be resolved at the time they artifacts are to be used.

3 Understanding Software-Enabled Enterprise Transformation

Acknowledgement: This chapter is partially based on the following papers;

- *Babar, Z., Yu, E.: Digital Transformation – Implications for Enterprise Modeling and Analysis. In Trends In Enterprise Architecture Workshop (TEAR), Springer International Publishing (2019)*
- *Babar, Z., Yu, E.: Enterprise Architecture in the Age of Digital Transformation. In Trends In Enterprise Architecture Workshop (TEAR), pp. 438-443, Springer International Publishing (2015)*

There is an opportunity for researchers in the conceptual modeling and enterprise modeling community to provide enterprise architects and process architects with methods that would help organizations to become ever more adaptive in fast-moving and rapidly evolving environments, particularly taking advantage of emerging software-based technologies such as big data analytics, artificial intelligence, blockchain, cloud computing, etc. Before we do so, we first need to understand what we mean by the phrase "software-enabled enterprise transformation" by providing a characterization of this phenomenon. There exist several factors that should be considered whenever enterprises are incorporating digital technologies and software to foster innovation and change. Such commonalities across multiple organizations and industry segments can be extracted as a set of characteristics. Each of these characteristics may have been extensively studied in isolation, and have appropriate solutions proposed by scholars and practitioners. However, the problem posed by the characteristic may have been solved in a limited context while generally ignoring the collective impact these characteristics have on the enterprise.

3.1 Recent Trends in Software-Enabled Enterprise Transformation

We studied three current industry trends, digital transformation, bimodal organizations, and adaptive enterprises, all of which rely on the recent emergence of digital technologies and software innovation to help transform the organization through the introduction of new business models and digital-based strategies. There may be other relevant trends as well, but we were able to identify several characteristics through these to be able to proceed to determine the requirements for the modeling framework. Each of the three trends is further discussed below. In a review of these topics in the remaining subsections of this chapter, we specifically view them through the prism

of business and software processes, as our motivation is to address enterprise change through identification, analysis, and management of alternative process architectural configuration.

3.1.1 Digital Transformation

Digital transformation entails the transformation of core business operations of an enterprise by leveraging digital technologies [3][122]. Such a transformation is a significant shift from the previous modus operandi and results in broad-ranging and potentially disruptive enterprise-wide transformation enabling enterprises to move from a brick-and-mortar style operation to one that is more encompassing of digital technologies [123]. There is no one agreed definition of digital transformation, although recent literature reviews [124][125][126] attempt to provide a set of properties for enterprise digital transformation.

The major drivers for digital transformation are digital technologies, digital capabilities, enterprise strategies and evolving business models; with there being an impact on the products and services offered by the enterprise, the processes that produce those products and services, as well as the overall organization structure [126]. The operational processes would need to be optimally designed to align with and support enterprise strategic objectives, including those around customer experiences. For this, enterprise architects require an enterprise modeling framework that would provide a systematic and structured mechanism for managing change in the enterprise at multiple layers and perspectives.

3.1.2 Two Speed or Bimodal Organizations

Organizations born in the digital age are better able to meet customer expectations as they do not have legacy business processes or IT infrastructure weighing them down. Consequently, *two speed or bimodal IT architecture models* have been adopted by traditional organizations to be able to stay competitive and remain customer-centric [127]. The term “two-speed” refers to the relative frequency at which each “section” of the enterprise operates, with the management of the customer-centric front-end systems being separated from the legacy back-end enterprise systems to allow for independence of decision-making and operations.

Such an approach is useful as it allows organizations with significant slow-moving business models and technology infrastructure to stay competitive, albeit this does come with several challenges [128] for the enterprise architect. Having disconnected sections of the organization means that there is less proliferation and exchange of ideas. Sociotechnical challenges with integrating these two disconnected segments remain, with strategic initiatives never really having a cross-organization complementarity of offerings. Finally, both the financial and non-financial cost and complexity of having such an architecture is significant and needs justifying against the perceived (and unproven) benefits that would result from such an enterprise architecture.

3.1.3 Adaptive Enterprises

Organizations are seeking new ways to become more agile and adaptive [7]. As mentioned in Chapter 2, an adaptive enterprise is one in which the output of the organization (in the form of goods or services) is continually changing while being synchronized with the expectations of its customers. Adaptive enterprises are in a continuous reorganization state, with the change being influenced by both the internal adoption of technologies and the general pervasiveness of digital technologies in the environment that they operate in. Such enterprises have "sensing" and "responding" characteristics, with them needing to "observe" and be aware of such situations based on which it would initiate and undertakes activities of adaptation and change.

Such paths of change can be analyzed in terms of sense-and-response loops through which the enterprise continuously adapts and improves [37]. In the sensing part, the enterprise would (proactively or reactively) determine the cause and need for change. In the responding part, the enterprise would determine the best possible alternative for change. The trait of adaptiveness in an enterprise is a desirable goal, but this often competes with other objectives. As with bimodal organizations, the cost of maintaining a state of adaptability in the organization could be high and must be balanced against other enterprise goals [7]. Further, IT resources and capabilities need to be designed with flexibility in mind to ensure agility and an ability to react to change in the market conditions [129].

3.2 Methodology

A systematic literature search was performed using the eight-step process provided in [11] for conducting literature reviews in information systems research. This process is visually presented in Fig. 3-1. Through this review, we intended to determine the underlying characteristics of software-enabled enterprise transformations by identifying behavioural commonalities across multiple enterprises undergoing transformation using the three trends introduced in the previous section.

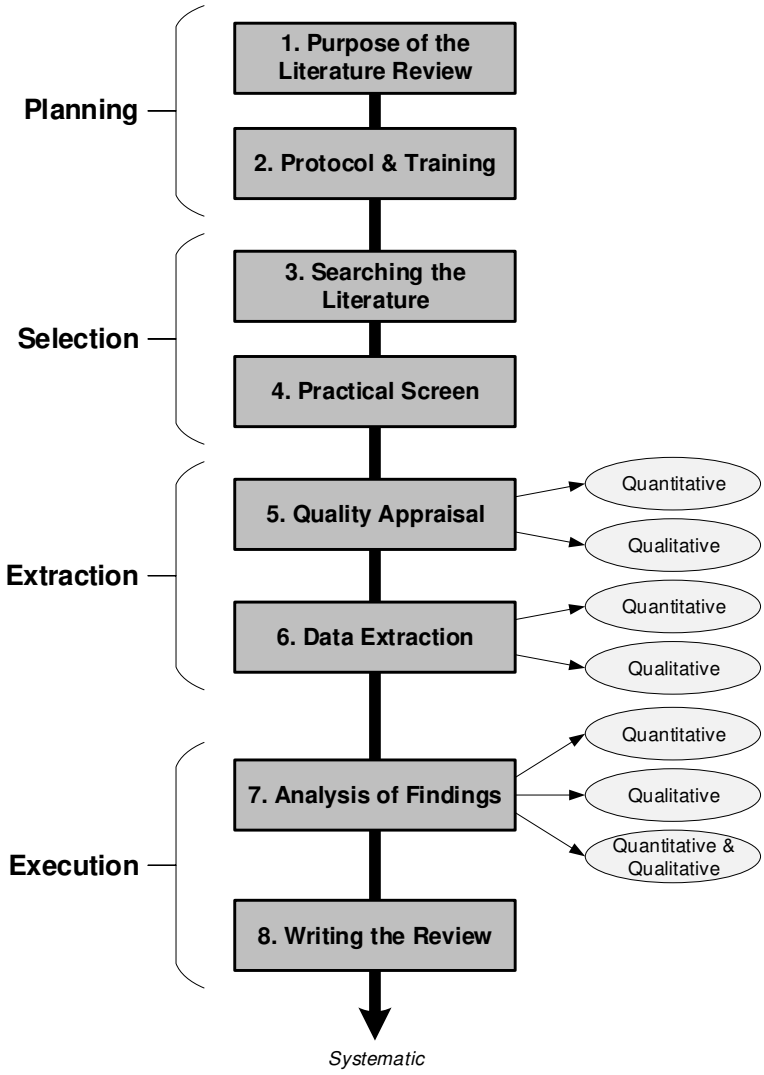

Fig. 3-1: Systematic literature review method adopted (Source: [11])

Step 1: Purpose of Literature Review

We first portray the problem by identifying the characteristics of software-enabled enterprise transformation. Our intention was to use these characteristics to come up with definite requirements for an enterprise modeling technique that would allow modeling and analyzing enterprises undergoing transformation due to emerging digital technologies. We emphasize that the purpose of this systematic literature review was not to provide a precise definition and description of software-enabled enterprise transformation, but rather to identify common underlying traits in such enterprises.

Step 2: Protocol and Training

A protocol or training document wasn't needed as the review employed only one reviewer.

Step 3: Searching for the Literature

We started with an inclusion criterion and identified all those papers that used the term "Digital Transformation", "Two Speed Organizations", "Bimodal Organizations", "Adaptive Enterprises" in the paper title, abstract or keywords. Although different phrases (such as "Digital Strategy", "Enterprise Digitization", etc.) may be considered as a viable alternative, we chose to limit our search to just the ones mentioned as adding alternate terms to the research may lead researchers into a biased understanding, as not all terms are semantically similar to our selected terms.

Step 4: Practical screen

The search for articles was conducted in early 2019 and only for journal articles and conference papers that had a publication date of 2010 and later, as we wished to focus on enterprises that were transforming due to emerging digital technologies. Further, only papers written in English were selected. Using this inclusion criterion, we ran a search query against the ProQuest database, which returned a total of 818 articles. By manually reviewing the search results, we identified 120 duplicates which were eliminated to obtain a final list of 698 articles (see Fig. 3-22 for distribution by year).

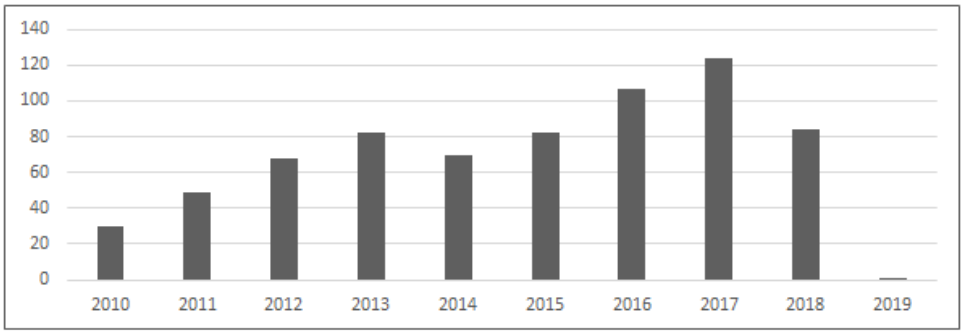

Fig. 3-2: Distribution of selected articles by year

Step 5: Quality Appraisal

In the screening process, we reviewed the title and abstract of the 698 papers to determine if the papers attempted to define attributes, characteristics, or adoption challenges for enterprise transformation. We specifically considered the following questions during this phase of the review process. Does the article cover one of the following points in detail?

- Define enterprise transformation?
- Discuss its characteristics?
- Discuss the primary drivers in its adoption?
- Specify adoption challenges in enterprises?
- Share experiences in real-world settings?

During this screening processing, we found out that most papers superficially introduced or mentioned such enterprise transformations in passing. These papers did not qualify based on the screening criteria set above and were excluded. The final selection consisted of 36 articles that were then reviewed in more detail in the next step.

Step 6: Data Extraction

In this step we extracted segments from across the 36 papers that based off the following thematic areas,

- Traditional and Digital Business Models
- Operational Processes
- Emerging Digital Technologies

- Customer and Customer Experiences
- Organization Culture
- Social and Employee Implications
- Legacy Technology Infrastructure

The thematic areas were identified by studying the drivers and impact areas as identified in [37][124][125][126][127][128][129]. Collecting and grouping these paper segments based on the thematic areas was important as it simplified the data to be synthesized in the next step.

Step 7: Synthesis of Studies

Through a process of qualitative reasoning, each thematic area was coded to determine the nature and types of discussion points and arguments presented. Through this we determined eight concrete characteristics across our final data set (shown in Table 2). We were careful only to isolate those characteristics that were present across multiple papers. The paper count by characteristic is given in Fig. 3-3. The total number of articles exceeds 36 as multiple characteristics may appear in an article.

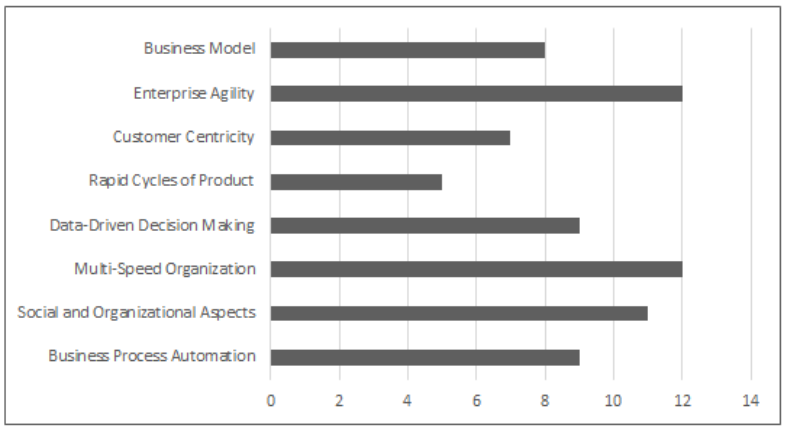

Fig. 3-3: Distribution of articles by characteristics.

Step 8: Writing the Review

In this section, we provided the systematic literature review process employed for independent reproduction by other researchers. We do acknowledge that the review relied on qualitative reasoning and analysis of articles. It is conceivable that other reviewers executing a similar review

process may thus see slightly different results or uncover additional characteristics. These could then be used to determine a set of requirements for an enterprise modeling framework.

3.3 Characteristics Relating to Enterprise Transformation

The literature review exercise performed resulted in the identification of eight concrete characteristics that are presented in Table 2.

Table 2: List of Characteristics and related papers

Characteristics	Papers
Business Strategy and Business Models	Berman [130]; Resca et.al. [131]; Lan & Lui [132]; Schallmo et.al. [133]; Remane et.al. [134]; Kotarba [135]; Matt et.al. [136]; Loonam et.al. [137]; Delmond et.al. [138]
Enterprise Agility	Delmond et.al. [138]; Earley [139]; Berman & Marshall [140]; Westerman [141]; Hossain & Lassen [142]; Heavin & Power [143]; Andriole [144]; Burden et.al. [145]; Shrivastava [146]; Narayanan [147]; Kaivo-Oja et.al. [148]; Shaughnessy [149]
Customer Centricity	Berman [130]; Loonam et.al. [137]; Shrivastava [146]; Narayanan [147]; Shaughnessy [149]; Wahi & Medury [150]; Weill & Woerner [151]
Rapid Cycles of Product and Solution Delivery	Kaivo-Oja et.al. [148]; Shaughnessy [149]; Wahi & Medury [150]; Weill & Woerner [151]; Masuda et.al. [152]; Troilo et.al. [153];
Multi-Speed Organizations	Berman & Marshall [140]; Hossain & Lassen [142]; Andriole [144]; Burden et.al. [145]; Shrivastava [146]; Narayanan [147]; Shaughnessy [149]; Wahi & Medury [150]; Troilo et.al. [153]; Masuda et al. [157]; Basole [158]; Alos-Simo et.al. [159]; Ardolino et.al. [160]
Data-Driven Decision Making	Schallmo et.al. [133]; Delmond et.al. [138]; Berman & Marshall [140]; Westerman [141]; Hossain & Lassen [142]; Narayanan [147]; Troilo et.al. [153]; Gölzer & Fritzsche [154]; Pikkarainen et.al. [156]; Masuda et al. [157]
Social and Organizational Aspects	Resca et.al. [131]; Loonam et.al. [137]; Heavin & Power [143]; Andriole [144]; Narayanan [147]; Shaughnessy [149]; Kolbjørnsrud et.al. [155]; Alos-Simo et.al. [159]; Schwarzmüller et.al. [161]; Sainger [162]; Nwaiwu [163]; Andriole [165]
Business Process Automation	Schallmo et.al. [133]; Westerman [141]; Heavin & Power [143]; Andriole [144]; Kaivo-Oja et.al. [148]; Shaughnessy [149]; Weill & Woerner [151]; Kolbjørnsrud et.al. [155]; Weber & Monge [164]

This list of characteristics is not meant to be exhaustive or absolute, as the identification process was based on qualitative reasoning and may be prone to observer bias. Our intention of discovering these characteristics is to develop an understanding of the key challenges in modeling enterprises that are underdoing enterprise transformation. A narrative for each is provided below based on common discussion points across the papers in the final dataset.

3.3.1 C1: Business Strategy and Business Models

Business strategies and business models are utilized by enterprises to develop and maintain competitive advantages in a changing landscape [130]. Alignment between an enterprise's business strategy and operational processes allows for improved enterprise performance [130]. Through the usage of emerging digital technologies, enterprises are now increasingly developing digital business models (or digital models) [132][135]. Cloud computing platforms, in particular, have allowed for the creation of new digital business models, such as eCommerce or SaaS based solutions, and enabled evolving enterprise strategies [131]. Enterprise business and technology processes need to be supportive of enterprise strategies and need to be aligned, designed and tailored accordingly [136]. In a changing environment, enterprise strategies and business models evolve, and software process need to be flexible and adaptive enough to serve the needs of such evolving needs through closer and ongoing association between business and software development functions [138]. Additionally, enterprises can adopt agile approaches to requirements engineering for managing the challenges of rapidly changing technology, or by aligning innovation in business models and changes in business strategy with the organization process setup and design [137] [138].

3.3.2 C2: Enterprise Agility

Enterprises undergoing transformation due to disruptive software technologies are responding to ongoing changes and evolving environmental factors, increased competition, and the emergence of new market entrants from non-traditional sectors [140]. Disruptive technologies and continual business model innovation require enterprises to react and adapt to change more quickly than ever before [140][141]. Emerging technologies are used to inform and shorten product development cycles and increase product release cadence [142]. To this end, enterprises are expected to be

adaptable by relying on rapidly configurable IT and software systems and accompanying processes for the development and delivery of appropriate products and services with agility and responsiveness [138][145]. Similarly, IT and software processes too need to be supportive of enterprise strategies to deal with transformation challenges and need to be aligned, designed and tailored accordingly [143]. Recent innovations in software processes and artifacts allow for faster and rapid software product development [149]. Such improvements can be in both the developed software system *and* the software processes that develop such systems. Collectively, such an approach would allow enterprises to become more agile and responsive.

3.3.3 C3: Customer Centricity

Enterprises strive to offer continuously improving customer experience through greater customer-centricity with respect to the products and services that they offer [130]. Recent advances in digital technologies allow for greater customer involvement and engagement between the customer and the enterprise than ever before [149]. Thus, a significant motivator for ongoing software process and product evolution and innovation is the ability to rapidly and efficiently satisfy emerging customer preferences and trends [146]. Organizations are making their processes more agile by bringing their software development processes closer to the user so as to respond to evolving customer trends and change requirements faster [147]. Such a shift allows for more significant customer consideration during the design, develop, deliver and operate phases of the final solution.

3.3.4 C4: Rapid Cycles of Product and Solution Delivery

Continuing from the previous points, enterprises need to have faster and rapid cycles of product and service delivery [149]. Prompt delivery of software features enables an enterprise to reduce the time-to-market of new products and features, provide greater customer-centricity by introducing new features based on evolving customer needs, quickly resolve operational and support issues, and show improved responsiveness to changing internal context and external environmental situations [150]. Rapid cycles can be achieved by either increasing the frequency of process execution or through the reduction in time required for a cycle execution. Practically these are attained by reconfiguring various segments within business and technology processes or reducing the number of activities within [151]. Technology and software systems, tools and

processes are being redesigned for ensuring continuous and rapid delivery of software products and artifacts while supporting on-going evolving enterprise cycles of innovation. These redesigns are done by considering the multi-faceted implications of introducing such practices within an enterprise [153].

3.3.5 C5: Multi-Speed Organizations

Enterprises that are undergoing change due to adoption of software-enabled technologies often have distinct areas, with each having separate culture, processes, methodologies, software tools and environments that are locally relevant and conducive [145]. This is particularly evident in two-speed organizations, where one side is more responsive to customer needs by allowing it to be "decoupled" from the side containing legacy systems (and its associated processes) [146][147]. Such separation is attained by adopting multiple approaches and can involve diverse perspectives, including organizational structure, separate technology and software systems, and differing business, IT and software processes [149]. Digital technology has allowed customers to be increasingly engaged with enterprise service providers. The technology solutions used within the digital enterprise transcend two distinct areas having different characteristics and timescales. The side closer to the customer is characterized by quick development and deployment cycles with customers providing immediate feedback, whereas back-end enterprise systems have longer, more cautious development cycles [158]. Thus, business features affecting front-end and enterprise backend systems would be managed, developed and delivered differently using different enterprise areas, levels, methodologies, tools, and timescales.

3.3.6 C6: Data-Driven Decision Making

Within any enterprise, there exist data-driven cycles of ongoing feedback and improvement that are considered during planning activities [153]. Advances in big data technologies are enabling the capture, retention and processing of large volumes of enterprise data, which are then utilized as part of decision making for incrementally improving on operational processes, strategic decision making, product design, amongst other areas [154][156]. Analyzing process execution allow for the identification of inefficiencies in operational process execution, which can then be used to re-engineer business processes better [147]. The use of feedforward loops is also evident in order to

have the enterprise respond to changing environmental conditions. Software engineering methodologies, too, are increasingly utilizing enterprise feedback and feedforward loops to help improve in decision making for software process and software systems design [140].

3.3.7 C7: Social and Organizational Aspects

Changes to the enterprise are frequently accompanied by changes in roles and responsibilities of organizational units and individuals [155][161][162]. These organizational changes need to be carefully and systematically deliberated while considering changes to enterprise processes, and technology and software systems [165]. Creating and understanding these multi-faceted associations is difficult while factoring in the complexities of enterprise architecture and design, as well as other enterprise considerations such as culture, context, and alignment [161]. It is generally convenient to ignore such social and organizational aspects during the design and inclusion of software and technology and focus on how an activity is to be performed or what it entails, yet the why also needs to be understood to glean out the complex social relationships between the various enterprise and process participants [155][165]. The rise of popular software methodologies and concepts, such as Microservices Architecture [166] and DevOps [33], has been the direct result of requiring visibility and alignment between the social, technological, and process perspectives, with each being configured in a manner that supports the other

3.3.8 C8: Business Process Automation

Enterprises are increasingly investing in business process automation to improve process efficiency, reduce cost, and improve execution time [155]. Through the use of advanced software and technology, automation is resulting in changes to crucial enterprise processes [151]. The widespread support for business and software process automation through recent technological advances enables operational support for both enterprise business activities (by helping with automated decision making) and the technology environment (by aiding with software systems configuration, deployment, monitoring and maintenance) [164][149]. However, automation brings about certain complexities, and it needs to be justified against its perceived benefits [141]. Automation results in business processes operating at different timescales i.e., the manual development activity may take days, whereas the automation and deployment may now take hours

[143][148]. From a technology standpoint, automation has strongly influenced the software development and delivery model by allowing for short release cycles and more rapid delivery of software features.

3.4 Requirements for the Modeling Framework

The characteristics presented in the previous section can be abstracted out as a set of requirements for a modeling framework. In Table 3, we provide a matrix mapping between these transformation characteristics and the requirements; this is in addition to referencing the applicable characteristics (e.g. C1, C2, C3, and so on) within each requirement.

Table 3: Mapping Enterprise Transformation Characteristics to Framework Requirements

Characteristics								Selected Highlights for Requirements
(C1) Business Strategy and Business Models	(C2) Enterprise Agility	(C3) Customer Centricity	(C4) Rapid Cycles of Solution Delivery	(C5) Multi-Speed Organizations	(C6) Data-Driven Decision Making	(C7) Social and Organizational Aspects	(C8) Business Process Automation	
	✓		✓		✓			**(R1) Process Architecture**: Represent the overall software ecosystem through processes and their relations. Configuring processes would allow for introducing different enterprise behaviors while attaining enterprise objectives.
	✓		✓	✓				**(R2) Multi-Level Process Dynamics:** Indicate several process types and their levels. Different levels can be demarcated through process boundaries, with each level having similar process behavioral attributes.
✓	✓	✓						**(R3) Enterprise and Process Goals:** Align enterprise strategy and business models to processes responsible for delivery of products and services. Shifting enterprise objectives are attained through process reconfigurations.
✓						✓		**(R4) Trade-Off Analysis:** Compare possible process configuration alternatives against priorities of process participants, systems complexity, enterprise objectives through trade-off analysis using enterprise non-functional objectives.
			✓				✓	**(R5) Abstract Software Artifact Design:** Consider bidirectional influences between the design of software artifacts, and the design of surrounding enterprise processes, with software being designed with flexibility and adaptabilty in mind.
			✓				✓	**(R6) Design-Use:** Differentiating between designing and usage processes allows for configuring the process domain (along with supporting software artifacts) for greater process automation or human dependency.
			✓					**(R7) Plan-Execute:** Differentiating between planning and executing processes allows for configuring the process domain for flexibility or stability of execution.
		✓	✓		✓			**(R8) Feedback and Feedforward Paths:** Rapidly incorporate feedback from sources for reconfiguring process architecture and associated software artifacts, thus enabling continuous and ongoing improvements.
	✓	✓		✓			✓	**(R9) Process Cycles:** Confirm faster software delivery cadence for improved customer centricity and enterprise agility while comparing and selecting against multiple possible process configuration and software automation.

For each characteristic, we reasoned what affect it would have on the processes in the enterprise, the objectives that the enterprise was trying to attain, the impact on software design, and how data could be used within the process execution. We were able to deduce the requirements presented in this section through this process of reasoning. We acknowledge that there could be additional requirements were not discovered during our study and which could be uncovered in subsequent qualitative analysis. Stating the requirements in such a manner guides the development of a framework that would allow enterprise architects to analyze enterprises undergoing transformation while considering complexities of software systems and business process design, and stakeholder motives and intentions.

3.4.1 R1: Relationship Among Processes

Every organization relies on many processes that together ensure its success and viability, and to introduce certain agility in its operations (C2). Enterprise modeling techniques need to express and reason about the nature of the relationships amongst the various business and software processes and their resultant artifacts. The relationships are essential in the current context of dynamic enterprises, as these relationships are themselves subject to change. The structuring and capturing of overall enterprise and associated software processes would result in a process architecture that shows the various process segments, the nature of relationships amongst them, and any exchange of data or artifacts. Modifying the nature of the processes, and their relationships allows the enterprise to adapt to different cycles of solution delivery (C4). Identifying locations of data availability is important as it allows understanding of the possible process changes that can be introduced (C6). Holistically considering multiple processes, and their relationships to each other, allows for more meaningful analysis that goes beyond a single process, and permits asking the following,

- How would the interactions between several business, technology and software processes be visually represented to signify their associations and relationships?
- What are the architectural implications when changes are made to the process architecture for attaining particular enterprise business objectives?

3.4.2 R2: Multiple Types and Levels of Processes

Because process architectures encompass multiple individual processes, there can exist several "levels" of process dynamics within a process architecture, making the enterprise more amenable to change (C2). Different types of processes, for example, planning processes, design processes, operational and transactional processes, etc., may take place over different timescales and have different frequencies of occurrence and execution (C4 and C5). Some processes provide inputs to other processes. These levels of dynamics are not entirely evident to the casual observer, nor are the boundary transitions (between these process levels) apparent. Thus, a process architecture would have to incorporate details and attributes that would allow for the identification of, and differentiation between, various process segment types and levels. These types and levels would be delineated through process boundaries and all process segments within would share similar attributes and behaviour. The constituent activities of any process could be moved across level boundaries, still there would be resulting implications which need to be understood.

- Would it be possible to identify and aggregate similar process segments (in the overall process domain) with regards to their behaviour and attributes?
- How would the movement of the activities across level boundaries and the resultant implications be captured and understood?

3.4.3 R3: Enterprise and Process Goals

Enterprises and local units have defined functional and non-functional objectives that align to the business strategy, with business processes assigned to attain them (C1). For introducing and maintaining enterprise agility, enterprises need to reconfigure their business and software processes while considering trade-offs amongst non-functional goals (such as customer centricity) based on organizational priorities (C2 and C3). These objectives can be viewed as functional requirements and non-functional requirements and goal-modeling techniques be used to represent, compare and contrast alternative business, technology and software process configurations for that domain. Including enterprise and process goal-oriented perspective to the overall domain study allows for identification of process structures (in the process architecture) that collectively work

together to serve some common (higher-level) goal. Associating the process architecture with goals allows for the following analysis,

- What are the processes that collectively contribute to some enterprise goal, and can the link between processes and goals be represented?
- Can there exist multiple process configurations that allow for the attainment of an enterprise goal at the expense of different non-functional goals? How to compare, contrast and select between these alternatives?

3.4.4 R4: Trade-Off Analysis

Introducing innovation through technology can no longer be considered to have limited implications and must be regarded along multiple perspectives. In many cases, there are several possible reconfigurations of the process architecture that align to the attainment of strategic enterprise objectives (C1). However, such possible alternatives need to be confirmed against other aspects, such as the availability of data, the priorities of process participants, and implications on systems complexity and any consequence to those enterprise goals and objectives (C7). Trade-off analysis would need to be done to consider the impact of and to various systems-, enterprise-, process-, and social-level factors. Analyzing and deciding between process architecture configurations (along with supporting software tools design and usage) can thus be done by evaluating the satisfaction of objectives. Some of the recent innovations and approaches in software engineering emphasize multi-dimensional perspectives by incorporating factors pertaining to software design, software development and deployment processes, organizational structure, and enterprise culture. Questions such as the following can be asked,

- Would there need to be a change in roles and responsibilities mandated by reconfigurations to process architectures? How would such trade-offs be evaluated?
- How should the organization structure and software delivery pipeline processes be aligned to reflect the optimum software architecture?

3.4.5 R5: Abstract Software Artifact Design

There exist bidirectional influences between the design of software artifacts and the design of surrounding enterprise processes that need to be studied and reflected upon. Often, software needs to be designed with adaptability in mind to enable run-time decision making and having flexibility in how the software is delivered (C4). Software can also be used as tools for processes that are themselves responsible for software artifact development, thus facilitating process automation (C8). Detailed software design artifacts can be produced using modeling techniques such as UML diagrams, which allow for precise software development and implementation. However, abstract distinctions need to be understood to permit conceptual and visual analysis of the software's contribution to, and participation in the overall enterprise business and technology processes, particularly those which need to be altered to introduce change.

- How to incorporate the contribution of software artifacts and tools to the overall enterprise processes while meeting enterprise objectives (such as automation)?
- How would changes in the enterprise processes (or their objectives) reflect on the need to redesign software to support them (and vice versa)?

3.4.6 R6: Pushing Design Decisions Downstream

There exist two levels of processes, one where the process is responsible for the creation of a tool, capability or artifact while the other being responsible for (repeatedly) using the designed artifact. Different designs (along with their associated processes) may be prepared based on how they are to be used by downstream processes. Some design decisions would be deferred to at runtime (or use-time) to ensure some flexibility in the use of the design artifact (C4). Consequently, there exist many possibilities regarding the degree of designing that should be done before which an artifact can be used by a human user or as part of some process. These can range from *full designing* where all design decisions have been made for that artifact and the user must "use" the artifact, to *minimum designing* where many choices regarding artifact use are left at run-time to allow for the most considerable flexibility in the artifact usage. Each possibility would be accompanied by complex trade-offs. Software can be designed to create persistent capabilities in the form of software tools and artifacts that can repeatedly be (re)used during automated process execution

without being aware of how the capability is constructed (C8). Such variations in design-use process configuration permits the following questions,

- Should some activities or decisions be deferred closer to usage time to take advantage of near-real-time data thus creating flexibility in artifact use?
- Is greater automation of processes preferred (with no runtime decision making), or should human intervention be part of the process execution?

3.4.7 R7: Upfront Planning vs. Deferred Planning

Processes modeling techniques (such as BPMN) generally describe the activities that are to be executed, but not how these activities are planned or determined. Such a consideration is vital for enterprises undergoing change, particularly around changing their solution delivery to meet varying customer expectations (C4). Different plans (along with their associated processes) may be prepared based on how they are to be executed by downstream processes. Some plan-related activities may be left for later because of the unavailability of updated contextual data, analytics, or to ensure some flexibility in process or system design. Thus, there exists a degree of planning that can be done that would influence the behaviour and execution of downstream processes. This can range from *full planning* where each downstream activity is thoroughly "planned" out by providing detailed instructions or constraints, to *minimum planning* where many decisions regarding process execution are left at run-time / execution time. These possible process configuration alternatives would have to be carefully considered against potential trade-offs. Categorizing the processes as planning or executing allows the following questions to be answered,

- Should activities or decisions currently performed in a planning stage be moved to an execution stage, and what are the placement trade-offs?
- What degree of planning (full to minimum planning) be done for downstream execution activities?

3.4.8 R8: Feedback and Feedforward Paths

The enterprise needs to "observe" and be aware of evolving situations, based on which it would initiate and undertake activities of transformation and change to meet shifting customer

expectations (C3). Such paths of change can be analyzed in terms of sense-and-response loops through which the enterprise adapts and improves (C4 and C6). Sensing and responding take place in business and technology processes that exist at different levels of dynamics and timescales. For example, the sensing part can happen at machine timescales (through the use of automated data-driven systems) with the acting part existing in human timescales (through managerial decision making). In the sensing part, the enterprise would determine the cause and need for change. In the responding part, the enterprise would discover the best possible change alternatives. These feedback and feedforward paths need to be mapped out in the process architecture and tagged to activities. Such techniques would allow for,

- Sketching out linear paths or cyclic loops as they exist in any enterprise while indicating the interactions that the paths may have with other paths or objects.
- Depicting the various timescales that a loop traverses and the process-level implications in reconfiguring these loops or moving activities across different timescales within the loop itself.

3.4.9 R9: Represent and Reason about Speed, Timescales and Process Cycles

Software-enabled enterprise innovation and transformation allow for increased enterprise benefits, such as automation, higher productivity, and improved release cadence (C2 and C3). Different environments may require different approaches to be adopted. However, these often come at the expense of other enterprise considerations, like an increase in complexity of the solution, higher cost, reduced flexibility, reluctance in adoption, etc. Sometimes there are deviations within, where the software artifact itself dictates the speed of delivery that should be adopted. The processes may be executing at different process cycles depending on which part of the enterprise they belong to (C5). The process cycles need to be represented in a manner that would allow for possible reconfigurations to improve execution frequency and changing process execution activities (C8). Enterprises need to be aware of questions, such as,

- How to represent and reconfigure processes to introduce faster software delivery and release cycles for their customers?
- What is the more suitable approach for improved delivery cadence amongst the various possibilities mentioned above? How to select the most appropriate one?

3.5 Inadequacies of Existing Techniques

There are several conceptual modeling techniques available to model and analyze enterprises. In this section, we consider two popular modeling techniques and their inability to deal with the complex challenges of modeling enterprises that are dealing with software-induced change as per the requirements presented in the previous section.

3.5.1 BPMN

The first modeling language that we consider is BPMN. Here, we model two processes using this notation. For the sake of simplicity, we do not show the entire domain example mentioned in Chapter 1 but rather just the primary participants and processes that are necessary to accomplish the business outcome. This is still sufficient to illustrate some of the limitations of BPMN with regards to satisfying the requirements presented in the previous section.

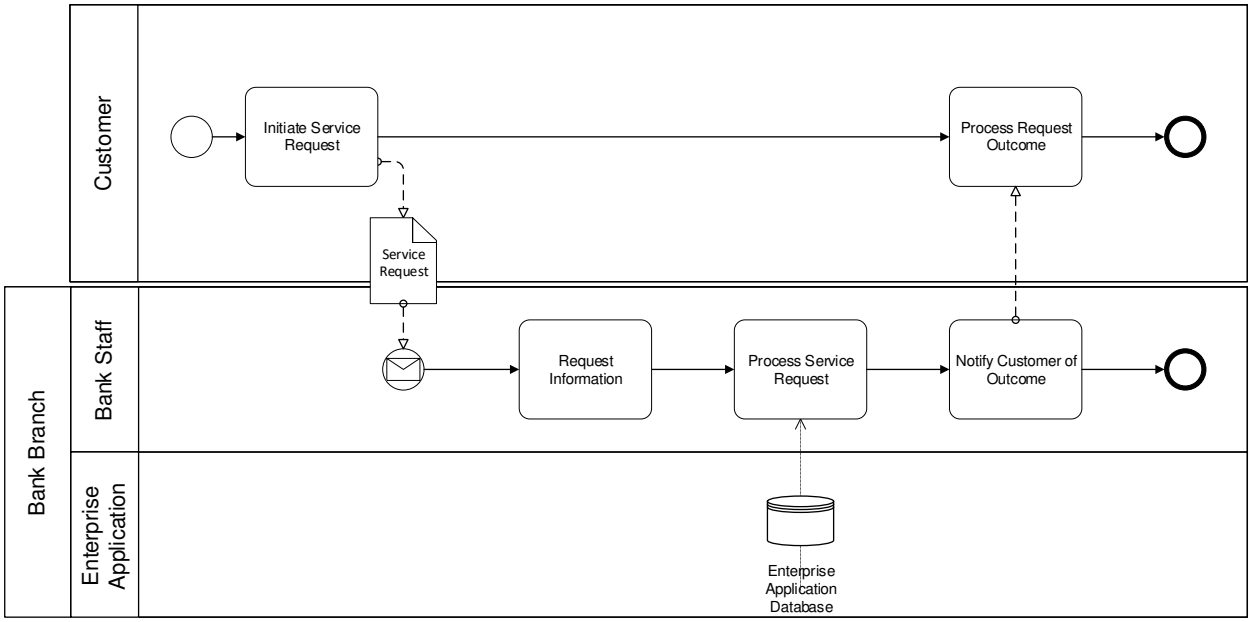

Fig. 3-4: A simple BPMN model presenting the customer request processing by bank staff

The first process is a simplistic representation of the servicing of a customer at a bank branch by a staff member. Fig. 3-4 shows the main participants involved in this business process, i.e., the **Customer** and the **Bank Staff**, with the **Enterprise Application** also shown as a process participant. The business process starts once the customer arrives in the bank branch and **Initiates Service**

Request. The **Bank Staff** receives the **Service Request** and processes it using some data artifact from the **Enterprise Application Data**. The business process ends once the customer receives the outcome of the service request processing.

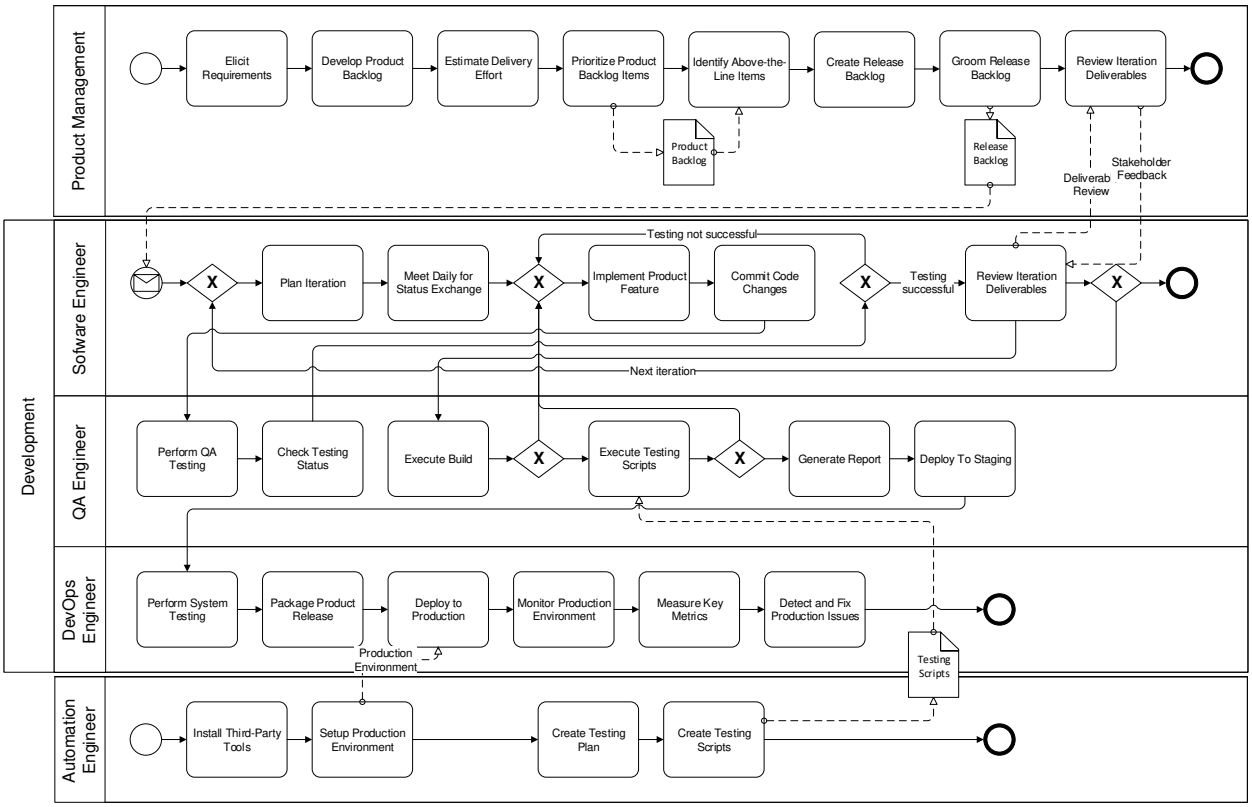

Fig. 3-5: A BPMN model representing a typical DevOps approach

We show another process in Fig. 3-5 indicating the primary participants and the significant activities in a typical DevOps-inspired software process that exists in this bank. We have developed this context by referencing published literature from multiple sources, such as [168][169][170], intending to highlight how the various activities in DevOps can be better configured to serve a variety of enterprise functional requirements and non-functional requirements. 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. 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 reconfigurability of software process configurations.

In our sample enterprise setting, there could be various forms of change occurring in these processes. We discuss some examples of these below,

- Any developed feature has to be functionally tested before it goes through the continuous delivery 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 they can collaborate with the software engineer to quickly validate the functionality before the codebase is committed to the code repository. While activities for both situations can be shown through BPMN models, along with the changes in their sequencing, the implications of any activity reordering cannot be determined.
- 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., re-ordered and re-estimated) based on an on-going change in circumstances and priorities. There is no way to distinguish such a situation using the BPMN notations, where the outcome of some prior processing is repeated used or is regenerated each time.
- DevOps is characterized by the usage of third-party tools for continuous integration and continuous delivery, 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. 3-5 as a separate **Automation Engineer** pool. Here there exist multiple process levels, with the outcome of some processes (i.e., creation of DevOps tools) at one level feeding into those at an upper level (usage of those tools for deployment). The multiple levels of process-driven dynamics and the relationships between the process levels are not apparent in the BPMN model.

The multiple business and software processes shown in Fig. 3-4 and Fig. 3-5 are coming together to provide some feature functionality, but the nature of their relationship is not explicit in the model. It's not obvious as to the changes that can be introduced by moving activities from one process participant to another, and the accompanying trade-offs that need to be considered. Similarly, enterprises rely on sense-and-respond loops to improve their operational processes. While the BPMN model in Fig. 3-5 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.

3.5.2 ArchiMate

In the second case, we discuss an enterprise architecture for our sample setting using the ArchiMate enterprise modeling language. Specifically, we present two viewpoints for discussion here that are closely associated with the conceptual modeling framework requirements presented in the previous section. In ArchiMate, a viewpoint contains a relevant set of ArchiMate notations that presents a particular perspective of the enterprise architecture [44]. By allowing such views of the enterprise, enterprise architects can focus just on the perspectives that are of interest to them, rather than being mandated to study and design the enterprise in its entire collective.

The first viewpoint that we present is the *layered* viewpoint, as shown in Fig. 3-6. The layered viewpoint contains several enterprise architectural layers in a single diagram. In the figure below, we shown three layers, the **Business Layer**, the **Application Layer**, and the **Technology Layer**. In the layered viewpoint, each layer exposes certain services which then service the next layer. This structure is indicated using two relationships, the *realizing* relationship and the *serving* relationship. Example, in the figure below, we see two services being present in the **Technology Layer**, the **Database Management** service and the **Application Hosting** service. Both of these services collectively help *serve* the **Enterprise Banking Application** software component. This component in turn *realizes* the **Application Service** that is then made available to the **Process Service Request** process that exists in the **Business Layer**. Thus, through such a layered structure principle, we have one layer exposing a set of services which are realized by the elements in the same layer. These services exposed in this layer are then utilized for serving the layer above. The

layered viewpoint shown in Fig. 3-6 also shows aspects that can appear in the **Application Usage** viewpoint. In this viewpoint, we shown how the application are used to support business processes that exist in the **Business Layer**. In our case, we show the that **Enterprise Banking Application** is used to support the **Process Request** activity that is part of the **Process Service Request** business process.

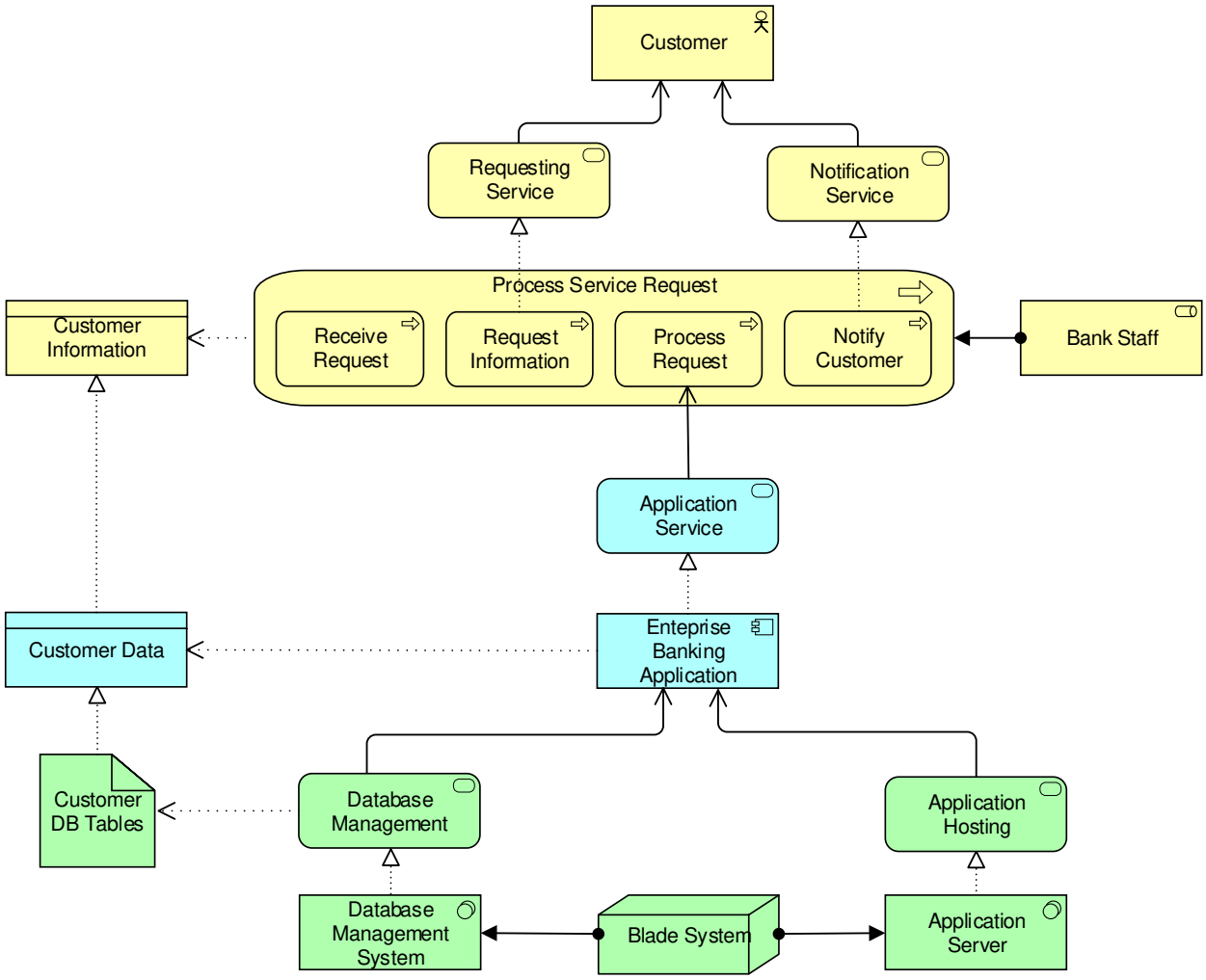

Fig. 3-6: ArchiMate layered viewpoint for the banking example

The second viewpoint presented in Fig. 3-7 is the *business process cooperative* viewpoint. This viewpoint shows the relationships between various processes that exist in the same layer. Through such a diagram, the enterprise architect can study the design and dependencies of various processes.

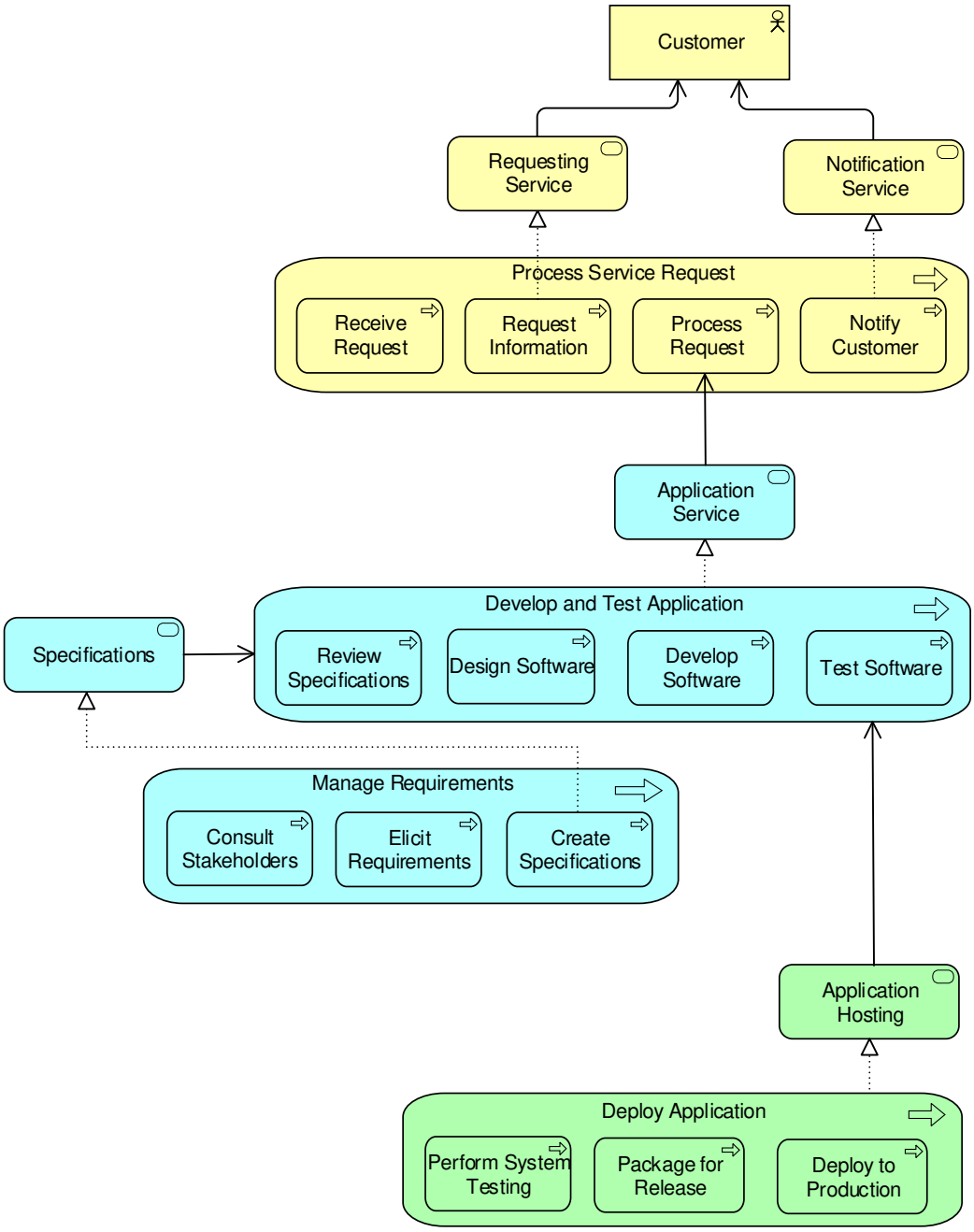

Fig. 3-7: ArchiMate business process collaboration viewpoint for the banking example

In the figure above, we highlight the various processes that exist across each of the three layers. Two processes are shown in the **Application Layer**, **Manage Requirements** and **Develop and Test Application**, with the latter process depending on the output of the former process for its execution. The realizes and serves relationships are used to associate both these processes, with the direction

of arrow indicating the dependency relationship between both processes. Here the **Specification** service is realized through the execution of the **Manage Requirements** process. This service in turn is used to serve the **Develop and Test Application** process.

While we just present two viewpoints above, there are many other viewpoints that are part of the ArchiMate specifications. Two other viewpoints that have some relevancy to the framework requirements are the *Goal Realization* viewpoint and the *Migration* viewpoint. The goal realization viewpoint allows an enterprise architect to analyze how certain high-level enterprise goals can be attained by refining them into progressively more tangible goals. These sub-goals are then refined into requirements or constraints that need to be considered when coming up with the enterprise architecture design. The migration viewpoint allows the contemplating of transiting from an as-is enterprise architecture to a to-be enterprise architecture by providing a high-level model of the activities that need to be performed using ArchiMate notational elements specific to the migration viewpoint.

Despite the presence of a wide range of modeling elements and several viewpoints, ArchiMate still does not fully meet the requirements for the conceptual modeling framework that we presented in the previous section. Specifically,

- The relationships between multiple processes can be represented in the business process cooperation viewpoint. However, it is not entirely evident how activities can be moved from different processes within the same layer, or even across layers, and the trade-offs that are to be considered. There could be situations where moving certain activities from one layer to another layer would better help server enterprise goals, but it's not apparent how activities can be moved across layers, and the surrounding enterprise architecture changes that would need to happen to accommodate such design changes.
- The types of conjunctive relationships between the processes are also limited, with just the realizes and serves relationships being used to depict the dependencies between multiple processes. This visual notation cannot capture or differentiate between the different types of relationships, such as that where one process is building a design that will be used by another process, or where one process is responsible for providing a plan that is to be executed by

another process. The full complexity of how several processes work together to attain some common objective is not evident.

- In the presented viewpoints, the lower layer realizes the services that are then used to serve services to the higher layer. Often, in an enterprise architecture, there would be cases where the higher layer also influences the behaviour of the lower layers. For example, planning activities are usually done at a higher layer (business layer) which are then used to influence the design and execution of processes or systems in lower layers. Through *flow* relationships, transfer (of information) can be represented between different elements, however, there is no way of modeling sense-and-effect relationships between different parts of the enterprise.

3.6 Conclusion

This chapter provides a set of requirements that guides the development of an enterprise modeling framework. Enterprise architects can use such a framework as part of their arsenal to represent and understand changes to the enterprise that are being introduced and supported by software innovations and emerging digital technologies. We use recent trends, such as digital transformation, bimodal organizations, and adaptive enterprises, to highlight some of the challenges that such organizations face when adopting software-based technologies for change and transformation. In order to determine these requirements, we first performed a systematic literature review to identify and review academic literature published in this decade and isolated eight characteristics that are common to software-enabled enterprises undergoing change. These characteristics were then abstracted out as a set of requirements for the enterprise modeling framework. We introduce this modeling framework in the next chapter.

4 The hiBPM Framework in Action

Acknowledgement: This chapter is partially based on the following paper;

- *Babar, Z., Lapouchnian, A., Yu, E.: Modeling DevOps Deployment Choices Using Process Architecture Design Dimensions. In The Practice of Enterprise Modeling (PoEM), pp. 322-337, Springer Publishing (2015)*

In Chapter 1, we introduced an example of a bank that is undergoing change in response to a shifting environment, competitive threats, and evolving customer preferences. We continue our discussion and analysis of this example by applying several concepts from the hiBPM framework for illustrative purposes. Specifically, we will consider three scenarios from our banking example. These scenarios were selected to illustrate the capabilities of the hiBPM framework to model and analyze multiple process architecture design alternatives, particularly under uncertain conditions caused by inadequate information needed for designing the process architecture.

In the first scenario, we provide a structure for the hiBPM model as it applies to a particular business process. Here a primary, yet simple, business process is selected and we review the design of this business process. In the second scenario, we consider innovations introduced to software processes that presently exist in the bank. These software processes are responsible for the development and maintenance of enterprise applications being used as part of business process execution. In the third scenario, we consider the need for re-architecting the enterprise to have two distinct areas, as is the case with bimodal organizations. One side contains the traditional processes and systems that are somewhat removed from the bank customer whereas the other side is closer to the bank customer and supports rapid cycles of change based on customer preferences.

In the following sections, we illustrate how the hiBPM model notations are used in three scenarios without going into details of the hiBPM framework constructs. The hiBPM framework constructs and methods are explained in detail given in Chapter 5 whereas in Chapter 6 we explain how to analyze change along multiple dimensions using these hiBPM constructs.

4.1 The As-Is hiBPM Model

A bank exists to provide banking services to its customers through its various channels. Traditionally this was accomplished through a bank branch but the advancement of technology has also enabled new alternate delivery channels, such as ATMs, PoS machines, Mobile Apps, and Internet Banking. Let us consider the case of a customer who visits a bank branch to get access to various financial services. Some customer service requests could be simple and quickly executed by the bank teller, like making a cash withdrawal, or submitting a cheque. Others may require the customer to spend more time working with a financial services advisor before the requested service is fully processed. These services could entail the customer applying for a mortgage loan or discussing how to set up an investment portfolio. We abstract away from the specifics of the customer request, and generally consider the case of a customer making a service request at a bank branch, with the assistance of a bank representative, like a bank teller or a financial services advisor.

Here we assume a typical bank business process that needs to have flexibility ingrained in its design to be able to respond to changing environments and technology innovation. We show a simple hiBPM model in Fig. 4-1 that visualizes the key aspects of the business process, software development for applications, and operational support processes that are commonly present, and contains multiple hiBPM constructs, such as process elements, process stages, process phases and the relationships among them. An explanation of how this hiBPM model was derived, and the hiBPM model notations used, is provided in subsequent sections of this chapter where we highlight some notable aspects of the hiBPM model while introducing different hiBPM constructs as we study and understand the domain example

The hiBPM model in Fig. 4-1 contains multiple processes, with the model itself emphasizing the relationships between these processes. The primary business process is where the customer service request is processed. Other processes support the execution of this business process. These supporting processes may include processes responsible for building software artifacts, processes where these artifacts are used, processes responsible for strategic planning, and other processes

where these plans are then used for operations. These related processes all come together as a process architecture.

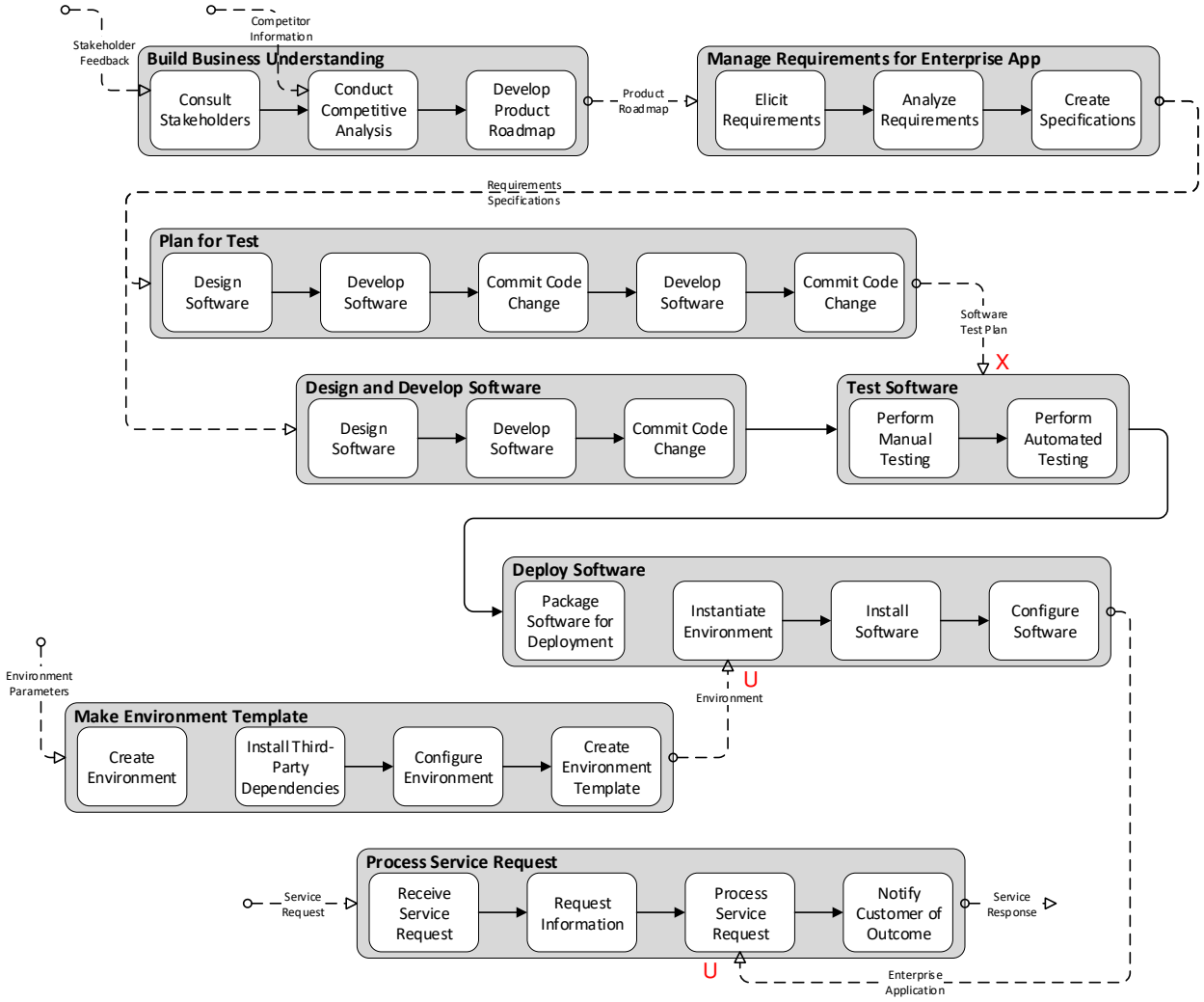

Fig. 4-1: An As-Is hiBPM model for the banking domain example

In the following sections of this chapter, we talk through the methods used to arrive at a reconfigured hiBPM model that is more optimally redesigned for the three scenarios that we introduced at the start of this chapter.

4.2 Scenario 1: Analyzing a Multitude of Business Processes

The enterprise architect needs to design the business process to be a simple, yet efficient, sequential execution of activities, along with the transfer of information amongst multiple process participants, e.g. the customer, branch staff, and other banking personnel, through the use of various enterprise applications. However, it is not enough to focus on optimizing the primary business process as, in order to make this business process possible, there may be other supporting and surrounding processes that need to exist. These may include software processes which develop the enterprise software that is used by bank staff during business process execution, or other business processes that support the primary business process. Thus, when designing the business process, enterprise architects and process architects need to collectively consider the interrelated processes as part of the analysis. This enables an assessment of the kinds of changes that are to be made to support the transformative characteristics presented in Chapter 3 and the corresponding effect that it would have on other processes.

While its useful to create conceptual models of business processes by considering the activities that are being performed by these processes, and the flow of information between them, we need to first discuss the purpose of these various activities. How can the business processes and the software processes be collectively analyzed to ensure their optimum design? Can these processes be reorganized and rearranged in a manner to improve the overall enterprise objective(s) accomplishment, particularly as the enterprise transitions to a digital business model? Can shifts in the external environment be used to influence the design and execution of these processes?

To explain how this can be achieved, let us start with serving the customer objective. Customers can be served by receiving them in the bank branch and processing their **Service Requests**. For processing the request, there must be specific enterprise applications in place which are going to be used as part of the processing of the request. Further, the bank staff need to be being trained on those enterprise applications. We review goal models intending to explore and depict how the attainment of a certain kind of softgoal would be achieved. Goal models are an existing way of modeling the means to attaining high-level stakeholder goals [171] and we discuss them in more detail, including their use in the hiBPM framework, in Chapter 6.

In Fig. 4-2, we show a goal model where the **Serve Customer** goal is attained by the satisfaction of two sub-goals, i.e. **Receive Customer in Branch** and **Service Customer Request**. **Service Customer Request** is decomposed into two further sub-goals, **Be Trained on Application** and **Setup Enterprise Application**. The **AND** notation in the means-end relationships indicates that both sub-goals are to be satisfied before the parent goal can be satisfied.

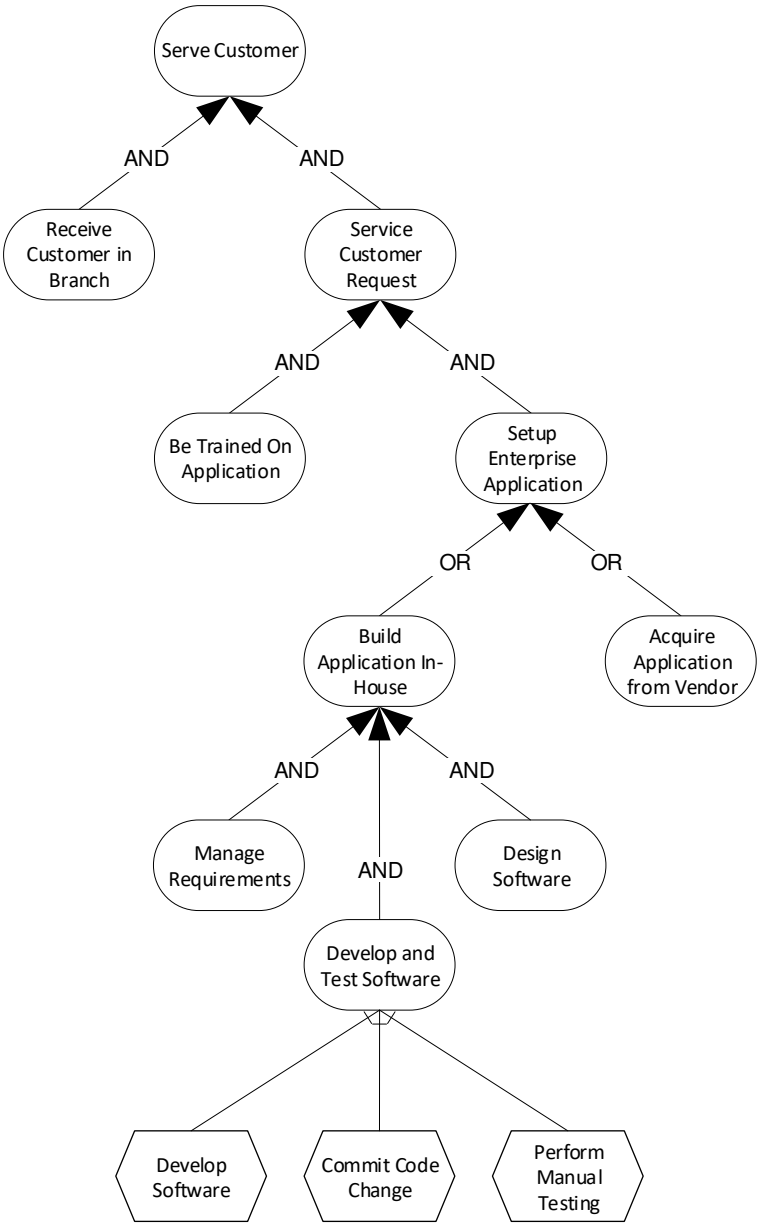

Fig. 4-2: Goal model for attaining bank operation objectives

As part of the hiBPM framework, we use goal models to (a) see how goal models can also be used to help determine the structure of the process architecture, and (b) analyze and guide possible configurations of the process architecture to help satisfy both functional and non-functional goals. The former is attained through constructing and navigating the goal graphs and seeing how a goal structure can be applied to an appropriate configuration of the process architecture model where the promise of a software-enabled enterprise (as it attains the functional and non-functional goals) is possible. For the latter, we consider different points in the hiBPM model where there could exist alternative process architecture configurations; here, the goal models would help decide between alternatives based on non-functional goals.

Goal models are used as a guide for determining additional process structures and alternative process architecture configurations that may need to be present in the hiBPM model. This does not imply a one-to-one mapping between the two modeling approaches, as goal models and hiBPM models can be at different levels of granularity and detail.

4.2.1 Determining Processes for Goal Attainment

Once we have a good understanding of the goals that need to be attained, we can determine the primary tasks that help attain that goal. Continuing this further, we are able to question how these tasks come together. Would they need to be executed collectively in order to attain the goal? Or can they be contributing to goal attainment, but not directly responsible for it?

Let us reconsider the **Setup Enterprise Application** goal shown previously in Fig. 4-2. By starting from this goal, we develop an idea of how to structure activities to ensure the attainment of this goal. There may be multiple ways of configuring the overall collection of business processes to ensure that the non-functional objectives for our banking domain are met, along with the functional objective. By delving further, we understand that there are two ways to attain this goal, as we show in Fig. 4-3. Alternative A (**Build Application In-House**) shown is for the situation at an enterprise that has achieved its functional requirements by building the enterprise application in-house using its staff, whereas Alternative B (**Acquire Application From Vendor**) is for another situation where the enterprise is acquiring the enterprise application from a vendor, and then deploying it. The softgoals for this particular situation are **Speed** and **Cost**, and we evaluate both goals options

against these softgoals. Here contributions links are shown as either contributing to the softgoal satisficing (**some +**) or not contributing to its satisficing (**some -**).

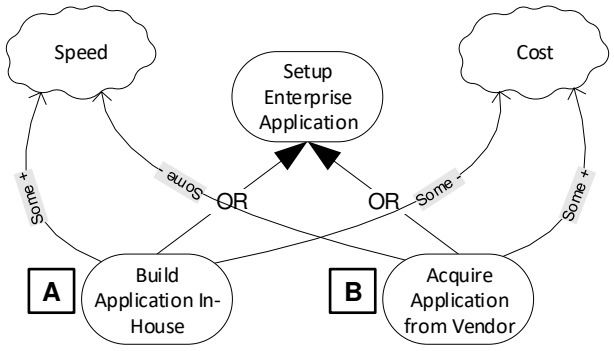

Fig. 4-3: Alternatives for attaining the Setup Enterprise Application goal.

Focusing on the goals and softgoals in such a way allows for contemplation without being hampered by the current design of the business process. We use techniques for goal satisfaction analysis to qualitatively assess if the softgoals can be satisficed [172]. Once goals and softgoals are determined for the banking example, we then explore and depict how the attainment of certain kinds of goals would be achieved. Through the goal model, we determine the corresponding, and similarly named, process stages in the hiBPM model that indicate how the tasks (from the goal model) are attained. Here, the focus is on goals, and these are used to show the associated process stages from a hiBPM perspective.

Process Stages (PS) are collections of activities that are to be executed collectively as part of the same execution cycle. Process stages are generally structured in a manner where they deliver some enterprise functionality, in the form of functional objectives or non-functional objectives (goals and sub-goals in goal modeling respectively). Process stages provide a generic representation of the structure of activities for the domain under study. These constructs represent various strategic, tactical and operational activities, at different granularities and different levels to each other. They are meant to attain some business or technological purpose and thus provide insight on "why" the processes are structured the way they are, rather than just providing insight into "how" specific business objectives are meant through process execution. We show examples of two process stages in Fig. 4-4, these are **Build Application** and **Acquire Application**.

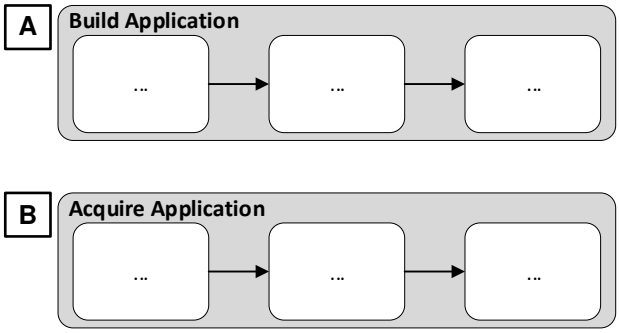

Fig. 4-4: hiBPM process stages as determined from goal model

Based on the goal model granularity and detail, the goals and softgoals are mapped to process stages, with the expectation that the goal models are sufficiently developed to indicate how the goals can be traced to process stages. We create process stages by reviewing the activities that help attain the goal or softgoal and determining if they would be part of the same recurrence, i.e., if they need to be executed collectively to attain that goal. If so, the process stage is thus determined. As the emphasis of the hiBPM is on architectural relationships between different types of processes, the internals of process stages is usually only defined at a level where they show sufficient detail on how the functional requirements are attained, including the activities required to produce the necessary output for those process stages.

Note, the process stages are responsible for functional or non-functional objectives' attainment; however with the goal model, we aim to show how they can be better configured also to satisfy the banking domain requirements. Thus, the configuration of the hiBPM model would be justified through the goal model, with specific configurations (mapped to alternatives in goal models) better suited to attain enterprise goals **Build Application In-House** and **Acquire Application from Vendor**, which correspond to Alternative A and Alternative B options for attaining the goals shown in Fig. 4-3.

4.2.2 Depicting Relationships between Processes

Once the process stages are determined, we need to start bringing some structure to the overall hiBPM model. For this, let us review the goal **Build Application In-House** from Alternative A that was introduced in the previous section. We can refine this further and come up with additional

sub-goals that are a means to accomplishing this. These sub-goals are **Manage Requirements**, **Design Software**, and **Develop and Test Software**. Conjunctive sub-goals from the goal graphs are separately operationalized in the hiBPM model, meaning that the goals have separate process stages, but they work together to attain some goal. AND de-compositions in the goal model from Fig. 4-5 are reflected in the hiBPM models where the AND relationship means that two process stages need to work together to collectively attain the upper-level softgoal.

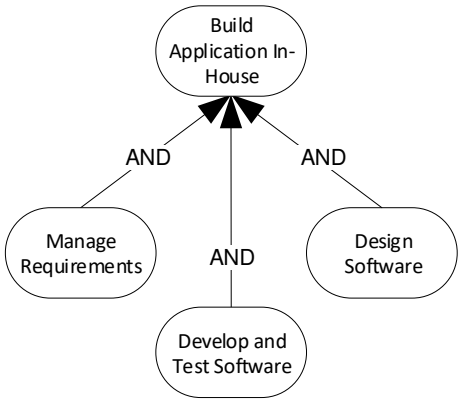

Fig. 4-5: Sub-goals contributing towards goal attainment

Using this goal model, we come up with corresponding process stages for the hiBPM model, as shown in Fig. 4-6. These map to the similarly named process stages of **Manage Requirements**, **Design Software**, and **Develop and Test Software**. Thus, there exist relationships between the three process stages (**Manage Requirements**, **Design Software**, and **Develop and Test Software)**, where the output of one is the input of the other. Here, the **Manage Requirements** process stage is shown to have a relationship with the **Design Software** process stage. Similarly, the output of the **Design Software** is **Software Design** that is used by the **Develop and Test Software** process stage. It should be clarified that the goal graph is not meant to be a precise reflection of the multi-level process stage relationship in process architecture models. Rather, the goal model interdependency structure is used as a guide to understand the relationships between different process stages.

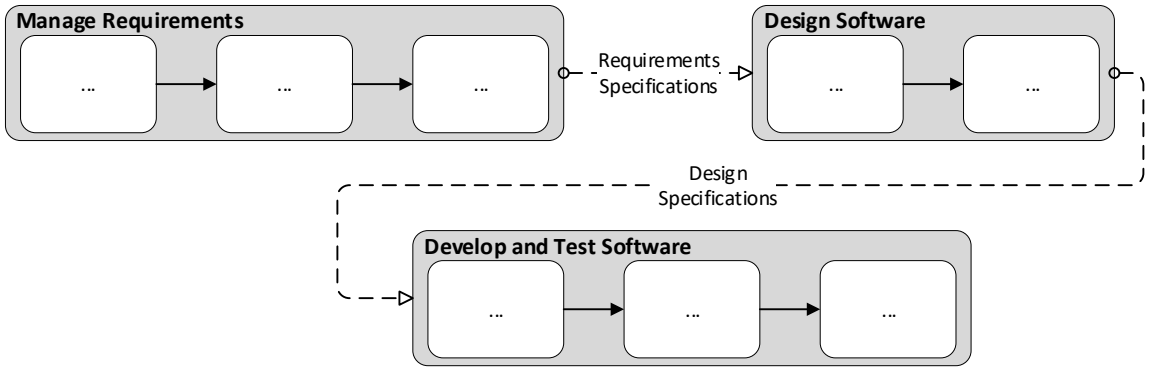

Fig. 4-6: Relationships across multiple process stages as determined from the goal model

By drawing these input and output relationships between the process stages, we can slowly form a structure for the hiBPM model. A particular relationship type is that of the data flow. *Data flows* can contain an exchange of data or artifacts that are needed by downstream process stages. The identification of these resources is done using goals models, which are then used to show the inputs to various process stages, further defining the relationships between the process stages. This concept is discussed further in subsequent chapters.

In any hiBPM model, there would exist areas with different behaviours and responsibilities. There may be specific processes that are responsible for planning vs. processes which are more operational. Certain processes would be designing and building tools or capabilities, which are then going to be used in other locations within the hiBPM model. There need to be constructs in the hiBPM notation that shows the relationships between these different areas, while also helping with the differentiation between different types of processes. Showing these is important as in hiBPM; these relationships connect "higher" level process stages to "lower" level process stages. Such relationships can exist between process stages, from process stages to process elements, and may even be within a process stage. These relationships provide associations between various process stages and process elements, and help answer some broader questions, such as what would be the sequence of execution of these tasks? Or can we change the sequence of process execution to introduce different behaviour? These relationships (and the differentiation of process stages into higher-level or lower-level process stages) are discussed in more detail in Chapter 5.

4.2.3 Activities for Alternative Goal Attainment

Often, process stages are not sufficiently granular to guide how to perform different tasks across the business process. For example, we may want to know the specific sequential order of executed tasks to accomplish a goal associated with a process stage. Having this knowledge allows us to consider alternative means of accomplishing that goal, e.g. by performing certain activities before others. In such a case, there is a need to see the composite tasks for a process stage, for reasons of analysis and reconfiguration.

Take the example of **Develop and Test Software** as this process stage is essential to ensure the domain softgoals. Again, the goal model provides a useful starting point for determining the internal details for the hiBPM model. In the goal model of Fig. 4-7, we see that **Develop and Test Software** is decomposed into several tasks, **Develop Software**, **Commit Code Changes**, and **Perform Manual Testing**. Depending on the granularity of the operationalized goals, these tasks (from the goal model) can be used to show similar activities in hiBPM. There is an AND relationship between these tasks, meaning that they come together to result in the accomplishment of the goal. Thus, when determining the process stage, we can show that this process stage comprises of several activities in the hiBPM, which help the process stage attain its goal. It is not mandatory to determine all these activities, as generally only those that help with analysis are beneficial and considered. For this, we consider process elements which are more fundamental process structures in the hiBPM modeling notation.

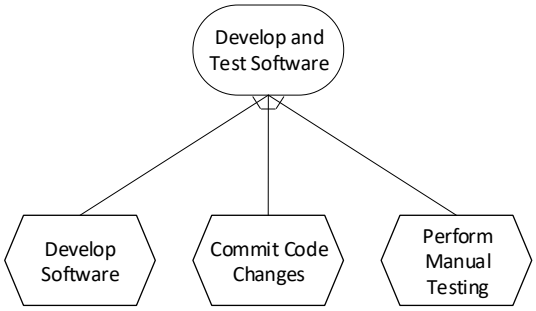

Fig. 4-7: Determining tasks from goals through decomposition

Process Elements (PEs) are basic activity units which produce some output or outcome based on a set of control and data inputs. They may also include the act of decision making as part of the

activity processing (i.e., what action to perform based on the data available). The specifics of how these actions are performed are not required in hiBPM as the focus is on how the process element contributes to the attainment of enterprise functions. Through envisioning process elements in hiBPM, we can ask questions such as whether specific tasks should be done before other tasks, or after? Whether sufficient information is available to make decisions? Can we remove tasks from the critical path in order to hasten the time required to release a product?

In hiBPM, the process elements are shown as activities within a process stage, which when executed on being provided with some input, produce an output that is then used by subsequent process elements or downstream process stages. This can be seen in Fig 4-8 as the **Develop and Test Software** process stage and the three internal process elements (**Develop Software**, **Code Changes**, and **Perform Manual Testing**) are being executed in sequence to produce the output. For each process stage, the internal process phases (introduced later in this chapter) and process elements need to be determined to understand better how the process stage can attain the functional objectives that it is responsible for, while supporting non-functional objective attainments. Process elements are grouped in process stages if they are executed together as part of the same execution cycle, and to attain a common functional or non-functional objective.

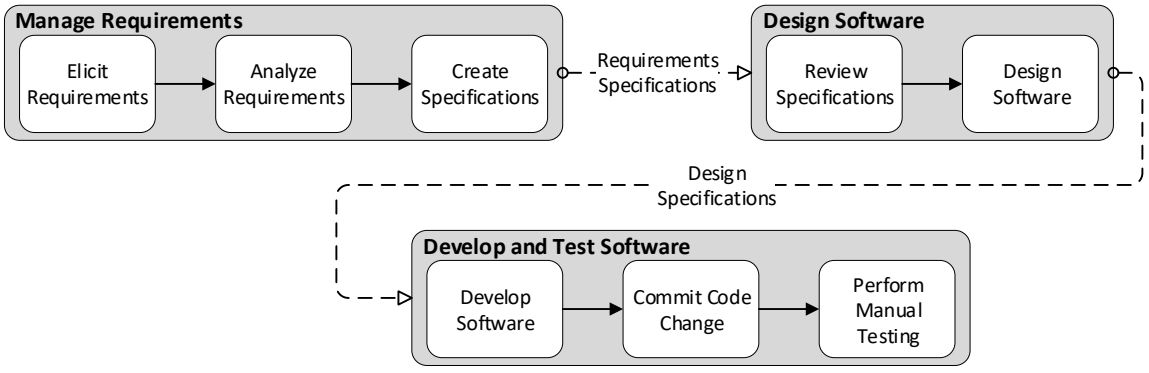

Fig. 4-8: Identifying hiBPM process elements for previously determined process stages

4.3 Scenario 2: Introducing Innovation in Software Processes

A challenge for enterprise architects is to redesign portions of the process architecture to incorporate innovations and improvements to the software development processes better, notably

to support digital technologies, such as cloud computing. The software processes would have to be periodically redesigned based on evolving situational needs, such as the need for faster delivery of new product features and bug fixes through higher deployment frequency to application production environments.

The enterprise architect designing the architecture for any such organization would need to consider the following,

- Automating activities in the overall software development process through introducing software tools and custom development of scripts, thus shortening the time required for new feature development and bug fixes through the reduction of manual effort. Such a design for automation enables software teams to deliver frequent releases to customers and users.
- Using feedback loops for continuously improving software development processes. Product feature development can be improved for speed and quality through 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 collaborate towards the attainment of enterprise objectives.

We consider DevOps to illustrate such a recent innovative practice in software development and maintenance. DevOps is a software development approach that enables enterprises to deliver software product features through process automation rapidly while improving inter-team collaboration and increasing operational efficiency through monitoring and measuring activities [33][34][35]. The term “DevOps” is a combination of two words “Development” and “Operations” and has been described as a methodology for rapidly and frequently delivering new software product features and service innovation 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.

The hiBPM model needs to be understood through a combination of these ideas and concepts, particularly in light of enterprise requirements for greater responsiveness and adaptability while managing uncertainty. No two enterprise-adopted DevOps approaches would be similar as each bank has unique characteristics and requirements.

4.3.1 Temporal Execution of Activities

The development of product features is done in DevOps 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 [89]. 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 variations in process architecture configurations.

Let us consider the case where a developed feature is to be functionally tested before it goes through the continuous deployment process. This testing can be carried out by Quality Assurance (QA) engineers in at least two ways: they can collaborate with the software engineer to quickly validate the functionality before the codebase is committed to the code repository, or they can retrieve the committed code from the code repository and test it on a test environment. As shown by the goal model in Fig. 4-9, the former approach has the benefit of being collaborative and encouraging both the software engineer and QA engineer to work together to solve the problem quickly. The latter approach is more methodological and allows for the proper (and independent) validation of the feature and the tracking of testing issues. There is uncertainty in the design of the testing process, with different enterprises (or the same enterprise under different conditions) selecting different ways of configuring their testing processes. Thus, there has be an element of design uncertainty incorporated where the final design can be determined, or a modified design selected, at some future point in time.

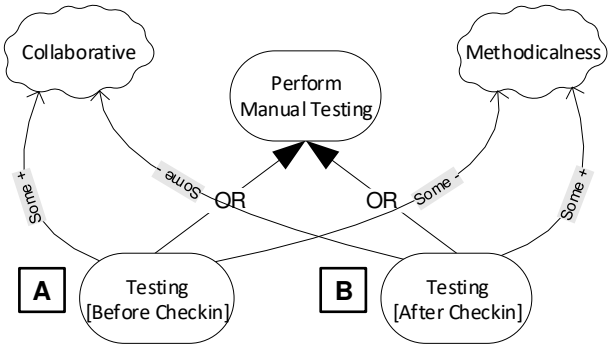

Fig. 4-9: Determining process elements for testing alternatives

In hiBPM, we emphasize the relative positioning of the process elements within the overall model. Repositioning a process element within a hiBPM model results in variable behaviour to support change objectives, as there may be multiple possible temporal placements for structural elements that achieve the same functional objective but are different in terms of their non-functional characteristics. When ordering these structural elements, we need to be mindful of the functional dependencies among them. Despite this, through temporal displacements of these structural elements can still result in functional goal satisfaction but differ in terms of their non-functional characteristics.

We can introduce reconfigurations in the hiBPM model by virtue of these temporal movements. E.g. a process element could be moved earlier or later in relation to other process elements while being within the same process stage. The output of the process stage would not change; however, how the process stage is executed would change. Considering the possible ways in which to reconfigure the hiBPM model along the temporal dimension, the placement of any process element should be carefully considered with regards to various softgoals, subject to inherent temporal constraints among the process elements. The appropriate order of the **Perform Manual Testing** process element is determined based on the organization's prioritization between the softgoals. We then select either one of the two alternatives shown in Fig. 4-10, i.e., the QA engineer verifies the developed feature (**Perform Manual Testing**) after the software engineer checks in the code to the code repository (**Commit Code Changes**) or before the code is checked in.

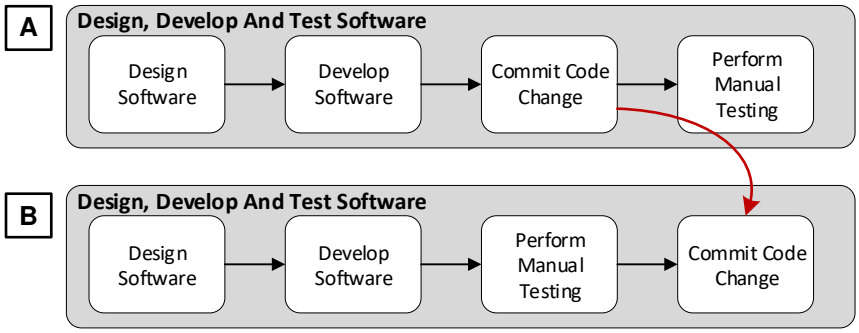

Fig. 4-10: Moving process elements across temporal dimensions

The particular temporal placement of a process element can bring about certain benefits. A process element can either be advanced (and be executed) before other process elements or postponed after those process elements. *Postponing* a process element provides the benefit of executing it with the latest context and information available, thus reducing the risk and uncertainty that are inherent in any software process. *Advancing* a process element relative to other process elements reduces complexity and cost, as less effort is required to process the limited contextual information available at that instant. The testing of a product feature by a QA engineer illustrates the trade-offs between advancing and postponing a process element.

In Fig. 4-11, we assume a case where changing the sequential execution order of the **Commit Code Changes** and **Perform Manual Testing** process elements does not result in any change in softgoal satisfaction. In such a case, there is no reason to show them as having a certain temporal order, and we can have a process phase encapsulate them. *Process phases* are sections within a process stage that produce the same result irrespective of the arrangement of process elements within, i.e., the temporal reordering of process elements does not result in any change in the outcome of the process phase. A process stage may contain one or more process phases. An output of a process phase can only be used by the subsequent process phases of the same instance of process stage.

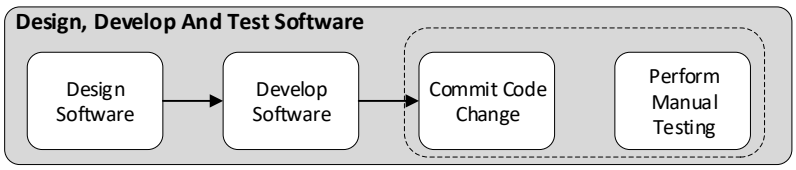

Fig. 4-11: Determining process phases in a process stage

4.3.2 Designing for Reusability or Customizability

When designing DevOps to support automation, there has to be an element of repeatability of process execution; this can be attained by having software tools, capabilities or design artifacts that are available for repeated use. Such artifacts are either going to be built or used during the execution of a software process. These artifacts would be shown as either input to the process stage (if the artifact is used during processing) or as outputs (if the artifact is produced by the process stage). As long as there is an understanding of how to use this artifact, it is not necessary to know *how* this design artifact is built before it can be used by a user performing an activity.

In Fig. 4-12, we show where a design **Environment Template** is built by **Make Environment Template** process stage that is subsequently used by another process stage **Deploy Software** in the hiBPM model. Here greater automation of the software development lifecycle is attained by using environment templates. These templates are pre-built and configured for use in any DevOps implementation.

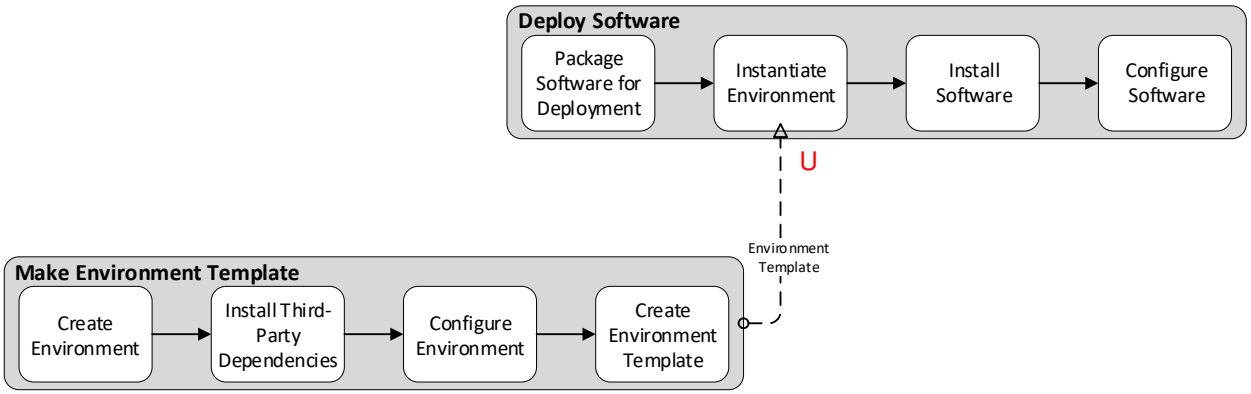

Fig. 4-12: Design-use relationship between two process stages for deploying software

Having such a concept allows imagining places in the hiBPM model where such *designs* are being built, and are used elsewhere. This way, one can see what activities can be automated, and the type of designs that would need to be built to support that automation effort. Being able to repeatedly use the design also enables automation of process execution which helps in reducing the time and cost of process execution; thus, repeated reuse of design through automation is an essential factor for attaining these softgoals.

In hiBPM, the process stage producing the design is called the design stage and the process stage using the design is called the use stage; both these stages come together in a *Design-Use* relationship. Through this design-use relationship between the two process stages, we can show the location at which the template can be integrated into the software process; this is useful as it allows ensuring that suitable process and data dependencies are fulfilled at or before that point. The introduction of any artifact in the design-use relationship should be evaluated against the softgoals. In Fig. 4-13, we show two alternatives for **Deploy Software**. Alternative A has a design-use relationship while in Alternative B no designs are built and the entire software deployment is done by always creating an environment from scratch and then deploying the software.

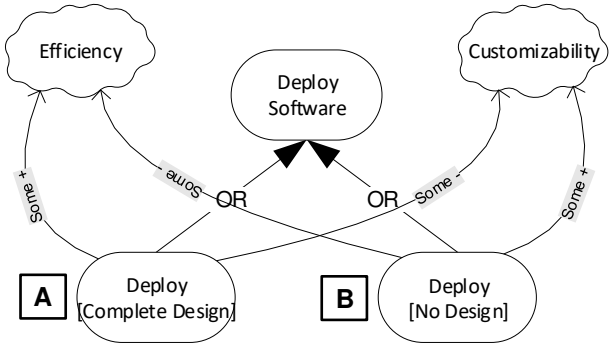

Fig. 4-13: Goal model showing alternatives for complete designs versus no designs

4.3.3 Planning Ahead, or Deferring Planning

When executing a process, there will be a need to have modifiable behaviour of process execution, as and when the situation demands. This is particularly true when situation context changes and the software process needs to behave differently. For example, there may be a need to perform additional activities to ensure a quality release, which goes beyond the normal level of testing. Thus, we should have a means for deciding on how to change the way the process is executed; this is done through *plans*. Through plans, we can induce some change in the hiBPM design or result in some behavioural change in the business process execution.

Let us consider another example, this time around the testing of the software, with the hiBPM model shown in Fig. 4-14. Typically, **Software Test Plans** are used as guides to during the execution of the **Test Software**, process stage. **Software Test Plans** are prepared by the process

stage **Plan for Test** and enable automated testing of the developed features. Here, there exists a *Plan-Execute* relationship between the **Plan for Test** process stage and the **Test Software** process stage. A plan-execute relationship can be considered to have two distinct process stages, where one process stage, **Plan for Test**, is responsible for creating a plan, which the other process stage, **Test Software**, would then execute one or many times. We call the process stage producing the plan the planning process stage, and the process stage executing the plan the process execution stage. Both of these work in conjunction to achieve some upper-level business objective, which requires the conceptualization of both plan and execute process segments. A plan can be re-executed many times.

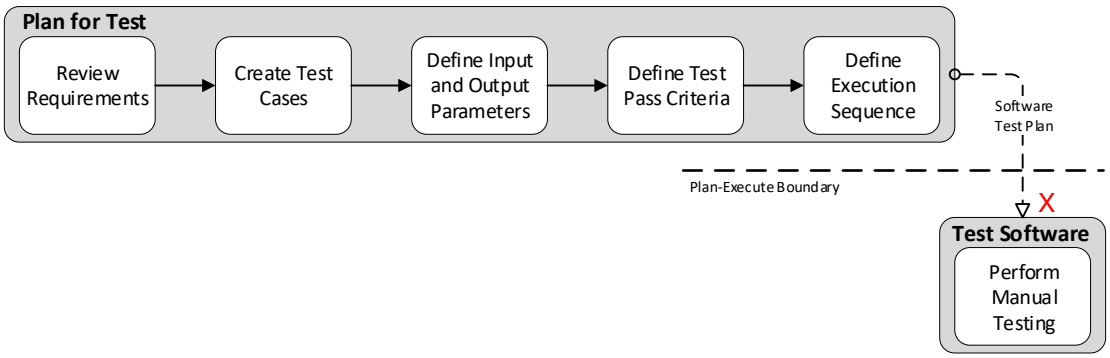

Fig. 4-14: A plan-execute relationship between two process stages for testing software

As shown in Fig. 4-15, there are two possibilities for testing the software. In Alternative A, we develop **Software Test Plans** that are detailed and complete and do not require any interpretation for their execution. The **Test Plan** consists of not just the overarching testing criteria but also the test execution sequence that is to be followed. In Alternative B, there is no prior test planning done and the testing is done in an ad hoc and unsystematic manner. The trade-offs between both the alternatives would be between the repeatability and flexibility of the automated testing being performed.

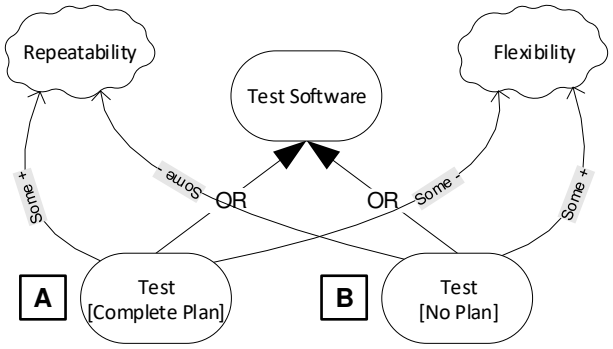

Fig. 4-15: Goal model showing alternatives for complete plans versus no plans

We present a comparison between the design-use and plan-execute relationships in Table 4. A notable difference between plans and designs is that the process stage using the design does not need to be aware of how the design is built whereas the process stage executing the plan needs to be aware of the information as codified in the plan in order to ensure proper execution.

Table 4. Comparison of design-use and plan-execute relationships

Design-Use	Plan-Execute
Design of system tools and artifacts that are to be used repeatedly and to enable process automation.	Construction of plans that are used as instructional guide(s) for process execution, or process architecture reconfiguration.
A design is a software tool or artifact that is used by downstream process stages to accomplish enterprise goals and softgoals.	A plan provides instructions for execution of activities to accomplish enterprise goals and softgoals.
Downstream process stages are not aware of the internals of the design and can directly use it.	Downstream process stages need to be aware of the instructions as codified in the plan in order to ensure proper execution.
The use of a design by the downstream process stage is denoted by a relationship (with a "U" annotation) terminating at the bottom of the process stage that uses the design.	The execution of the plan by the downstream process stage is denoted by a relationship (with an "X" annotation) terminating at the top of the process stage that executes the plan.

4.4 Scenario 3: Designing Two-Speed Enterprise Architecture

In Chapter 3, we presented the concept of two-speed or bimodal organizations where each "section" of the enterprise operates at a relative frequency to the other. Transforming a legacy enterprise architecture of the bank to one that supports the objectives of a two-speed organization

brings about interesting challenges for the enterprise architect. Here, the management of the customer-centric front-end systems is kept separate from the legacy back-end enterprise systems to allow for the independence of decision-making and operations. Each section is responsible for the attainment of distinct goals and softgoals; such a separation enables both sections to be optimally designed for the realization of these goals and softgoals. However, both sections still need to meet to fulfil broader enterprise goals and thus, there would always be some constraints on the design of processes and software systems; these would need to be considered during tradeoff design analysis. These constraints can pertain to the sociotechnical isolation, the exchange of data, or the difficulty in aligning process and software design across both sides.

Enterprises that adopt a two-speed enterprise architecture have done so by separately designing the software and business processes (and the software applications used within) for both areas, to satisfy the softgoals for each. E.g., the enterprise back-end side may use traditional phase-based (or waterfall) software development methodology to prioritize stability and reliability of the developed product, at the expense of rapid delivery of new features in the back-end enterprise application. On the customer-facing side, the bank may use Agile development practices [173] to deliver mobile and internet applications that are quickly updated with new features. However, this may come with some degree of disruption and risk to the customers, i.e., in case software bugs are introduced or customers dislike new feature rollouts. These issues can be reduced by incorporating feedback loops that are entirely contained within the customer-facing side.

Finally, in order to support such enterprise architectures, additional software processes (like blue/green deployments [174] and A/B testing [175]) may be introduced to minimize the potential impact of disruptions and risks to enterprise operations. To reconcile both sides, the bank may introduce some form or middleware (or an API gateway) that conforms to an agreed-to standard interface, which frees both sides to evolve and develop independently, with their processes, architecture, user experiences, and softgoal priorities [176].

4.4.1 Managing Relative Execution Frequencies

In an enterprise, there would be processes that execute at a higher frequency than other adjacent processes. This is important to understand when designing the bimodal enterprise, we need to

decide which business and software processes to place on either side. Let us illustrate this by considering the two applications being used in our present scenario, namely the enterprise application used by bank staff members to process customer service requests and the mobile app that bank customers use themselves to perform their financial transactions.

Developing either the enterprise application or the mobile app requires first understanding the features to be developed. This entails consulting stakeholders for eliciting requirements, shown in the **Manage Requirements** process stage. As enterprise applications do not frequently go through rapid cycles of development, the requirements that are determined for the enterprise application can be done once in **Manage Requirements** to prepare design specifications that are assumed to be fairly stable during the duration of the development cycle, as performed in the **Develop Enterprise Application** process stage. This is shown in Fig. 4-16 where the for every execution of the **Manage Requirements** process stage, the **Develop Enterprise Application** process stage executes once too.

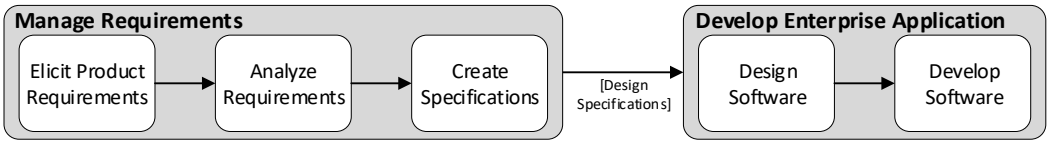

Fig. 4-16: Recurrence relationship for managing requirements and developing the enterprise application

However, in the case of mobile apps, the requirements would need to be periodically reviewed in and new development items (in the form of a groomed **Product Backlog**) be determined. Thus, the enterprise-side process stage of **Device Backlog** may be executed once to determine the requirements for the mobile app, but the fast-moving customer-side process stage of **Plan for Release** and **Perform Sprint Cycle** would be executed frequently as there may be changing priorities to the requirements provided. Continuously revising the order of these requirements (present in a **Product Backlog**) allows for the development teams to **Develop Feature** based on the organization's shifting priorities to meet softgoals such as customer-centricity or service adaptability. We show this in Fig. 4-17.

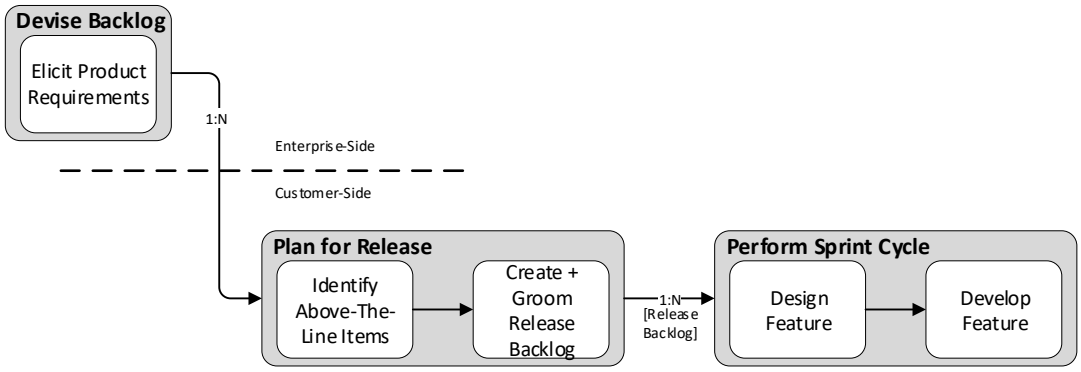

Fig. 4-17: Recurrence relationship across the bimodal process boundary for the mobile application

In the figure above, we do not show all the surrounding software processes, but it is implicit that the enterprise-side uses different software development practices (such as Waterfall) than the customer-side, which uses iterative and incremental development approaches (such as Scrum). Through *recurrence* relationships, we can see the location where both sides of the enterprise come together. Recurrence relationships allow the considering of what decisions and actions in the enterprise need to be re-executed, and under what conditions. Boundaries can exist between process stages; these need to be visually displayed to show the differing characteristics that can exist between adjacent process stages. This is necessary for analysis so that we can determine the impact of moving certain process elements or process stages across different portions of the hiBPM model. In hiBPM, *process boundaries* are means of identifying and demarcating similar process stages with respect to certain process-related attributes, such as the frequency of execution, the type of process output, and the behaviour of the process stage.

4.4.2 Dealing with Adaptation

We previously mentioned that the customer side of the enterprise needs to evolve and deliver new product features quickly. However, this brings up a situation where the pressure for faster development and deployment cycles may inadvertently introduce software bugs or undesirable product features; these need to be fixed immediately to ensure ongoing customer satisfaction. If the normal enterprise development feedback cycle is followed, the changes that need to be made would first be reviewed by the product team on the enterprise side, before they turn up as product backlog items on the customer side. This would not work as there has to be an immediate response

to any determined issues, and these need to be fixed in hours and days. There needs to be sense-and-response linkages within the enterprise to solve this problem; these exist between the sensing part of the enterprise and the responding parts of the enterprise. In the situation described above, these linkages are to be entirely contained within the customer side of the enterprise architecture.

Let us review this in our hiBPM model example in Fig. 4-18. The mobile app solution is running in a production environment, with the state of the environment being monitored for both user behaviour and system metrics in the **Monitor Environment** process stage. The various metrics are then sent to the **Review Environments Metrics** where these are evaluated. This is shown as a *sense* flow between the **Monitor Environment** and **Review Environment Metrics** process stages. The metrics may indicate that there is a change in user behaviour (for example new features introduced are less preferred than earlier versions) or that the recently developed code is resulting in performance degradation. The feedback from these validation cycles is used for further design-time improvements to the product. Through such an evaluation, additional development tasks are decided which are then passed on to the **Design, Develop and Deploy Software** process stage. After this, the normal structure of the hiBPM model follows. Thus, there exists a sense-and-respond path between the **Monitor Environment** and the **Design, Develop and Deploy Software** process stages, the purpose of which is to have ongoing improvements to the mobile app.

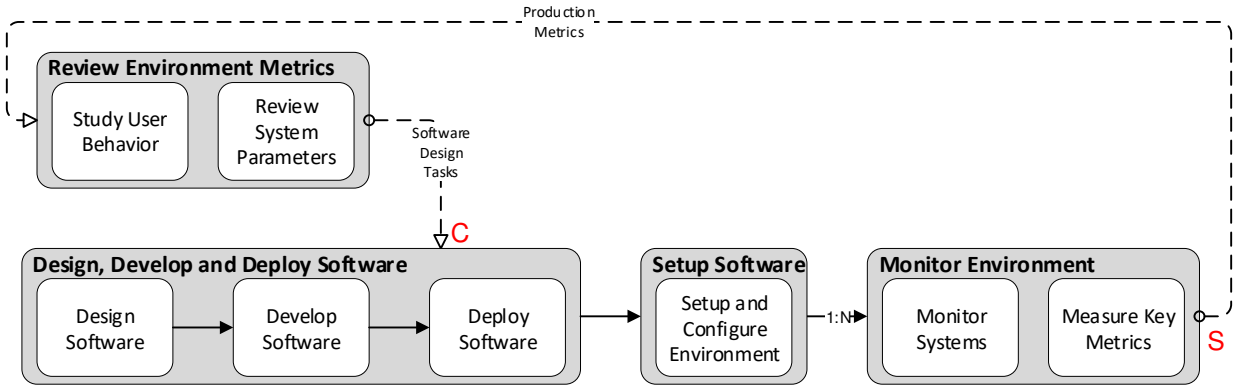

Fig. 4-18: Sense-and-control path for responding to production metrics

By differentiating control and design inputs from normal data inputs, and sensing from normal outputs, we can locate adaptive loops as they exist within a hiBPM model through the use of sense and control flows. Lower-order use process may execute many multitudes of times more frequently

than the high-order design process. Thus, the higher-order process (the P and D sides of the plan-execute and design-use relationships respectively) typically has a lower recurrence frequency than the lower order process (the X and U sides). Adaptation loops are discussed in mode detail in Chapter 5 and Chapter 6.

4.5 The To-Be hiBPM Model

At the beginning of this chapter, we presented an As-Is hiBPM model. While understanding the structure of this As-Is hiBPM, we identified and analyzed alternative configurations of the hiBPM model based on the three scenarios presented. The alternatives were analyzed against various non-functional objectives for the bank, in addition to the primary function objective, i.e., providing financial services to the customer. Using analysis methods provided by the hiBPM framework, we modified the design of this process architecture by moving various activities around, or the changing the relationships that connect multiple processes. Such changes in the process architecture led to alternative design configurations of the hiBPM model; these alternative design configurations still met the enterprise objectives but the non-functional objectives attainment may be different.

In Fig. 4-19 we show the To-Be hiBPM model that encompasses all three scenarios described at the beginning of the chapter. Such changes to the hiBPM model allow us to cope with the transformation characteristics of software-enabled enterprises described in Chapter 3. The hiBPM model further indicates two separate sections in the enterprise that exist to support situations where customers demand increased service responsiveness. Each section (corresponding to the customer-side or the enterprise-side) ensures that the processes within are suitably designed for increased customer-responsiveness or stability of operations.

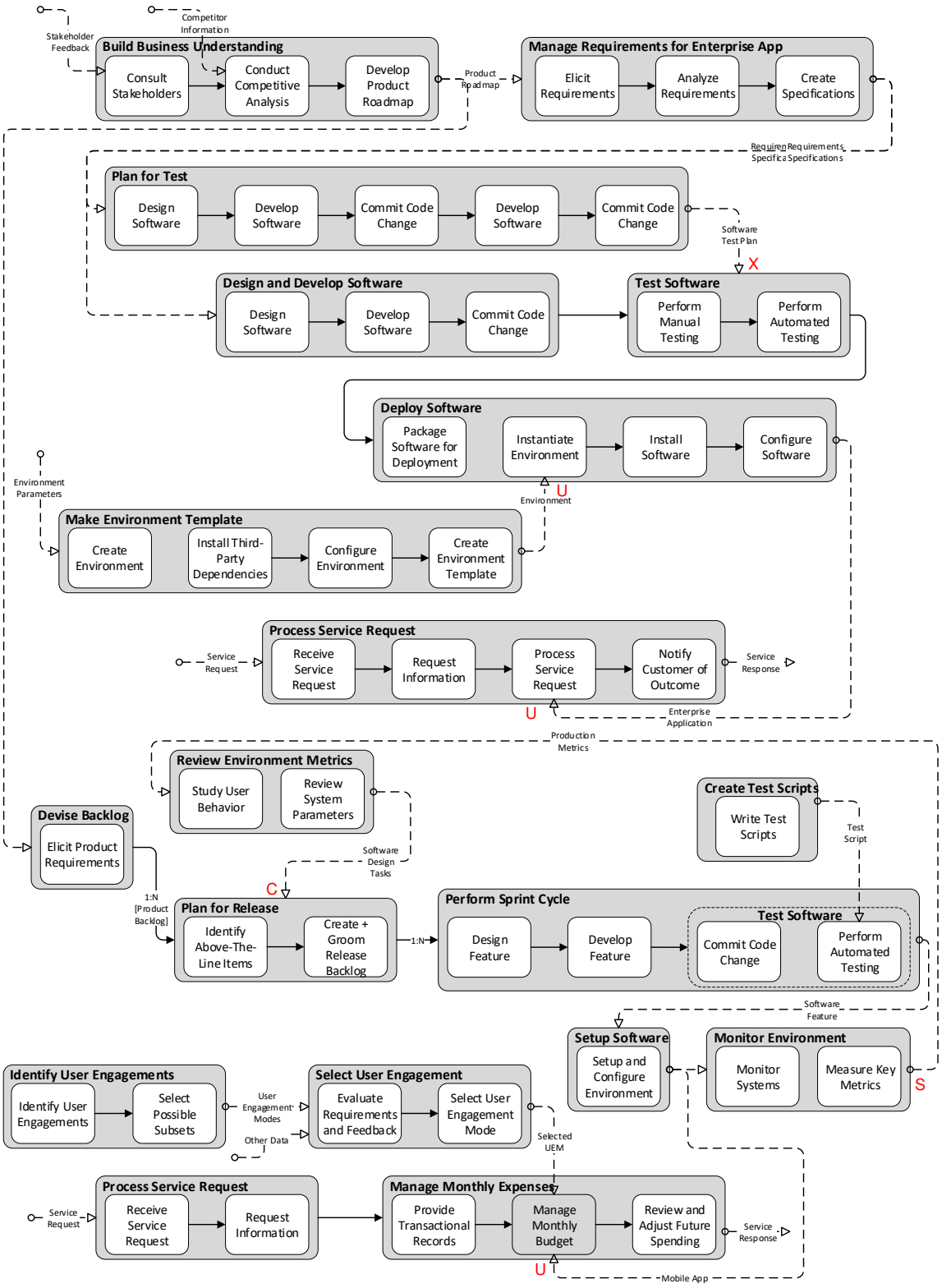

Fig. 4-19: The redesigned hiBPM model for the banking domain example

4.6 Conclusion

A major challenge for information systems design is dealing with enterprises that undergo change. It cannot be assumed that a single design cycle and implementation would be sufficient, as there are numerous uncertainties. Change in such a dynamic enterprise would be long-lasting, and impact numerous areas in the enterprise. Any change exercise in the enterprise would have varying degrees of success, as measured by their attainment of enterprise objectives, and the feedback from these would be used to refine subsequence change cycles iteratively. Thus, dynamic enterprises evolve in response to changing environmental circumstances and business objectives, frequently going through cycles of learning and improvement.

In this chapter, we demonstrated how the hiBPM framework can be used to systematically analyze multiple scenarios in a bank that is undergoing software-enabled change. Here we created a hiBPM model for the overall collection of business processes and used it to progressively reason about and introduce changes in banking processes to ensure that the enterprise continues to attain its organization objectives. In the next two chapters, we provide a detailed explanation of the hiBPM framework constructs and methods, including how to analyze change along multiple dimensions using these hiBPM concepts.

5 Creating Models using the hiBPM Framework

5.1 An Architecture of Processes

A *hiBPM model* is visually depicted as a conceptual model that considers various (process-related) constructs while providing the ability to express different design configurations of processes by describing activities, and their relationships needed to accomplish enterprise objectives. hiBPM emphasizes the existence of decision-making points and offers expressiveness to allow relevant process architectural properties to be analyzed, and for contrasting among alternative process architecture design options. Detailed process sequences or elements, such as that offered in business process modeling techniques, are not needed as they do not lend to the analysis of alternative ways to reconfigure the hiBPM model. The idea is to have expressiveness of the domain for allowing capturing and evaluation of alternative hiBPM configuration options, without having to detail all domain elements, their relationships, operational sequences, specifics on activity execution, or information flows.

Portions of hiBPM model could be redesigned based on evolving functional and non-functional enterprise objectives. Thus, many possible process architecture configurations (also referred to as a "design space") exists in the domain under study. Such a design space would include the possible configuration options that exist in different areas of the hiBPM model, and under the conditions in which those options would be valid. Thus, in hiBPM, we do not consider the process architecture to be static, but something that can (and should be) re-analyzed and re-engineered. The process architecture can be configured in multiple ways while simultaneously permitting trade-off analysis between several enterprise non-functional requirements. Alternative designs are, therefore, different ways of respectively modifying or implementing the process architecture. Such redesigns would allow a process architect to ask questions such as,

- Does the present configuration of the process architecture permit the attaining of enterprise non-functional objectives?
- Where in the hiBPM model can redesigns be introduced for enterprise flexibility and to attain specific non-functional objectives?

- Should certain activities be postponed for later to ensure that real-time data is used in the processing, or would the additional cost of this configuration outweigh the benefit of delayed execution?

Enterprise architects and process architects can navigate through the design space of a hiBPM model and come up with alternative designs of process architecture. This permits enterprises to be designed with flexibility to permit change, while taking into consideration criteria such as cost, performance, uniformity, etc. The hiBPM model is part of an overall hiBPM framework. Here, the term "framework" is emphasized to indicate that this design artifact consists of a set of prescriptive constructs and modeling notations that collectively allow analysis, reasoning and evaluation-related activities. Accompanying methods and rules allow for the prescribing and qualitative determination of process architecture configurations based on enterprise requirements while evaluating trade-offs between multiple configuration alternatives.

The hiBPM constructs, modeling notations and methods are presented in Chapter 5 and Chapter 6 of this thesis. Specifically, in Chapter 5 we discuss in detail the various ways that a hiBPM model can be design and configured, whereas in Chapter 6, we present the analysis capabilities to decide between design space options in the hiBPM model. We continue to use the domain example introduced in Chapter 4 to explain the concepts covered in Chapter 5 and Chapter 6.

The notion of business process architecture, including the types of relationships and various dimensions of change, was initially proposed in [26] and [27]. This research project is based on this preliminary work and extends it by adding new constructs and providing additional details and preciseness to the ones that were originally introduced to develop a complete conceptual modeling framework.

5.2 A MetaModel for hiBPM

We present a meta-model for the hiBPM model in this section. According to Seidewitz [177], "a meta-model makes statements about what can be expressed in the valid models of a certain modeling language". A meta-model for the hiBPM framework is needed as it helps with the analysis and construction of a hiBPM diagram that models a domain under study. Through this

meta-model, we define the various modeling notational constructs needed for creating a hiBPM model. This meta-model helps decide amongst possible modeling constructs and modeling variants as they apply to different situations and aids in navigating the space of possible hiBPM design options. This balances the need for creating manageable and straightforward hiBPM models against the inherent complexity that accompanies any attempt to model a multitude of business processes.

A simple metamodel for the hiBPM modeling framework was introduced in [26]. We present an updated meta-model with additional process constructs and updated relationships between the various meta-model elements in Fig. 5-1. For developing the hiBPM framework meta-model, we utilized concepts adapted from business process modeling [50], functional modeling [179], and goal modeling [171]. Such an approach permitted us to consider and assess optimum process architecture design alternatives while keeping in mind enterprise functions, business processes and their associated goals. The meta-model is structured as two primary parts; the first is for creating a hiBPM model, whereas the second is for analyzing alternatives using a goal model based notation.

The first part of the meta-model is further divided into two primary categories of constructs, **Structural Elements** and **Relational Elements**, the former being used to define the significant constructs that allow for the structuring of domain (process) activities with the latter being used to depict relationships between the structural elements. There can be different types of structural elements – **Process Stage**, **Process Phase**, and **Process Element**. Relationships exist between structural elements and provide a means of association and information transfer; the relationships include **Data Flow**, **Sequence Flow**, **Recurrence**, **Design-Use** and **Plan-Execute**. Through this meta-model, we can express the attributes of the modeling constructs, the mechanisms in which they could be used, and the associations they have with other modeling constructs in hiBPM. The modeling constructs themselves are discussed in greater detail in subsequent sections of this chapter.

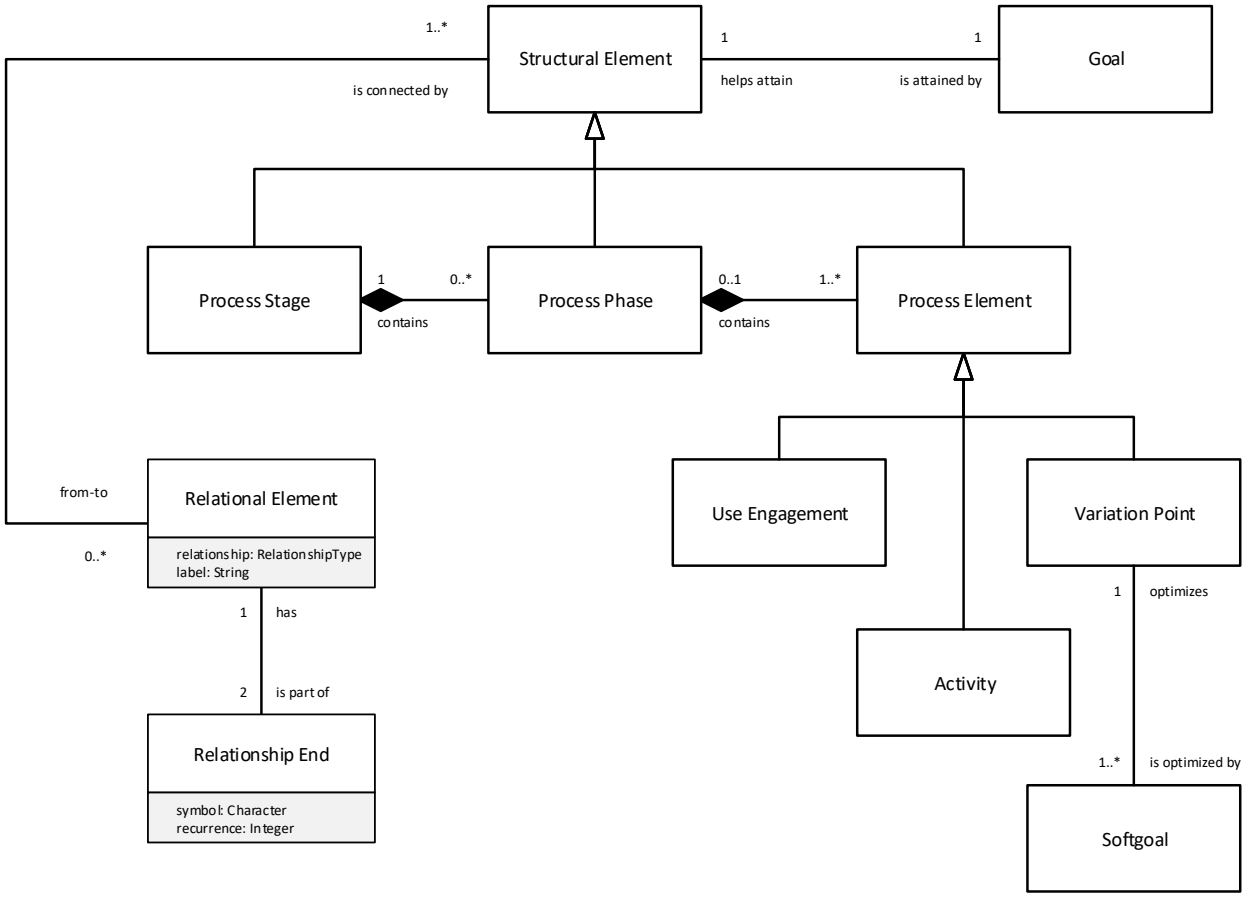

Fig. 5-1: Meta-Model for the hiBPM modeling framework

In the second part of the meta-model, we provide the necessary hiBPM modeling constructs that help in coming up with alternative process architecture configurations at key points in the hiBPM model. Enterprise objectives are attained through structural elements. This is shown in the meta-model as a **Structural Element** being associated with a **Goal**. The execution of the process elements in any process stage help attain objectives for a process stage; thereby process element objectives are aligned to, and contribute towards, process stage objectives. The hiBPM model may contain **Variation Points** where different possible **Process Element** configurations may be present. These may appear anywhere within the hiBPM model. During the analysis of the hiBPM model, we consider the different options of hiBPM design that exist at a variation point. Each possible option is referred to as a variant that exists at that variation point. These then contribute either positively or negatively towards the satisficing of one or more **Softgoals** that exists at that variation point. **Goals** and possible ways of reconfiguring the hiBPM model are discussed further in Chapter 6.

5.3 Structural Elements

Structural Elements are a set of constructs in the hiBPM model that provide a generic representation of the structure of activities for the domain under study. These elements are not meant to give a detailed description of the domain but rather are a selective representation of activities that provide some means for domain depiction and analysis. When selecting structural elements for creating hiBPM models, the emphasis is not on depth and detail of the resultant models but instead on the contribution that those structural elements have on model analysis. That is, only those structural elements are selected and shown if they help explain the design of the multitudes of business processes that exist, and if they contribute towards coming up with, or deliberating between, alternative design configurations of the hiBPM model. Detailed activities are not shown if they do not assist in coming up with alternative designs.

These structural elements represent various strategic, tactical and operational activities at different granularities and different levels to each other. The structural elements typically are meant to attain some business or technological purpose, and thus provide insight on "why" the processes are structured the way they are, rather than just providing insight into "how" individual business objectives are meant through process execution. Thus, these structural elements are associated with the accomplishment of specific goals in the enterprise. Therefore, their identification is done through understanding the goals (and sub-goals) that exist in various parts of the enterprise and the process tasks and activities that exist to accomplish them.

In hiBPM, there are three types of structural elements, Process Elements, Process Stages and Process Phases; these all have a label associated with them that describes their purpose and accomplishment. The label takes a particular form (verb action on the entity) and is usually written in a manner that represents the operation being performed. They take in specific inputs, perform some processing for attaining the associated objective, and generate an output. These structural elements work in conjunction to achieve some collective goal, with various configurations being possible.

5.3.1 Process Elements

A fundamental construct in the hiBPM modeling notation is that of the *Process Element*. Process elements are basic activity units which produce some output or outcome based on a set of control and data inputs. Process elements indicate the behaviour or actions performed by a domain actor or participant that results in an accomplishment of function, or the generation of measurable output. The specifics of how these actions are performed are not apparent nor required. Instead, the focus is on how the process element contributes towards the attainment of enterprise functions and the relative positioning of the process elements within the overall process architecture. Generally, process elements are at a level of granularity where they represent a functional or non-functional accomplishment. The execution of process elements may or may not be atomic, as that is not a pertinent requirement for selecting process elements. A process element may access enterprise resources to be able to attain the behaviour that is expected of them while being generally chained in some order where collectively, these process elements accomplish some enterprise objective. Here the output of one process element serves as the input of another process element for onward processing.

Process elements can also include the act of making decisions as part of the activity processing. As part of designing the hiBPM model, a decision may need to be made on which alternative design to proceed with. These decision points in the hiBPM model are also represented as process elements. There may be uncertainty in how the processing is performed at that process element, and the process element itself is represented as a *variation point* in the overall hiBPM model. Binding all possible alternative design options at the variation points allows for creating complete and inflexible process architecture, whereas leaving variation points open for binding provides for flexibility of configuring the process architecture. Designing hiBPM models for flexibility and configurability is discussed further in Chapter 6.

In Fig. 5-2, we present various ways that a process element may appear in a hiBPM model. Fig. 5-2(A) shows a single process element **Elicit Requirements** (represented by a rectangular shape with rounded corners) that receives the input **Product Roadmap**, processes it and generates the output **Requirements**. A process element has a label associated that describes the operation performed.

Fig. 5-2(B) shows a slight variation of this; here there are multiple inputs, **Customer Survey** and **Product Backlog,** that are collectively processed to generate an output. A process element only has one output. Depending on the type of input, we show the inputs as coming in from the left, top or bottom, with the output emerging from the right boundary of the process element. These are further discussed in subsequent sub-sections in this chapter.

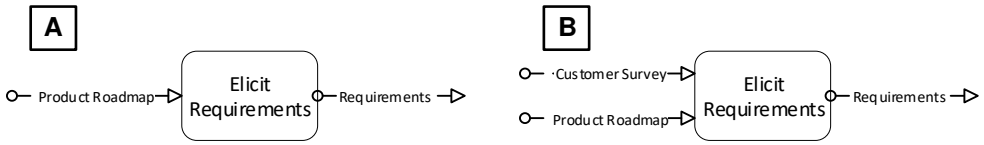

Fig. 5-2: (A) A process element with a single data input and a single data output. (B) A process element with two data inputs and a single data output

In Fig. 5-3 we show multiple process elements working together to form a chain of processing activities; these are **Elicit Requirements**, **Analyze Requirements** and **Create Specifications**. The links between the process elements are explained more in Section 5.4. These process elements collectively process the incoming inputs, **Customer Survey** and **Product Roadmap**, to produce the output **Requirements Specification**.

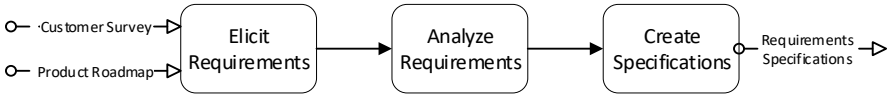

Fig. 5-3: A chain of process elements working collectively to process incoming data inputs to generate an output

While process elements may appear in anywhere in the hiBPM model, they are not independent constructs and are part of process stages and process phases. Process elements can be linked to another structural element (such as a process element or process phase) with relational elements within the same process stage, or to another process stage. Process phases and process stages are discussed in the next sections.

5.3.2 Process Stages

Process elements can be placed in collections called *Process Stages* based on the similarity of execution frequency, and their relationship to each other. Alternatively stated, process stages are collections of process elements that are to be executed collectively. A process stage contains one

or more process elements and will be structured in a manner where it delivers some enterprise objective in the form of functional or non-functional goals. Here a process stage represents a (sub-)process within a broader business process that attains a defined business objective. This output of a process stage is available to other subsequent process stages as an input to be used for its execution multiple times (as needed). The output of a process stage is available until the process stage re-executes, at which time the previous output is discarded and new output from that process stage is available. This concept will become important when recurrence relationships are discussed in later sections in this chapter.

We present two possible forms of process stages as they can exist within a hiBPM model in Fig. 5-4. Fig. 5-4(A) shows a simple process stage, **Manage Requirements**, with several process elements, **Elicit Requirements**, **Analyze Requirements** and **Create Specifications**. The process stage is depicted as a rectangle shape with rounded corners that contain one or more process elements within, with the process stage having an identifying label that appears in the top left corner. In Fig. 5-4(B) we show the same process stage; however, this time only depicting certain process elements with others omitted as these would not have contributed to the analysis. Additionally, the sequential relationship is not shown for all process elements as there may not be a sequential execution between the process elements in that process stage.

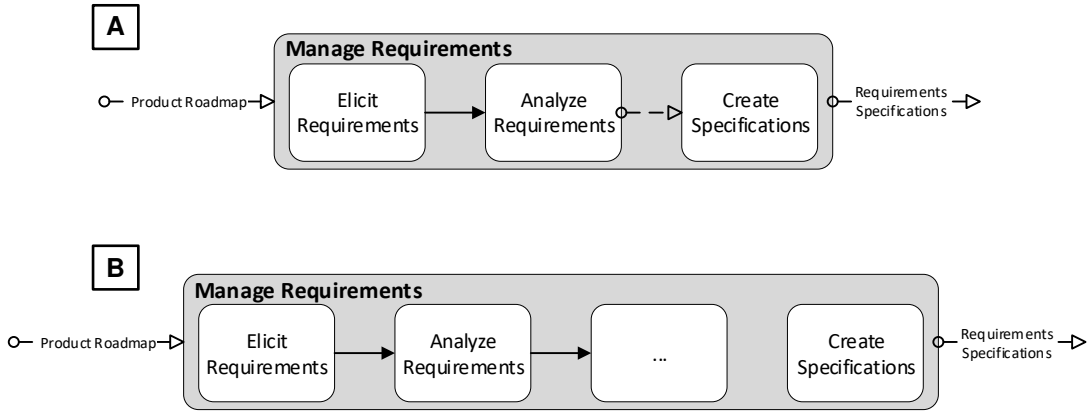

Fig. 5-4: (A) A process stage with multiple process elements that execute in some sequence (B) Process stage containing multiple process elements that execute collectively to attain a common objective

We illustrate another form of the same example that is presented in Fig. 5-5. Here two process stages, **Manage Requirements** and **Generate Specifications,** are setup in an upstream and

downstream configuration. By upstream we mean the process stage that executes first and results in an output that is then processed by another process stage (the downstream process stage). This is a relative temporal arrangement of process stages and does not imply the upstream process stage is immediately before the downstream process stage.

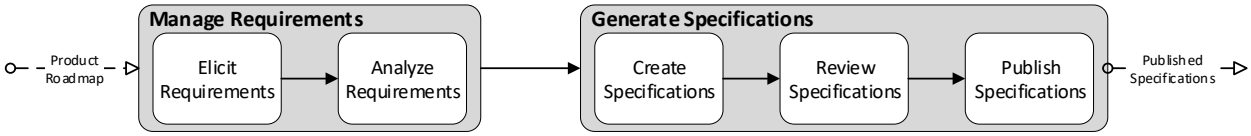

Fig. 5-5: Two process stages in an upstream and downstream configuration where the output of the upstream process stage acts as an input to the downstream process stage

Process stages can be represented as a loose collection of dependent actions that each run at approximately the same time duration and frequency for attaining a common goal. Process stages are determined by identifying process elements that need to be executed with the same frequency or are triggered by the same data-driven trigger. Further, these process elements should contribute to some common enterprise functional or non-functional goal. Once process elements are identified, they can be combined into a process stage with the output of that stage to be reused multiple times by the subsequent stages, thus saving time, money and possibly other resources. Process stages can be executed on-demand, having been triggered by appropriate events. We do not need to determine all the process elements that would be part of the process stage but only those that lend to hiBPM model analysis; thus, certain process elements or sequence relationships can be omitted when visualizing a process stage.

5.3.3 Process Phases

Another structural element is *Process Phases,* which are collections of process elements within a process stage that produce the same result irrespective of the arrangement of those process elements. We highlight such parts within the process stage as any temporal reordering of the process elements does not result in any change in the outcome of the process phase. This is important as moving process elements within a process stage may or may not result in a different output for that process stage; we need to differentiate between both these situations. A process stage may contain one or more process phases, with the output of one process phase being used by

the subsequent process phases or process elements of the same process stage instance. Introducing process phases as a modeling construct helps to reduce the number of possible process element placement alternatives. This leads to a decrease in analysis effort and allows for more focus on more important issues by abstracting over some hiBPM modeling details, like the specific sequential arrangement of process elements.

When determining process phases in a process stage, we first understand the temporal constraints of process elements, as this then helps identify process phases that exist within a process stage. By temporal constraints, we mean that if changing the order of sequential process elements results in a different output (in the form of non-satisfaction of a functional or non-functional goal) of the process stage that the process phase is a part of, then we say that there is a temporal constraint. Conversely, if changing the order does *not* result in a change in the output (which is tied to the functional or non-functional goal for that process stage), we say that there is no temporal constraint. Given this collection of process elements within a process stage, process elements that do not need to be in a defined execution sequence should be encapsulated as part of a process phase. For other process elements, they can be shown outside of this process phase, with sequential relationships indicating the temporal order of execution. Unlike process elements and process stages, process phases need not be associated with any functional and non-functional goals, and are just representative of the ordering of the process elements within a process stage.

A process phase is represented with a rectangle with rounded corners and dashed lines (instead of the solid lines used in process stages and process elements). A label helps describe and differentiate the process phase, although this is not strictly necessary. In Fig. 5-6, we show a process stage, **Develop Software Product**, with multiple process elements, some of which are part of a process phase, **Commit and Test**. The process elements outside of the process phase (**Review Design** and **Implement Product Feature**) have a sequential ordering (indicated by the sequence flow) whereas the ones within the process phase, **Commit Code Change** and **Perform QA Testing**, have none. This is to suggest that moving the process elements within the process phase does not result in a change in the output of that process phase or that of the process stage that this process phase is part of.

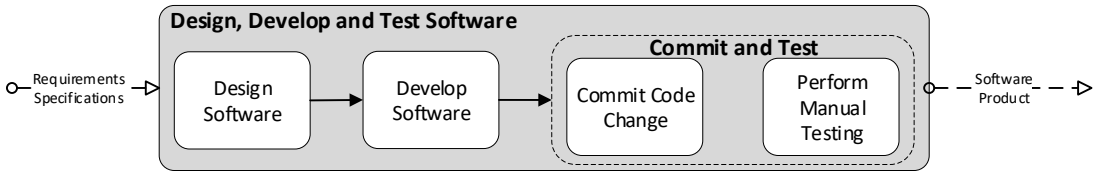

Fig. 5-6: A process stage with a combination of process elements and a process phase

In Fig. 5-7, we show a process stage with two process phases, Design and Develop and Commit and Test, that demonstrate the same behaviour as described previously. Here we show the output of the first process phase, Design and Develop, going into the second process phase, Commit and Test; however, other possible configurations may exist. The input to a process phase may be from another process element, process phase, or even a process stage. The output of a process phase can go to another process element or another process phase only.

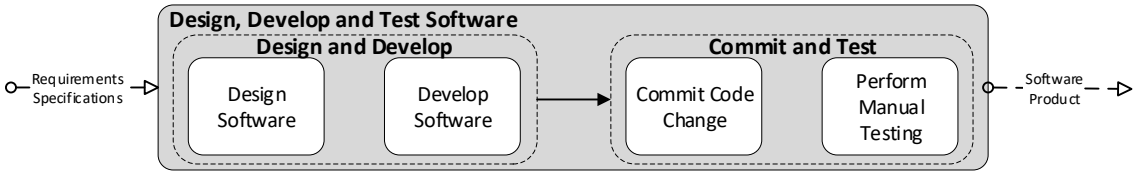

Fig. 5-7: A process stage with two process phases with the output of the first process phase feeding into the second

5.3.4 User Engagement Process Elements

hiBPM models emphasize the nature of relationships between technology and software processes where an artifact or tool is generated, or some data produced, and where it is ultimately used during the execution of a process. Such an exchange may not be as one-dimensional as it would initially appear. To explain this, let us assume a (human or system) user that is involved in the execution of a activity within some business process. This user relies on the output produced by some software system for either decision making or process execution. Here the user has learnt how to use and adapt to such systems as part of executing their business processes, where the systems themselves were designed and configured as per each enterprise's unique requirements.

However, in dynamic enterprises, the output produced by the process stage could change, not just in the output value, but also in the format and nature of the output. For example, upon reaching the desired levels of trust, confidence, accuracy, and reliability, the process stages may produce

different tools, with the software systems themselves increasingly becoming more entrenched into the enterprise. The above results in changes to the design and execution of business processes, and in the responsibilities of users in charge of these business processes. Thus, it is important to indicate the relationships between the output of software process stages (that produce some data or tools) that are then inputs to some other structural element (like a process element), with the likelihood that this relationship will change over time due to the inclusion of factors such as assignment of responsibilities for communication/collaboration, issues of trust, and the ability to override the system. We refer to such relationships as user engagements, implying that a user (human-based or system-based) is engaging in the use of an output from another process stage.

As part of applying hiBPM, we map user engagements into process elements where the applicable user engagements (injected into those process elements) are then are executed by both human process participants and activities executed by information systems. Such a process element, **Manage Monthly Budget**, is shown in a shaded form in the hiBPM model shown in Fig. 5-8 to differentiate it from other process elements within the same process stage, **Manage Monthly Expenses**. Such a process element has parameters that are likely to be different across every execution instance or is expected to receive inputs that can change (along a spectrum of user engagement dimensions) over time. An example being where the output of the **Manage Monthly Budget** is purely information with the user having to do the actual balancing of their budget. Later on, the user engagement mode may change and now the output of the **Manage Monthly Budget** is completely balanced, and the user is just reviewing the provided budget and approving it.

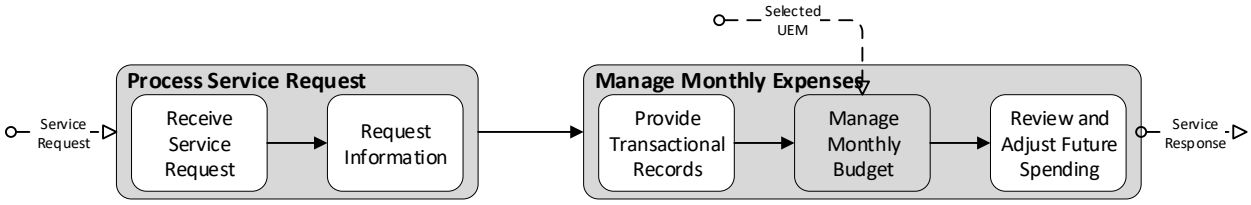

Fig. 5-8: A user engagement process element where the user engagement mode changes based on input from a separate process stage

5.3.5 Process Boundaries

Within any process architecture, there would be areas where the structural elements have similar traits. This may be to do with the frequency of execution, the type of process output, the behaviour of the structural elements, or generally the contribution that they have to enterprise strategic, tactical, or operational goals. Through process boundaries, we ensure that we can identify and demarcate between two adjacent, yet dissimilar, sections that may exist within a hiBPM model, and their relative positioning to each other. We include process boundaries in structural elements as it helps differentiate between two adjacent process stages and provides additional depth (for understanding and analysis purposes) to the structure of the hiBPM model.

Identifying such process boundaries permits the determining of common attributes of process stages. For example, a process stage may exist at a *higher* level (in terms of process stage objective, or recurrence) than another *lower* level process stage. Among these processes, there can be a number of different relationships. For instance, a higher-level process can control a lower-level one by adjusting that process' parameters. On the other hand, there can be another relationship between processes, where a higher-level process designs (or redesigns) a specification for the lower-level one. Through process boundaries, we can show the difference between process stages that exist at different levels and are connected by relational elements, such as recurrence relationships, design-use relationships, and plan-execute relationships. This is discussed further in subsequent sections of this chapter. The "hi" in hiBPM refers to the presence of multiple business processes in the domain space, with some being at a higher level than others.

Process boundaries can only exist between two process stages and not any other type or combination of structural elements. This is because the output of a process stage is persistent (i.e., can exist after the process stage executes) and can be an input to a process stage in another part of the hiBPM model. This is opposed to the output of process elements or process phases which are immediately passed on to subsequent process elements or process phases (within the same process stage). In case one process phase is operating at a higher execution level than another, it would be better to show them as two process stages with a process boundary in between. Process boundaries are indicated by a simple dashed line that is drawn between two process stages. We try to have this

process boundary sketched horizontally, with one process stage placed higher than the other. This helps with the visualization of multiple levels of process stages and how one contributes to the other in a particular manner.

Fig. 5-9 shows two process stages **Build Business Understanding** and **Manage Requirements for Mobile App**, with a process boundary between them. Recall that in Chapter 4 we introduced the concept of two-speed enterprises where some parts of the enterprise operate at a different speed than others. Here the process boundary, shown by the dash-line separating both process stages, is indicating of the two process stages belonging to the separate parts of the enterprise, **Enterprise-Side** and **Customer-Side**. Other examples of process boundaries separating process stages across Design-Use and plan-execute relationships are given in subsequent sections of this Chapter.

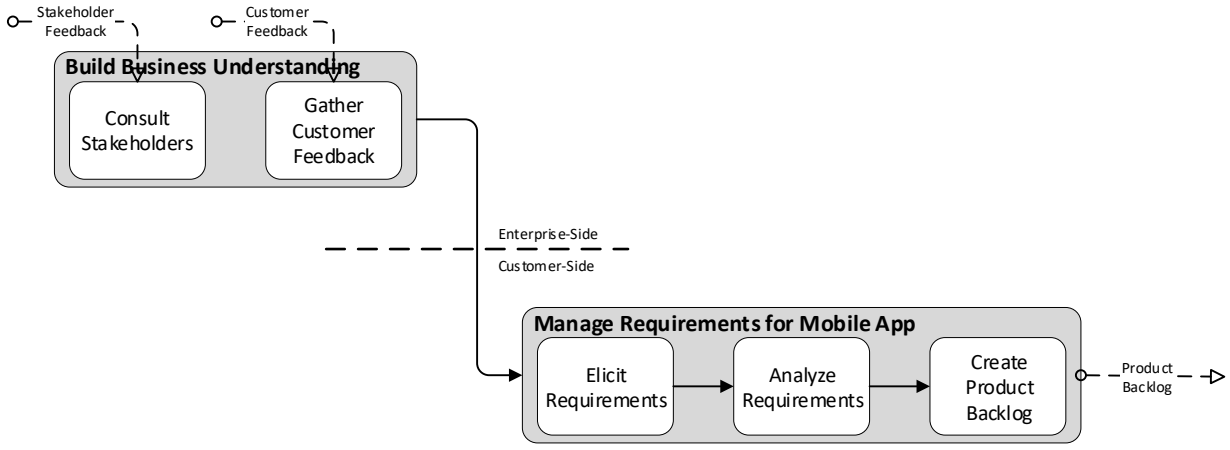

Fig. 5-9: Process boundary showing the Enterprise-Side and Customer-Side divide between two process stages

Process elements (part of a particular process stage) can be moved across process boundaries (to another process stage on the other side of the boundary), though they then inherit the behavioural traits of the target process stage. Moving process elements across phase boundaries may also affect the quality of decisions and the outcome of actions. Thus, it's important to understand the ramifications of placing process elements during the design of a hiBPM process architecture and the resulting behaviour that may emerge. We discuss the placement and movement of process elements in greater detail in Chapter 6.

5.4 Relational Elements

As with any process modeling approach, there exist several relational elements in hiBPM that associate multiple structural elements together. They are an essential part of the overall process architecture by providing the relationship aspect through linkages. A relational element connects two structural elements with one source and one (or more) destination connector, and can only be between structural elements. Moreover, a relational element can connect two different types of structural elements. E.g. the originating structural element may be the output of a process stage that goes into a process element that is part of another process stage. Such combinations of structural elements connected by relational elements are permissible in hiBPM.

Relational elements have a direction indicator (an arrow), through which the source and destination structural element can be determined. The arrow broadly and generally provides information on the upstream and downstream processing of activities but is not meant to determine sequential execution (in the sense of BPMN). It also permits identification of the initiator and recipient of the object transfer that takes place as part of this association. The arrow emerges from the left side of a structural element and terminates at either the right, top or bottom side of a destination structural element. The shape and flow of this arrow vary based on which relation is being represented; these are discussed in further detail in the following sub-sections of this chapter. In cases where process stages exist across a process boundary, a relational element would connect them while crossing the process boundary for providing a means of object transfer and association. Despite this, the object transfer occurring through the relational element does not get transformed in any way when crossing the process boundary and gets delivered in the originating form.

In hiBPM, there are various forms of relationships; these include data flows, sequence flows, recurrence relationships, sense-and-control flows, design-use relationships, and plan-execute relationships. The table below shows the permitted relationships between an originating and destination structural elements.

Table 5: Forms of relationships in hiBPM

	DF	SF	RECUR	SF	CF	D-U	P-E
PE to PE	✓	✓	×	×	×	×	×
PE to PP	✓	✓	×	×	×	×	×
PE to PS	×	×	×	×	×	×	×
PP to PE	✓	✓	×	×	×	×	×
PP to PP	✓	✓	×	×	×	×	×
PP to PS	×	×	×	×	×	×	×
PS to PE	✓	×	×	✓	✓	✓	✓
PS to PP	✓	×	×	✓	✓	✓	✓
PS to PS	✓	✓	✓	✓	✓	✓	✓

DF = Data Flow
SF = Sequence Flow
RECUR = Recurrence
SF = Sense Flow
CF = Control Flow
D-U = Design-Use
P-E = Plan-Execute

5.4.1 Data Flow and Sequence Flow Relationships

Data Flow and Sequence Flow are elementary constructs that link two structural elements in any hiBPM model. The output of an upstream structural element is transferred as an input to a downstream structural element through these flow relationships. The execution of a downstream structural element cannot start until the upstream structural elements (as represented by the flows) is completed.

Data flows are a means of representing the transfer of information objects from a structural element to another and indicate a simple association between structural elements. These information objects could take the form of data that is required by the downstream structural element for its processing. In Fig. 5-10, we show a data flow relationship, **Raw Requirements**, between two process stages, **Manage Requirements** and **Generate Specifications**.

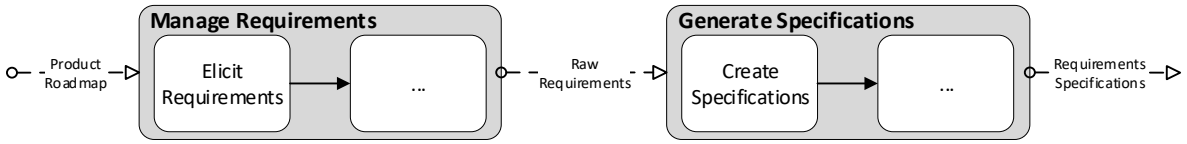

Fig. 5-10: Data flow relationship between two process stages

Sequence flows represent the temporal arrangement between two structural elements, where an upstream structural element would be linked to a downstream structural element by a sequence flow. They provide some indication of the sequential execution of process stages in the overall process architecture and help with the readability of the model. Sequence flows are shown as solid lines that emerge from one structural element and end at another, whereas data flows are shown as dashed lines.

Data flow and sequence flow may appear separately in a hiBPM model or shown collectively, which means that there would be a temporal arrangement between two structural elements that also happens to have a transfer of data. In this case, we use the same sequence flow notation but add a data flow label to it. In Fig. 5-11(A), we use the same hiBPM model as that in Fig. 5-10, this time emphasizing that the there is a sequence flow between the two process stages. Fig. 5-11(B) shows an example where the sequence flow label is not shown as it doesn't contribute to the analysis of the hiBPM model.

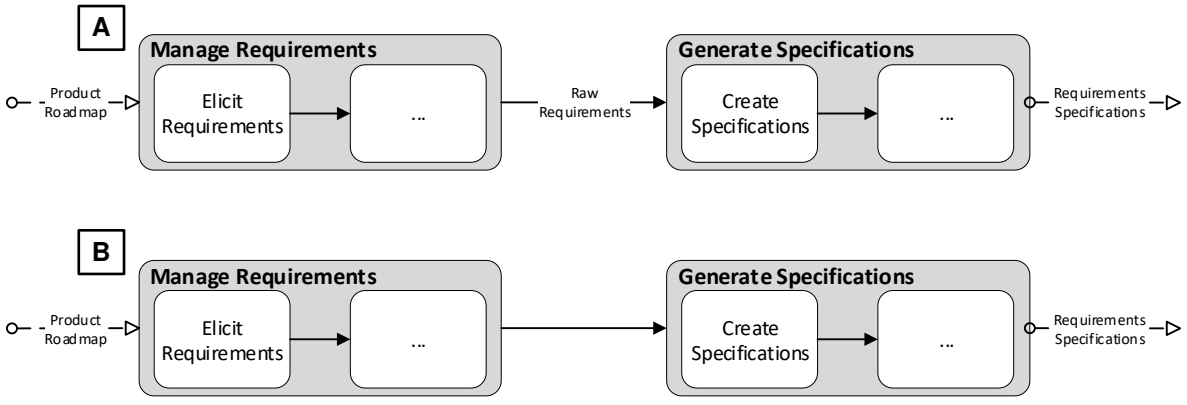

Fig. 5-11: (A) A sequence flow between two process stages with a data label present, (B) A sequence flow without the data label present

Labels (that indicates the information object being transferred) may be optional for both data flows and sequence flows. Both these flows are generally represented directionally from left-to-right and

may bend at 90 degrees depending on the positioning of the downstream process stage. While there is flexibility in drawing the flows, the general direction of flow should be such so that the temporal ordering and higher-level ordering is apparent in the hiBPM model.

5.4.2 Recurrence Relationships

Recurrence relationships exists between two process stages when both stages are at separate execution frequencies. By this, we mean that an upstream process stage may execute at a rate different from a downstream process stage. This difference in execution frequency between both process stages is represented by a recurrence relationship. Here, an attribute is associated with this link that indicates the relative recurrence of process stages at both sides of the recurrence relationship. Note, the emphasis here is on the relative execution between both associated process stages, and not their absolute execution frequencies.

Within a recurrence relationship, the process stage that executes more frequently is referred to as having higher recurrence (and exists at the higher recurrence side of the relationship) and the process stage that executes less frequently is referred to as having lower recurrence (and exists at the lower recurrence side of the relationship). Recurrence is primarily a temporal repetition motion. Through recurrence relationships, we aim to highlight the relative execution frequencies between different segments of the hiBPM model that permit analysis amongst various design variants.

There are several possible cardinality relationships between two process stages. Each of the below representation of recurrence relationships between process stages is shown in Fig. 5-12. Here **M** and **N** are variables that represent the execution of the process stage. They are not meant to represent absolute values and generally signify that a process stage having a recurrence of **M** or **N**.

- **1:N** - Each execution of **Manage Requirements** results in multiple executions of **Develop and Deploy Software**. In other words, the output of the higher recurrence **Manage Requirements** process stage is available to be used as an input by a lower recurrence **Develop and Deploy Software** process stage for multiple execution cycles. This is the more common case in a typical hiBPM model.

- **M:N** - The **Develop and Deploy Software** may execute multiple times before **Monitor Production Environment** would execute. Here **N** > **M** so the lower recurrence **Monitor Environment** process stage will execute **N** times for every **M** executions for the higher recurrence **Develop and Deploy Software** process stage.
- **N:1** - The **Monitor Production Environment** will execute multiple times before the **Analyze and Audit Environment** would execute. This can show situations where higher recurrence **Monitor Environment** process stage execute once for multiple lower recurrence **Analyze and Audit Environment** process stage executions. Such a configuration allows **Analyze and Audit Environment** to aggregate and analyze the data that **Monitor Production Environment** generates.

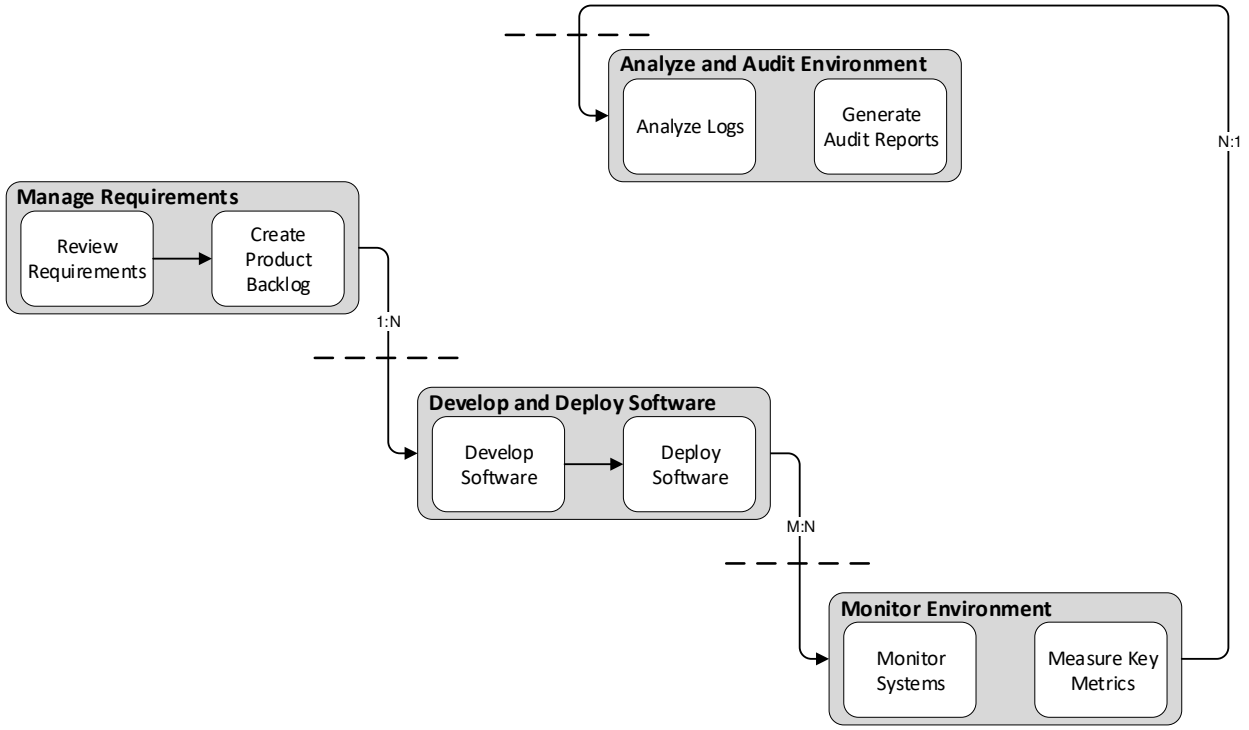

Fig. 5-12: Multiple 1:N, M:N and N:1 recurrence relationships between process stages

Recurrence between combinations of process elements and process phases is assumed to be 1:1. If a recurrence relationship other than 1:1 becomes apparent, then the process elements or process phases need to be converted to process stages. Introducing recurrence relationships in hiBPM models allows for flexibility in representing process stage execution amongst the various parts of the process architecture.

In the recurrence relationship, we assume that at the execution completion of an upstream process stage automatically results in the downstream process stage execution. However, this may not always be the case, and at times, the downstream process stage may need to be manually executed; for this, we rely on triggers. *Trigger* relationships are a special kind of data flow that exists between a structural element (process element, process phase or process stage) and another process stage. Triggers are used to indicate that the completion of an upstream process stage (or a process element) results in the immediate execution of an immediately downstream process stage. Thus, this represents some form of temporal relationship between these two associated process stages.

Triggering conditions pertain to changes in context and their availability, thus detected changes in context allow for reconfiguration in other parts of the hiBPM model. When a stage is triggered, it is re-executed, with the triggers themselves being manual or automatic. Manual triggers are those that are human-initiated and can be "fired" even if triggering conditions are not satisfied. Automatic triggers are those that are fired by the system when certain triggering conditions are satisfied. However, in hiBPM the specific temporal details, e.g. on how the recurrence is triggered, are of less concern compared to process architecture redesign and thus not specially indicated in a hiBPM model.

5.4.3 Design-Use Relationship

The execution of a process stage can result in the creation of a tool, capability or artifact that can be used, with the tool or capability referred to as a "design". This design is used to build an artifact that is then made available to downstream process stages to be utilized repeatedly in the stage's execution. In hiBPM, designs are conceptualized as pre-built black box artifacts that can adopt different forms; they may be physical objects, a digitized entity or even be informational. The designs are black-boxes as a user that uses the design artifact as part of the process stage execution does not care (nether is informed) about the (internal) structure of a design artifact or how its built and is only concerned about whether the functional and non-function objectives are achieved when invoked during usage.

Such an act (of creating this tool or capability) can be considered as having two distinct process stages, one creating the designed artifact, with the other (repeatedly) using the designed artifact.

Thus, a design-use relationship exists between these process stages. A design-use relationship only exists if the design is used in a downstream, immediate process stage. In design-use relationships, we consider the association between how the designs are built and the usage of the artifact as it not only allows a contemplation on how designs are developed but also how they are integrated and used in business processes. This is useful as it ensures that suitable processes and data dependencies are fulfilled at or before that point.

In Fig. 5-13, we show a design-use relationship between two process stages, **Setup Environment** and **Deploy to Production**, with the process creating the artifact called the design process stage and the one using the artifact called the use process stage. The relationship between the two process stages indicates the flow of the design, **Environment Template**, from the design stage to the use stage, with a **"U"** annotation representing using of design in the execution in the use stage, which enters the latter at the bottom following the Mechanism arrow notation from IDEF0. The figure also shows a design-use process boundary, thus differentiating between the designing part of the hiBPM model and the using part of the hiBPM model.

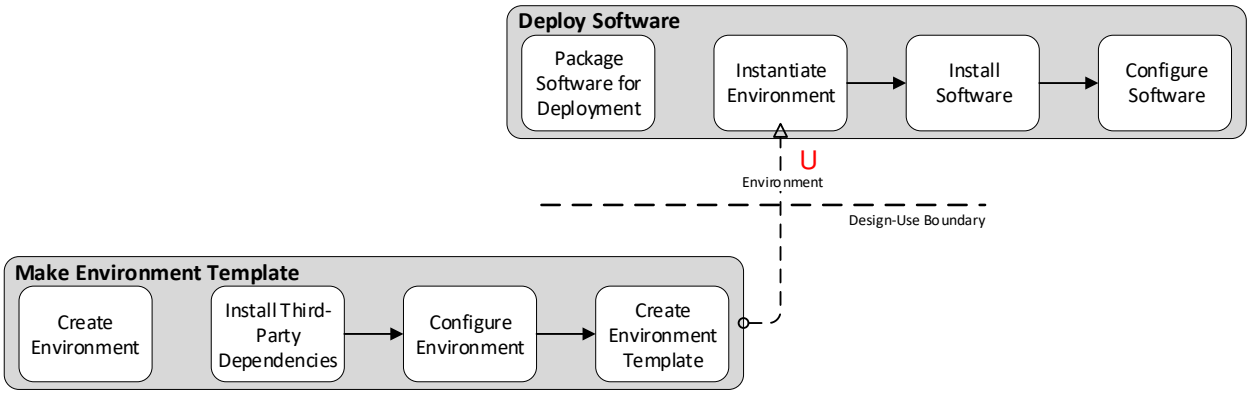

Fig. 5-13: Design-use relationships between two process stages that exist across a design-use boundary

A primary motivator for design-use relationships in hiBPM is to represent arrangements in a process architecture where certain process stages can invoke a capability without knowing how to create it, i.e., following the principles of encapsulation and abstraction. Thus, designs can be considered as black-box abstractions; we discuss this in more detail in Chapter 6. Design-use relationships also allow for introducing designs in the process architecture for (a) enabling

automation, and (b) reducing the process execution time and cost (amongst other reasons). The assumption is that the design stage will not be executed just once to produce a tool or a capability.

5.4.4 Plan-Execute Relationship

In typical business process modeling (in the context of enterprise agility), the process model describes the process that is to be executed, but not how this process gets determined. In hiBPM, a process stage can produce an output that is a plan for a subsequent process stage to execute. A *plan* provides instructions for execution of activities to accomplish enterprise functional and non-functional goals while simultaneously reducing the space of possible process execution possibilities, as there may be several possibilities to attain these enterprise goals. Thus, a plan is the output of the planning process stage and can be an arrangement of actions, or a set of specifications that describes the method, means and constraints of executing the plan. Downstream process stages need to be aware of the information as codified in the plan to ensure proper execution. We call the process stage where the plan is produced the planning process stage, while the subsequent stage is called the execution process stage. Such an association between the two sides is called a plan-execute relationship.

In Fig. 5-14, we show a plan-execute relationship between two process stages, **Plan for Testing** and **Create Test Scripts**. The relationship between the two process stages indicates the flow of the plan, **Test Plan**, from the planning stage to the executing stage, with an **"X"** annotation representing the executing of the plan execute stage.

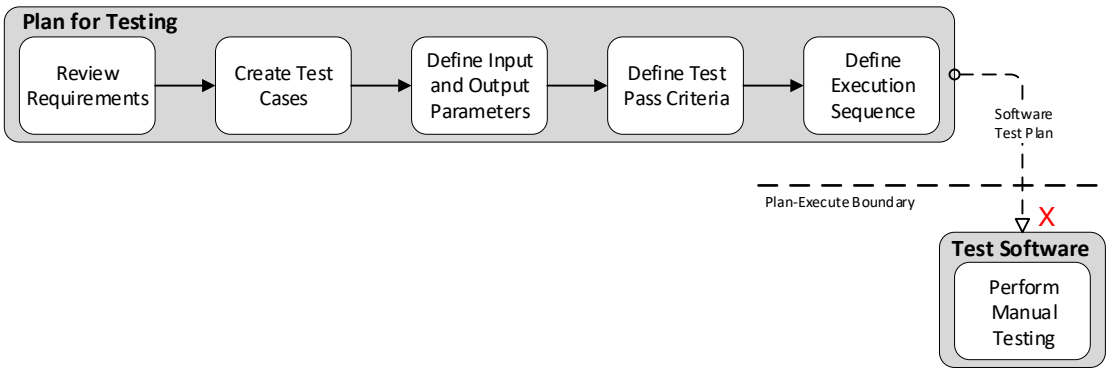

Fig. 5-14: Modeling notation for plan-execute relationships

Through plan-execute relationships, the hiBPM model can capture the relationship between the two process stages, i.e., where the plans are devised and where they are executed. Note that due to their nature, planning stages do not achieve domain-specific objectives – i.e., they do not change the state of the system or its environment, but rather indirectly assist in the attainment of these objectives through the execution of the plan. Processes may produce plans that are then executed by downstream processes to induce some change in the overall process architecture design or result in some behavioural change in the business process execution. The planning process stage devises the plan irrespective of how the execution process stage will execute it. The execution process stage is aware of the plan internals to interpret and best execute the plan based on requirements and trade-off analysis that can be done as part of its execution.

In plan-execute, the planning stage can be seen as being "about" or as "operating on" the execution stage, thus creating a higher-order effect when one process constructs another. The benefits of this include the ability to represent and analyze the capabilities of organizations/systems to evolve in the face of changes, which is crucial for analyzing the sustainability of systems in highly volatile domains. As mentioned previously, the "hi" in hiBPM refers to the presence of multiple business processes, with some being at a higher level than others. Here the planning stage exists at a higher level than the processes executing the plan.

The primary motivation for such a process relationship is to identify two distinct segments, each with their attributes and distinct process goals. One is responsible for planning, whereas the other is responsible for the execution of the plan. These are separate process segments but have an association relationship with each other because both goals are part of accomplishing some upper-level objective. So, attaining this upper-level objective requires the conceptualization of both plan and execute process segments. As a plan is informational, thus data flow relationships can be considered as a plan-execute relationship if a plan is being transferred between two stages and that plan is directly executed by the immediate downstream process stage.

While not shown in Fig. 5-14, we can visualization another level of plan-execute relationships to be present if we introduce a third process stage, say Process Stage_C to an existing plan-execute relationship between Process Stage_A and Process Stage_B. An execution process stage Process Stage_B

to a planning process stage Process Stage$_A$ can *also* be a planning stage for some execution process stage Process Stage$_C$. Thus there is a cascading relationships where Process Stage$_A$ produces a plan that is executed by Process Stage$_B$ which further provides a plan that is executed by Process Stage$_C$.

5.4.5 Sense and Control Relationships

In order to understand enterprise adaptiveness, we need to highlight the various sense-and-response paths that are implicit in a hiBPM model, and to see if they can support a desired level of enterprise adaptiveness. Such paths in hiBPM are used for revealing unique adaptation relationships between two process stages. There may exist process stages that are responsible for "sensing" different environmental parameters. These are then used to influence (or "control") the design or execution of other process stages in the hiBPM model. The visual representation in a hiBPM model supports the analysis of feedback paths e.g., whether the adaptation loop is appropriately designed, where to source the data flow for feedback, and where to insert the feedback flow back into a some other process stage.

By differentiating control and design inputs from regular data inputs, and sensing from regular outputs, we can locate adaptive loops as they exist within a process architecture model. In annotated process models, we can capture feedback path details through the use of design, sense, and control flows. To make feedback loops explicit, message flow that serves a Sense purpose are marked with an **"S"** on an output from a process stage where Control is represented as a message flow adorned with **"C"** to mark a message flow that serves as a control input to a process stage. The change is either through a "control" flow constraining the possible options for the target process at runtime or through an "execute" flow that changes the space of options for its target process by creating new capabilities. Hence, there is a hierarchy among processes that reflects their relative control order. The execution frequencies of both these levels typically differ as well.

Mechanisms indicate a capability output that supports a process stage. In hiBPM, we show mechanisms as designs, as mechanisms are to be used by downstream process stages. The terminology of "control" and "mechanism" flows are borrowed from the IDEF0 language. There can exist feedforward paths as well in process architecture, where a control signal flow from an

external environment, which results in some action in a process stage. Such feedforward paths are signified using the "C" symbol.

Fig. 5-15 shows sense-and-control feedback paths that exist between two process stages for the example introduced in Chapter 4. The **Review Production Metrics** process stage receives a sense flow from the **Monitor Production Environment** with a set of **Production Metrics** that are then used to then influence the future software feature development. This is represented by the control flow **Product Modification Tasks** between the **Review Production Metrics** and **Design, Develop and Deploy Software Product** process stages.

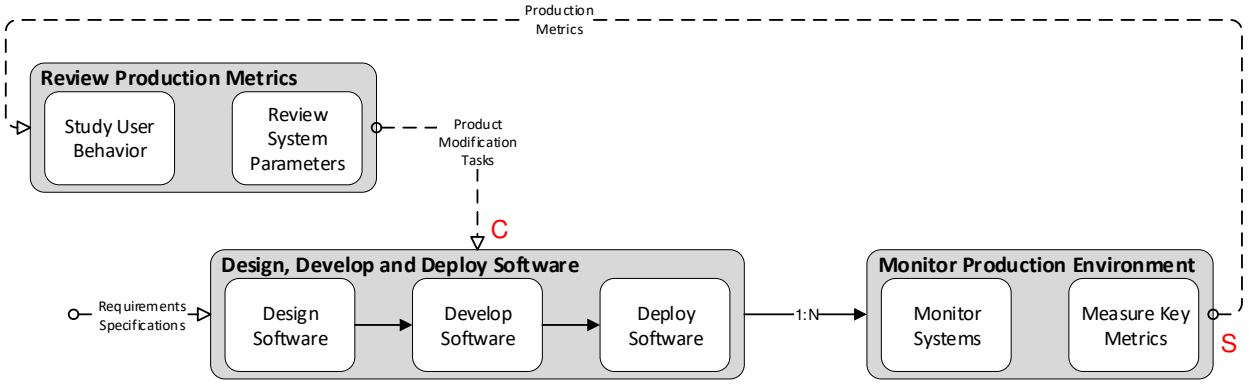

Fig. 5-15: Sense-and-control relationships for responding to production metrics

In hiBPM, a higher-order process stage can be a design or plan process stage to some lower-order use or execution process stages. It senses how well its lower-level process works and may change the way it operates. For example, in certain enterprises, the execute/use process stages may be at run-time (i.e., at machine-scale execution) while the plan/design process stages may be at design-time (i.e., at human-scale execution). Through feedback paths (along with design-use and plan-execute relationships), such a configuration can be shown as part of the overall process architecture. Most of the data will likely be monitored in an enterprise at the operational stage. This applies even to the data that is not analyzed at that level but is aggregated and can trigger a change in a much higher-level (earlier) stage.

5.5 Comparison with Related Work

5.5.1 Process Elements

Most conceptual modeling techniques that are process-centric, or have process considerations, have similar notions of activities. BPMN has the concept of activities which is a generic term for work that is performed as part of business process execution. BPMN defines activities as either being atomic or compound (i.e., a collection of atomic tasks). In SPEM too, there exists the notion of activities which are basic units of work within a software process. Other process modeling approaches, like flowcharts and UML sequence diagrams, too have concept of tasks that capture an operation that is performed within the process being studied. In goal-based and agent-based modeling techniques, there exists the idea of tasks. Goal-based techniques (like the NFR framework) decompose a high-level goal into subsequent goals until we get to operationalized goals (or tasks), whereas in agent-based techniques (like iStar) tasks are associated with the accomplishment of actor goals, or dependencies between actors. CMMI [179], has a construct called "process element," which is sub-process within another process. These process elements are interconnected together to form a more extensive process architecture. Our motive for process elements is to (a) capture fundamental activities in a process architecture model, and (b) show how their movements can result in different process architectural configurations. Therefore, the process elements in hiBPM are present not to provide depth and detail to the domain model but identify areas of the domain that could be reconfigured by moving the process element along various dimensions of change.

5.5.2 Process Stages

There are several approaches to express variability in business processes, which generally involve adding or removing process fragments in order to customize the business process for a particular situation [66]. Non-functional requirements are also used for modeling and deciding between multiple variants in the business process at run-time [70]. In SPEM, *task definitions* provide a detailed explanation for work that is to be performed to attain a particular software development objective [82]. Agent-oriented and goal-oriented modeling approaches, too, have tasks and resources that contribute to goal attaining. Our concept of process stages is different from these as

we considering process stages to be sub-processes that exist in the overall process architecture, that may span multiple business processes, but are collections of process elements that are collectively executed to attain a common functional or non-functional goal. Further, we externalize the reconfigurability of process design or process execution in the form of business rules. We then consolidate the commonalities of process execution into one process stage, with other varying parts into another process stage, whose execution behaviour varies in response to context changes. The latter process stage can then be reconfigured as necessary through analysis of the hiBPM model. This introduces flexibility in the design of the process architecture. This idea is conceptually similar to that of process line engineering [87], where process line architectures consist of processes that incorporate both commonality and variability.

5.5.3 Process Phases

In SPEM [82], groups are a collection of graphical elements that belong to the same category. A trait of this grouping is that this does not affect the sequence flows with the group. This is similar to our concept of hiBPM process phases, where the order of execution of process elements is not relevant to the output of the process phase. However, in SPEM the primary idea is to collapse the graphical elements and simply the diagram, whereas we introduce process phases to a hiBPM model to show that the placement of process elements within the model would not result in any changes in the output of the process stage, i.e., to reduce the possible design space of hiBPM model reconfigurations. Slightly related conceptually to process phases is the idea of phases as discussed in variability in the design of software product lines [47], with the difference being that unbounded variation points in downstream phases do not affect the design of upstream phases.

5.5.4 User Engagement Process Elements

Cognition is the process by which “an autonomous system perceives its environment, learns from experience, anticipates the outcome of events, acts to pursue goals, and adapts to changing circumstances" [183]. The impact of cognitive computing on business process management (BPM) is covered in [64] where multiple types and levels of business processes are discussed – transaction-intensive, judgement-intensive, and design & strategy support processes – resulting from the incorporation of cognitive capabilities within an enterprise and how cognitive processes

enablement can be attained. Human-Computer Interaction (HCI) involves the study of the communication of information and data between a human user and a computer through an interface, using various modes and mechanisms such as input/output peripheral devices, voice, text, sound, gesture control etc. [184]. HCI aims to greatly improve the experience of user interaction by leveraging the principles of information design, interaction design and sensorial design [185]. We approach the problem of user engagement with machines differently in that we consider the shifts in user approaches for decision making (as part of a routine business process execution) that are caused by changes in the design and implementation of cognitive agents.

Much research has also been dedicated to the formal handling of contexts in the area of Artificial Intelligence and Knowledge Representation and Reasoning [186]. In most modeling frameworks, "context" refers to what is outside the system that is being modelled; as the context changes, the model needs to be revised. What the context is, and how to deal with changes in context, is not modelled. Additionally, previous approaches for designing context-aware business processes focus more on specifying how organizations and their processes would react to changing business contexts. Context-aware business processes emphasize the use of context at design-time for business processes configuration through ongoing monitoring and capturing of context [187].

The approach proposed here differs from those above by considering context as part of the overall hiBPM model, and how changes in the context in one part of a hiBPM model may result in design and run-time changes in another region. Additionally, we focus on the design changes resulting where the cognitive agent is integrated into the business process as these changes are generally complicated and result in reconfiguration in at a systems level, process level, or how the user utilizes the cognitive agent. Organizations need to proactively explore the space of context information, identify and sense relevant context, and utilize it in a business process. The intention behind context determination and incorporation for business process redesign is to ensure that business processes continue to be aligned with enterprise objectives (modelled as goal models [171]) while considering design constraints in the business process.

5.5.5 Process Boundaries

Boundary is a familiar concept in enterprise architecture and systems design. ArchiMate has a concept of layers which represent different architecture segments. These segments divide the architecture into the business layer, application layer, and technology layer. The architectural components in each layer have distinct purposes in the enterprise. In systems design, two temporal layers are obvious; one being the design-time and the other being the run-time. An imaginary boundary can be considered between activities that happen at design-time versus activities that occur at run-time. In hiBPM, process boundaries do not represent temporal distinctions (like design-time vs. run-time) nor do they slice the process architecture into separate areas (like in ArchiMate). Process boundaries separate different areas of the hiBPM model where moving process elements across these process boundaries results in different possible hiBPM configurations and behavioural outcome.

5.5.6 Data Flow and Sequence Flow Relationships

We use the same graphical notations for sequence flows and data flows as they are defined in BPMN (with hiBPM data flows being matched to BPMN message flows, and hiBPM sequence flows being matched to BPMN sequence flows). Conceptually sequence flows and data flows are similar between hiBPM and BPMN, in the sense that sequence flows show the sequential arrangement between various process constructs and that data flows show exchange of information between two process constructs. Similar rules for the visualization also apply.

5.5.7 Recurrence Relationships

Approaches for representing cardinal relationships and different execution frequencies are present in different modeling approaches. ERD diagrams have concepts of cardinality but these show the relative instantiations between two entities. In software systems design, APIs are used to differentiate two areas that are governed by differing development behaviour and lifecycle. In hiBPM, we introduce recurrence relationships to show how different process stages can execute at relative frequencies and moving process elements between two such process stages would have implications on non-functional goal satisficing. Triggers are a much-studied topic in systems

design but there are some differences in how triggers are conceptualized here. In hiBPM, triggers are considered in the context of the overall process architecture and are fired when a process stage is to be executed due to the completion of a process stage elsewhere in the process architecture, although triggers are not explicitly identified through some hiBPM notation.

5.5.8 Design-Use Relationship

Simon [180] suggests that the design of artificial systems depends on the motives of the designer and how the design is expected to be used. The designer indicates the boundaries of the design (i.e., the interface), which separates the "inner environment" and the "outer environment" [180]. Baldwin and Clark [181] consider designs to be generally "complete" in the sense that the design itself would be completely usable having resolved issues around architecture and interface, and would be presented as a black-box to be used. However, in cases where environments are changing, or there is no clear identification of the boundary between the inner and outer environments, and such an "absolute" design would no longer suffice. Designs can be considered as being "both the medium and outcome of action" to handle such situations, thus accepting an incomplete specification of what the design is [182]. Our idea of designs differs from these, as we consider designs to be black-box artifacts, capabilities of tools that are produced in one part of the hiBPM model, and to be used at another part. The designs can differ in their "completeness" (as will be explained in Chapter 6), thus allowing for different forms of hiBPM model configurations.

5.5.9 Plan-Execute Relationship

The concept of "emergent workflow" is introduced in [26] where process definition or process planning is also recognized as part of the process that is being enacted. Having process model templates allows for a model that is geared towards adaption rather than a simple prescription of process execution. The idea of process definition being separate from process enactment is similar to our notions of having planning stages and execute stages, though we consider plan-execute (and others) as change dimensions for process reconfiguration as part of a business process architecture.

In the space of enterprise modeling, ArchiMate has multiple architectural layers (business, application and technology) with the lower service layer contributing to the higher service layers;

the lower layer provides the “primitives”, i.e., building blocks that the higher layer arranges into services. Two relationships that cross these layered boundaries are the serving relationship that “serves” to the upper layer functions, whereas the realization relationship indicates a realizing of data objects and application components [44]. The plan-execute and design-use relationships in hiBPM differ from these ArchiMate relationships as they provide reasoning about how the plan or the design came about (thus allowing contemplation of alternative configurations) as opposed to being a simple service relationship. Additionally, the plan-execute can be thought of as being within an ArchiMate architectural layer as it allows for rationalizing why and how an ArchiMate artefact is to be built in a certain way, with design-use being across architectural layers where the lower layer builds the design (including abstracting out the building of the design) from the layer above that uses this design.

5.5.10 Sense and Control Relationships

The concept of using feedback loops for adaptation is not new. System dynamics is a feedback-oriented approach for modeling complex systems and has been applied to a wide range of areas, including managerial decision making and organizational behaviour [49]. Here, a controller and a target system come together to form a feedback loop. The controller monitors a target system and modifies it if the output deviates from a defined threshold range. This is done by sending an input through a feedback path to the controller in order to alter the controller output in the next iteration cycle. Such a behaviour denotes the adaptation of the system. The design of adaptive software processes is also an area that has benefitted by the application of system dynamics [49] as software processes can be considered to be processes with inputs, outputs and feedback control elements; therefore, the principles of system dynamics can be used for adapting the software process. The feedback paths in software process dynamics [90] are used to assign resources and take corrective actions in the execution of processes, rather than redesigning them at an architectural level. Unlike previous research that considers adaptation loops in business process management (e.g. BPMN), we are mostly interested in emphasizing the information flow back into a higher-order process stage rather than the iterative executions of the same activities. The hiBPM model supports the visualization and analysis of these feedback paths, including deciding on where the paths should originate and terminate.

5.6 Conclusion

Any business domain contains multiple interrelated business processes, referred to as a process architecture. Our notion of process architecture as part of the hiBPM framework differs from other definitions of process architecture as we consider hiBPM process architectures to comprise of a multitude of business processes that work coherently, to achieve some common objective, and can be grouped for analysis. Such a hiBPM process architecture depicts relationships between multiple business processes that exist in a domain while abstracting away from process-level details. Therefore, the hiBPM architectural description will need to refer only to certain selected elements from process specifications, and not the complete sequence of steps, control flows, data flows, etc. Here detailed specifications are avoided in favour of highlighting relationships and process aspects that facilitate enterprise transformation while considering suitable trade-offs. This is done through abstraction and aggregation of activity units into different process segments types. The concept of studying a collection of abstract processes is similar to that of software systems architecture or enterprise architecture where software design or enterprise design is reviewed and studied at a higher level of abstraction to permit reasoning about design decisions. The emphasis here is on the major elements in the business process collection, including their relationships, while avoiding over-specification.

We introduce some concepts of the hiBPM modeling framework in this chapter to realize the objective of being able to guide the analysis and optimum determination of business processes configuration in evolving enterprises. Unlike traditional process-oriented modeling techniques, the purpose of this framework is not to depict the (sequential) execution of activities, information and data flows, and inclusion of software artifacts in the process structure. Instead, it is developed for modeling multiple business processes and their relationships, and how these relationships and structures can change, thus focusing on the process architecture for enterprises. In Chapter 6, we explain how the concepts introduced in this chapter can be used for representing alternative design configurations of hiBPM models, and provide a means for performing tradeoff analysis for an alternative selection.

6 Analyzing and Reconfiguring hiBPM Models

There are vital factors to consider when designing enterprise processes. Firstly, there may be insufficient information about the conditions under which these processes would execute. Secondly, even if there is sufficient clarity when designing the process stages, over time conditions may change, and it would be difficult to foresee all possible conditions that may result in the future. Therefore, when designing the business processes, there is always a degree of uncertainty in the environment in which the processes are expected to operate. There has to be flexibility in the design of enterprise processes to accommodate this uncertain environment. As a result, managing uncertainty is an essential idea in the design of flexible enterprises; the more uncertain the environment, the higher the degree of flexibility required.

Introducing flexibilities in an enterprise process architecture in response to evolving circumstances may sound like a good idea, but this comes at the expense of adding design complexity and the need for extra resources and capabilities to deal with these changing conditions. Additionally, it may not be possible to deal with all possible circumstances because of constraints, rigidities and barriers in the process architecture design - these would be expressed as various non-functional requirements that need to be maintained while considering the flexibilities in how the process is to be designed. Analysis of these non-functional requirements is thus important as they influence an enterprise's ability to be flexible and adaptable and take on suitable transformations as the situation demands. A natural solution would be to have a partial state of flexibility at key points in the hiBPM model.

In Chapter 5, we introduced hiBPM modeling notations to visually depict a static process architecture for an enterprise. In this chapter, we provide additional details on how the hiBPM framework can support expressing different design configurations to enable reasoning and analysis of the trade-offs between alternative designs for meeting enterprise functional and non-functional goals. These alternative designs can be exercised in situations where the enterprise objectives may not be attained anymore due to evolving circumstances. As there would be an associated cost to implementing these alternative configurations, the hiBPM framework provides mechanisms for considering trade-offs between the benefit and value of retaining these configurations.

6.1 A Design Space for Reconfiguring hiBPM Models

As mentioned previously, there is often a cost of accommodating uncertainty in the enterprise as multiple hiBPM model configurations need to be designed and maintained. If this cost cannot be justified or deemed to be unacceptable, then the enterprise could be designed in a manner that is optimized for efficiency but is unable to handle new conditions that may emerge in the future. Conversely, having highly flexible enterprises will introduce significant design overheads for supporting these flexibilities. This overhead could be in the form of additional processes to support alternative configurations and require adding new capabilities and resources for managing the different alternatives that exist. These overheads need to be kept ready even if they are not deployed or used, which results in unnecessary complexity and cost. hiBPM models should, thus, be designed with trade-offs being considered between the need for accommodating uncertainty and simplicity while minimizing rigidity and overhead.

Through the analysis of the hiBPM model, different points in the process architecture are systematically evaluated and the degree of flexibility is determined that should exist to deal with uncertain circumstances. Such possible alternative hiBPM model configurations are represented through variation points. A variation point has an associated objective and several variants for achieving them. By identifying and focusing on variation points, the possible alternatives to the hiBPM model design in the overall domain setting can be identified. The optimum variant is selected by evaluating their satisfaction with both enterprise functional and non-functional goals.

Uncertainty is attained by leaving these variation points “open” to the selection of different variants at some future point. By this we mean that the process architect or enterprise architect can select an alternative variant at that variation point. Conversely, the rigidity of design is accomplished by selecting a variant after performing a suitable trade-off analysis. A variation point is bound (or closed) when one of the possible variants at that location is chosen after appropriate analysis and reasoning. As enterprise conditions can change, the outcome of the analysis may change over time. The concept of binding variation points is similar to what exists in the software engineering discipline. Binding variation points in both software systems and software processes is discussed in greater detail in Chapter 2 of this thesis, as part of the literature review.

The design space for the hiBPM model consists of not just where the variants exist, including what kinds of variants, but also the conditions under which a variant would be selected and bound. Being aware of the space of design options (and the trade-offs involved in their selection) allows for analysis and selection by taking explicit decision criteria into account. Determining suitable variants, and when and where a variation point becomes bound to a variant is the basis of this chapter. Through analyzing the hiBPM model, the process architect can determine variants that fulfills enterprise requirements at certain points-in-time.

Once a suitable variant has been selected, an alternative state of the hiBPM model at a particular variation point is represented. Through these localized evaluation of variation points, we develop a to-be hiBPM model with the individual variation point changes providing instruction on how to attain that to-be state. The difference between the as-is and the to-be states can then be reviewed, with the process architects and enterprise architects determining the steps required from move from the as-is to the to-be state. In the design space, the hiBPM model could be reconfigured along several dimensions of change. There could be reconfigurability with regards to the placement of various structural elements, or reconfigurability in the relationships between these structural elements.

Different types of analysis can be performed through the hiBPM framework. There may often be insufficient information to pre-plan processes fully before execution and hiBPM models can allow analyzing configurations where activities can be deferred until context-dependent information is available. The relationship between a process that is building a design and the process that uses it can also be expressed using hiBPM, including considering the trade-offs that need to be considered if activities are moved from the former to the latter (and vide versa). These, and other types of analysis, are discussed in greater detail in the next sections.

6.2 Using Goal Models for Analysis

In the previous section, we discussed having multiple variants at a variation point and their binding at a suitable (future) point-in-time. How to compare variants, and select a suitable one, was not covered at that time. In this section, we discuss how goal modeling techniques can be used to perform such an analysis.

Goal Modeling [171] approaches are used to represent and analyze alternative configuration options systematically. In the discipline of Requirements Engineering, goal modeling based techniques are commonly used to capture software system development objectives and stakeholder interests. The functional objectives of the enterprise are represented as goals, whereas non-functional objectives are represented as softgoals. Using a goal model provides an intentional perspective while capturing the objective of the structural element as a goal. We use existing goal modeling techniques to provide the means to understand, analyze and guide possible configurations in the hiBPM model that help satisfy both enterprise functional and non-functional objectives.

Goal models are used to help enterprise architects and process architects analyze alternative options that can exist at a variation point. Alternative configurations of the hiBPM model are, therefore, different ways of respectively modifying or implementing the process architecture. A goal model permits such an analysis showing possible ways of configuring the process architecture for the domain under study. Here, the goals are linked to process actions, each of which achieves a certain functional goal. Reconfigurations to attain functional goals can be shown as multiple choices using OR decompositions. Different structural configurations of the hiBPM model should still result in the attainment of the goals but may result in the non-satisfaction of the softgoals. Conversely, the non-attainment of softgoals may result in the process architects and enterprise architects to propose an alternative hiBPM model configuration. These architects can evaluate these alternative configurations using simple qualitative analysis based on the contribution values of the alternatives against softgoals. Such a selection is made based on the positive or negative contribution(s) that the alternative would have on the softgoals. Evaluating the satisficing of softgoals is a much-studied area in conceptual modeling and requirements engineering, and several techniques are discussed in [172].

We show how a goal model is used to depict multiple alternative ways of attaining a goal at a variation point in Fig. 6-1, including the contribution of each against the applicable quality objectives (shown as softgoals) at that variation point. Here **Goal** is the root goal that needs to be attained at a particular variation point. There are two means of satisfying this goal, through $\mathbf{Goal_A}$ or $\mathbf{Goal_B}$, shown as OR decompositions for each of the alternative variant possible. To go one level

lower, **$Goal_A$** can be satisfied through further sub-goals, **$Goal_{AA}$**, **$Goal_{AB}$** or **$Goal_{AC}$**. The softgoals are also shown, as **$Softgoal_A$** and **$Softgoal_B$**, with their evaluation being done using contribution links. In the hiBPM framework, we only do a qualitative evaluation of softgoals using different contribution links, such as help+/hurt–, make+/break-, some+/some- etc.

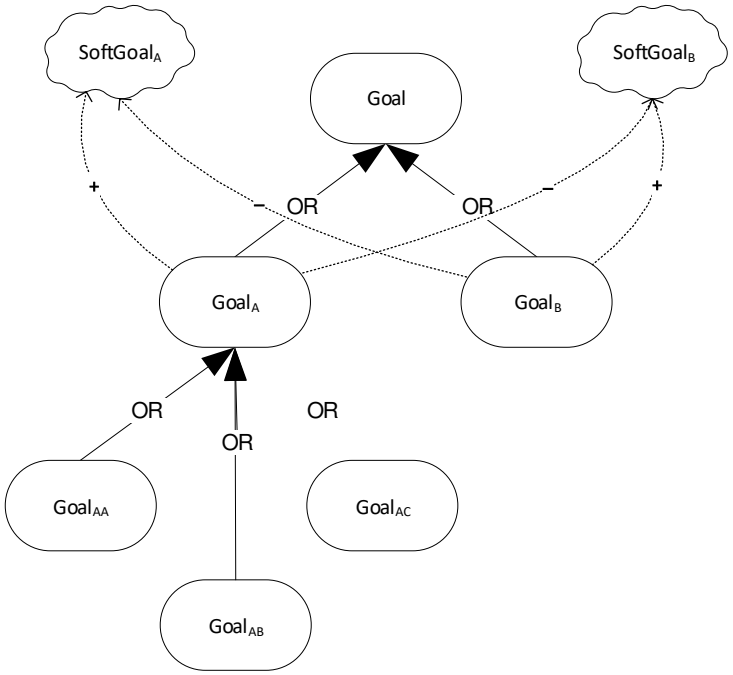

Fig. 6-1: Using Goal Models for analyzing hiBPM model alternatives

Such a concept of associating goal models and process models is not unique and has been previously considered in [188][189]. Our work differs from these as we consider associating the process stages with goals or softgoals (as determined from the goal models), as these process stages indicate some accomplishment of enterprise functional or non-functional objectives. As hiBPM is not a business process modeling notation, hiBPM models are not meant to be complete in their depiction of the domain. Therefore, there is no expectation that all leaf-nodes in the goal model would appear as corresponding process structural elements in the hiBPM model. Rather, the goal models are able to provide a starting structure for creating the hiBPM model, while also helping populate the internals of these process stages.

In Chapter 5, we introduced various modeling constructs for the hiBPM model. However, at that time we did not provide details on how hiBPM models can be reconfigured. We discuss hiBPM model reconfigurations in subsequent sections of this chapter.

6.3 Reconfiguring Structural Elements

There exists several ways of manipulating the structural elements of the hiBPM model for achieving model reconfigurability which involves the relative placing and movement of various structural elements within the hiBPM model. Some of these possible combinations are presented in Table 6 below and are discussed in more detail in the following sub-sections.

Table 6: The need and effect of reconfiguring structural elements

Reconfiguration	Need for Reconfiguration	Example Effect of Reconfiguration on Softgoals
Adding process elements to process stages	Process elements are added to the process stage to indicate the additional activities that need to be performed to attain softgoals, to introduce varition points, or to provide preciseness to the means for goal attainment.	Increase: Cost, Preciseness Decrease: Speed
Removing process elements from process stages	Process elements can be removed from process stage if they do not contribute to either softgoal attainment, or if they do not lend to the hiBPM analysis.	Increase: Speed, Simplicity Decrease: Cost
Splitting process stages	A process stage can be split if there are multiple goals and softgoals that are being attained by the process stage or if the ouput of some of the process elements need to be available until after the process elements complete execution.	Increase: Cost, Reusability Decrease: Rigidity, Simplicity
Merging process stages	Process stages can be merged together if they collectively contribute to the same goals or softgoals and if the output of individual process stages does not need to be separately made available.	Increase: Rigidity, Simplicity Decrease: Cost, Reusability
Converting process phases to process stages	A process phase can be converted to a process stage if the output of the process phase needs to be made available after the process phrase execution and the process phase can independently contribute towards goal or softgoal attainment.	Increase: Repeatability Decrease: Simplicty
Converting process stages to process phases	A process stage can be converted into a process phase and merged into an existing process stage if there is no need to make available the ouput of the process stage and there is no sequential dependency within a process phase.	Increase: Simplicity Decrease: Repeatability

6.3.1 Adding and Removing Process Elements

Process elements may be added to process stages and process phases or be eliminated from them altogether. Adding process elements to a process stage has the advantage of bringing into additional focus activities that need to be performed (or activities that need to be evaluated) during the overall analysis of the hiBPM model. We show a process stage, **Manage Requirements**, in Fig. 6-2(A), that has three process elements. Another process element, **Publish Specifications**, is added to this process stage, presented in Fig. 6-2(B). In this situation, the addition of this process element results in a modified output, **Published Specifications**.

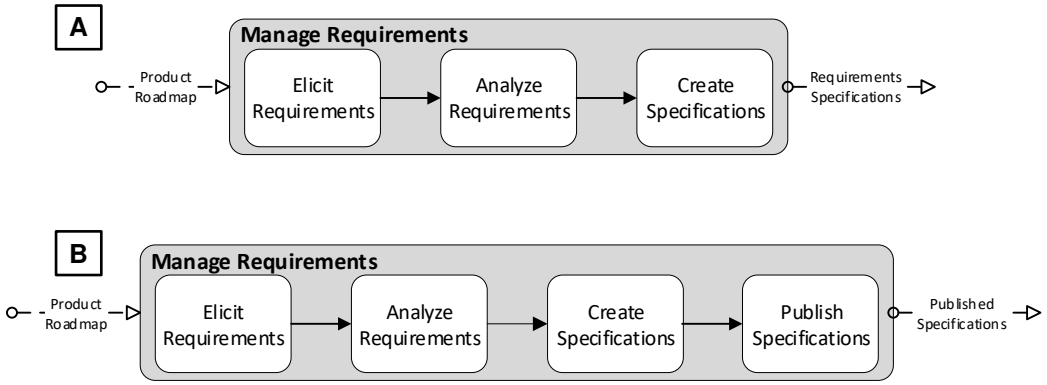

Fig. 6-2: (A) A process stage with three process elements, (B) Adding a new process element to the same process stage

Removing process elements can indicate that either the activity no longer has a consequence in the analysis or that the execution of the process phase or process stage needs now to be changed that requires the elimination of the process element. This could be necessary when the cost or time of process stage execution needs to be reduced. We present this example in Fig. 6-3 where a process element, **Analyze Requirements**, is removed from the **Manage Requirements** process stage to signify that there is no need to analyze the determined requirements, presumably because there is no value in doing so and eliminating this process element does not result a change in the output of the process stage.

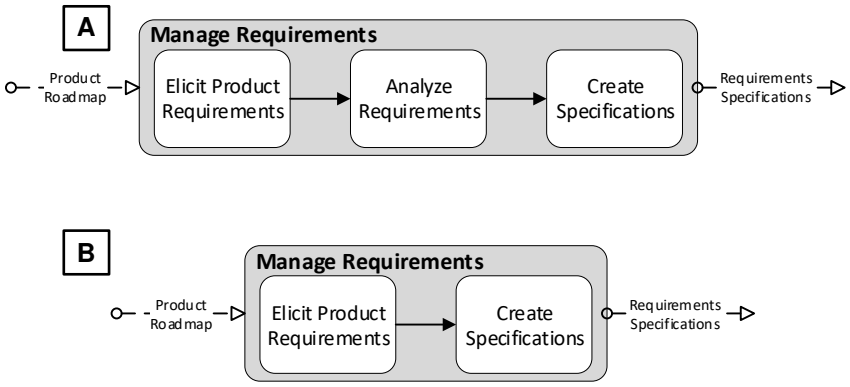

Fig. 6-3: (A) A process stage with three process elements, (B) The same process stage with a process element removed

6.3.2 Merging and Splitting Process Stages

Process stages may be created or eliminated because of changes in the process stage recurrence, or because process stages may evolve to have different objectives to fulfill. This results in a process stage being split into two or a process stage may be merged into another one.

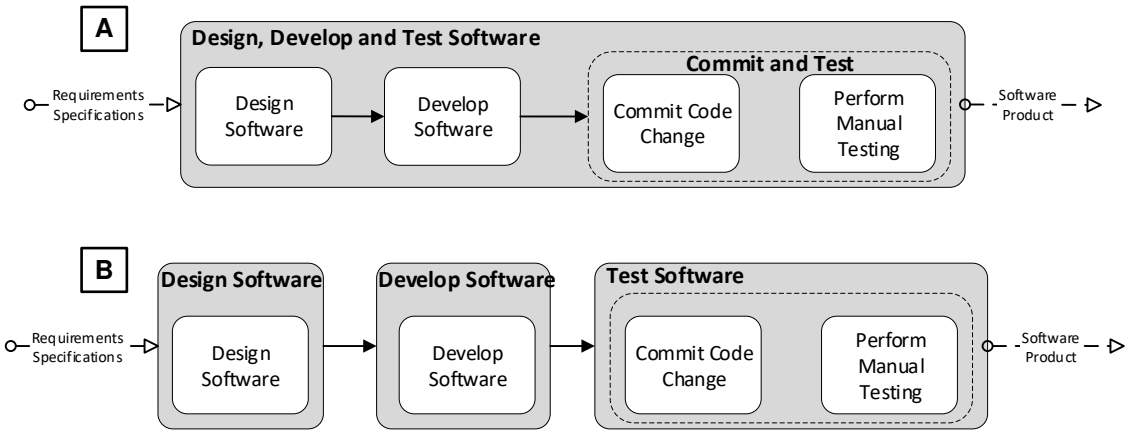

Fig. 6-4: (A) A process stage with several process elements and a process phase, (B) The same process stage being split into three distinct process stages

In Fig. 6-4(A) several process elements in the **Design, Develop and Test Software** process stage were all contributing to single goal attainment, that of producing the **Software Product**. If there was a need to retain the outputs of process elements or collections of process elements that are present within the **Design, Develop and Test Software** process stage, then it would be better to split this single process stage by having the process elements being part of a new process stage or the

process phases being split off into separate process stages. Such a case is shown in Fig. 6-4(B) where we now have three process stages, **Design Software, Develop Software,** and **Test Software,** each with their own outputs.

In another case, all process elements and process phases within a process stage have the same recurrence. If there were changes that were to be made with regards to the frequency of executing these process elements and process phases, then it would necessitate splitting the process stage into two or more process stages. Splitting process stages enables greater flexibility through introducing recurrence relationships or by retaining the outputs of the split process stages but introduces complexity as the enterprise architects now need to plan for managing these process stage outputs. Conversely, merging multiple process stages into a single process stage can be considered when the intermediate outputs are not needed or there is no need to consider recurrence between process stages.

6.3.3 Converting Process Phases into Process Stages

As mentioned in Chapter 5, a process stage may have multiple process phases. The reason such a composition may exist is because (a) the process phases conjunctively work together to accomplish the process stage objective and the exact sequence of process element execution is not relevant within those process phases. As in the previous case, a situation may emerge where these process phases are no longer collectively working towards a single goal accomplishment and may now be responsible for goals or softgoals that are independent of each other. Also, there may be specific types of relations that need to be explicated (such as recurrence) or specific types of outputs (designs or plans) that are to be highlighted for process phases. In such a case, it would be better to have the process phases as separate process stages, as shown in Fig. 6-5. Here the **Design, Develop and Test Software** process stage is split into two by converting the **Commit and Test** process phase into a separate process stage, **Test Software**. Through this separation, we can signify that each process stage is responsible for a specific output, i.e., an unvalidated **Software Product** in the case of the **Design and Develop Software**, and a validated **Software Product** in the case of **Test Software**.

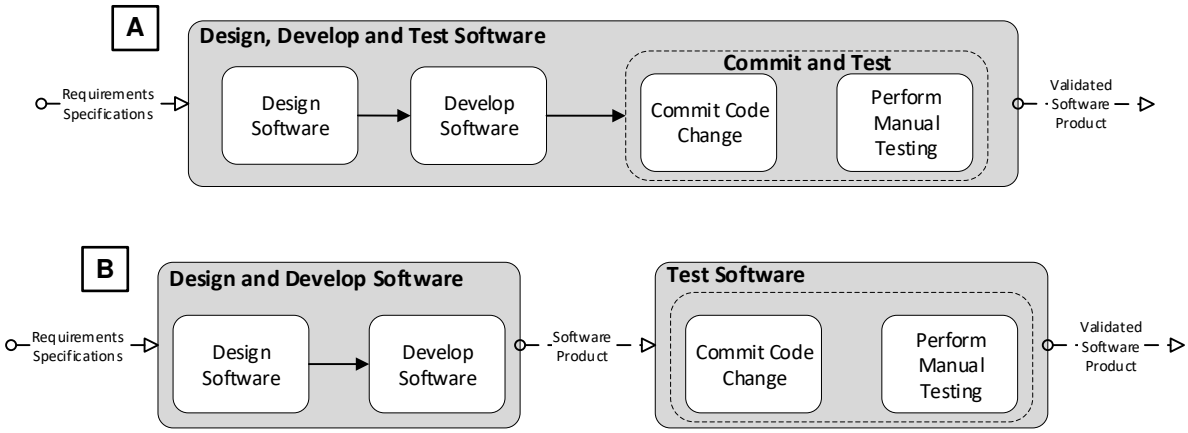

Fig. 6-5: (A) A process stage containing several process elements and a process phase, (B) The process phrase being separated into a separate process stage

The reverse case would also be true where if the enterprise doesn't need to maintain the intermediate unvalidated **Software Product**, then it can combine the process phase into a single process stage.

6.3.4 Moving Process Stages and Process Elements across Process Boundaries

Structural elements may move across process boundaries. Such boundaries exist across recurrence relationships, design-use relationships and plan-execute relationships. A movement of process elements or process stages across process boundaries adopts the traits (whether recurrence, design objective or planning objective) of structural elements on the other side of the process boundary, thus affecting the outcome of the process execution, or the quality of decisions made. For example, in Fig. 6-6, we show process stages across a recurrence boundary. We specify look at the **Design Feature** process element that is part of the **Perform Sprint Cycle** process stage. This process element can be placed in either the lower recurrence process stage (**Design Software**) or the higher recurrence process stage (**Perform Sprint Cycle**). Based on either of these hiBPM model configurations, the **Design Feature** process element would correspondingly adopt the recurrence attribute of that process stage. A more detailed understanding of why such movements are useful is provided in the sub-sections on recurrence, design-use and plan-execute in this Chapter.

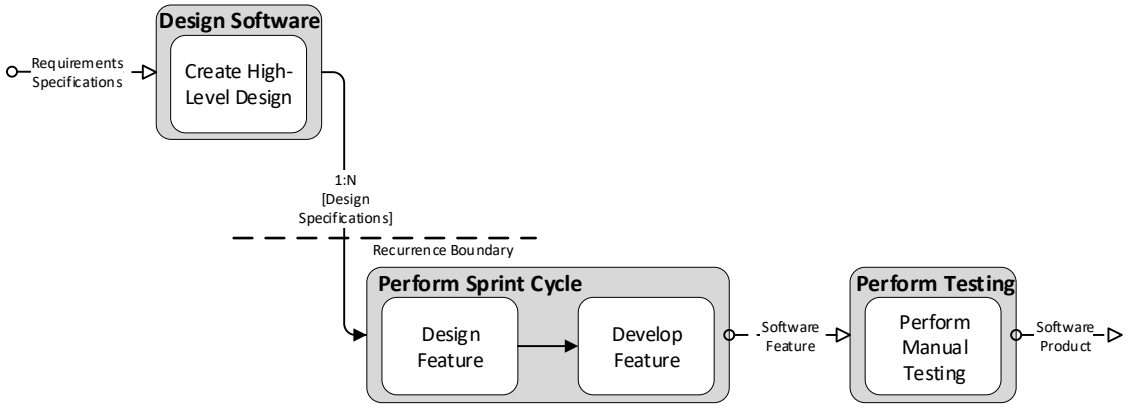

Fig. 6-6: Several process stages at different recurrences that span a recurrence boundary.

6.4 Reconfiguring across Temporal Placements

In the temporal reconfigurability dimension, there are multiple possible temporal placements for structural elements that achieve the same functional objective but are different in terms of their non-functional characteristics. A particular temporal placement of a structural element can bring about certain benefits, depending on whether it is advanced (and be executed) before other structural elements or postponed after those structural elements. Therefore, in hiBPM, a consideration is to not just determine the structural elements (and their relationship to other structural elements), but also where they should be placed within the hiBPM model.

The degree of impact due to permitted temporal movements may vary between domains. By permitted temporal movements, we mean those placements of structural elements within the hiBPM model that do not violate the constraints placed due to the input data not being available or prior sequential processing not having occurred. Such movements would have virtually no improvement in flexibility and other non-functional goals in stable domains but may result in very significant benefits in highly dynamic, volatile domains.

Let's consider some examples of how the temporal placement of structural elements would work,

- *Advancing* a structural element relative to other process elements reduces complexity and cost, as less effort is required to process the limited contextual information available at that instant. By advancement we mean moving a structural element within a hiBPM model so that it now

executes before elements that it previously used to execute after. Advancement provides for stability and uniformity and can be enabled by either settling on coarser-grained process elements that rely on less information and thus can tolerate uncertainty or by better predictions of the currently missing information.

- *Postponing* a structural element provides the benefit of executing it with the latest context and information available, thus reducing the risk and uncertainty that are inherent in any software process. By postponing we mean moving a structural element within the hiBPM model so that it now executes after elements that it previously used to execute before. Here there is an assumption that better, more precise information will be available at a later point in the hiBPM model. Postponing the structural element would allow for better execution outcomes.

Table 7: The need and effect of reconfiguring across temporal placements

Reconfiguration	Need for Reconfiguration	Example Effect of Reconfiguration on Softgoals
Postposing process elements within a process stage	A process element can be postponed for execution later in a process stage if the process element relies on the outcome of prior process element execution or if the necessary context is not available.	Increase: Context-Awareness Decrease: Stability
Advancing process elements within a processs stage	A process element can be advanced within a process stage for execution as early as permissible if the necessary context is available and there are no sequential dependencies on this process element.	Increase: Stability Decrease: Context-Awareness
Moving process elements to a later process stages	A process element can be postponed to a downstream process stage if the condition for delaying execution is present and the movement to a later process stage would lead to satisficing of softgoals for this process stage.	Increase: Context-Awareness, Repeatability Decrease: Stability
Moving process element to earlier process stages	A process element can be advanced to an upstream process stage if the condition for advancing is present and the movement to an earlier process stage would lead to satisficing of softgoals for this process stage.	Increase: Stability Decrease: Context-Awareness, Repeatability
Moving process element into process phases	A process element can be moved into a process phase if the sequential execution of the process element within that process stage is not of consequence and to simplify the analysis of the process architecture.	Increase: Simplicity, Efficiency Decrease: Rigidity, Customizability
Moving process element out of process phrases	A process element should be moved out of a process phase if there exists sequential dependency on that process element and the positioning of the process element within the process stage helps with the analysis of the process architecture.	Increase: Rigidity, Customizability Decrease: Simplicity, Efficiency

Advancement and postponement are a well-known approach in several domains. In business strategy, the aim is to minimize risks and maximize benefits by delaying some activities or decisions that require precise, up-to-date information until later. In variability in software, key components are introduced in a software system that can process data received in real-time and adapt accordingly. Even recent innovations in software development practices (such as Agile), the emphasis is to defer key development activities till the point where appropriate data is available to be able to start performing the development task required. The inverse is true for advancement for each of the domains above.

With a hiBPM model there may be several possible ways of moving around structural elements from a temporal perspective. We discuss these below.

6.4.1 Moving Process Elements within a Process Stage

Within a process stage, a process element could be moved earlier or later to other process elements, as shown in Fig. 6-7. The output of the process stage would not change; however, how the process stage is executed would change. One reason for such movements would be to induce a change in how softgoals are met. The process element, **Perform Manual Testing**, can be executed after the **Commit Code Change** or before. In both these configurations, the output of the **Design, Develop and Test Software** process stage still remains the same as the process stage still comprises of the same process elements, but how the process stage goals are attained may change resulting in the satisficing of certain softgoals over others.

The configuration shown in Fig. 6-7(A) creates a separation between software engineers and test engineers where the software engineers first commit their code into a code repository which test engineers then retrieve from. This is a more methodological way of working but comes at the expense of agility of development operations. The configuration shown in Fig. 6-7(B) results in software developers and test engineers working together to validate the developed code before it is committed into the code repository. Any reported issues in the code can be quickly fixed through this collaborative effort. Such a configuration would be preferred by an enterprise that values agility over structured operations.

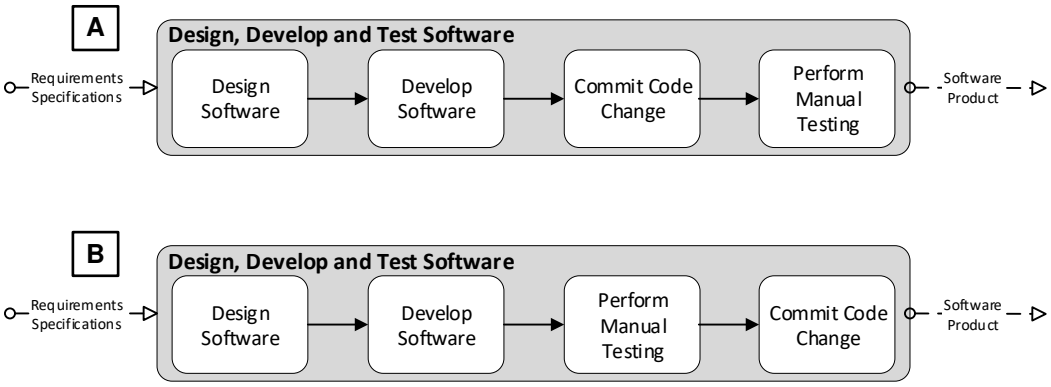

Fig. 6-7: (A) A process stage with several process elements, (B) The same process stage with the sequence of process elements modified

6.4.2 Moving Process Elements Across Process Stages

A process element may move amongst process stages by either placing it in an upstream process stage or a downstream process stage. In such situations the output of the process stage would change, as illustrated in Fig. 6-8.

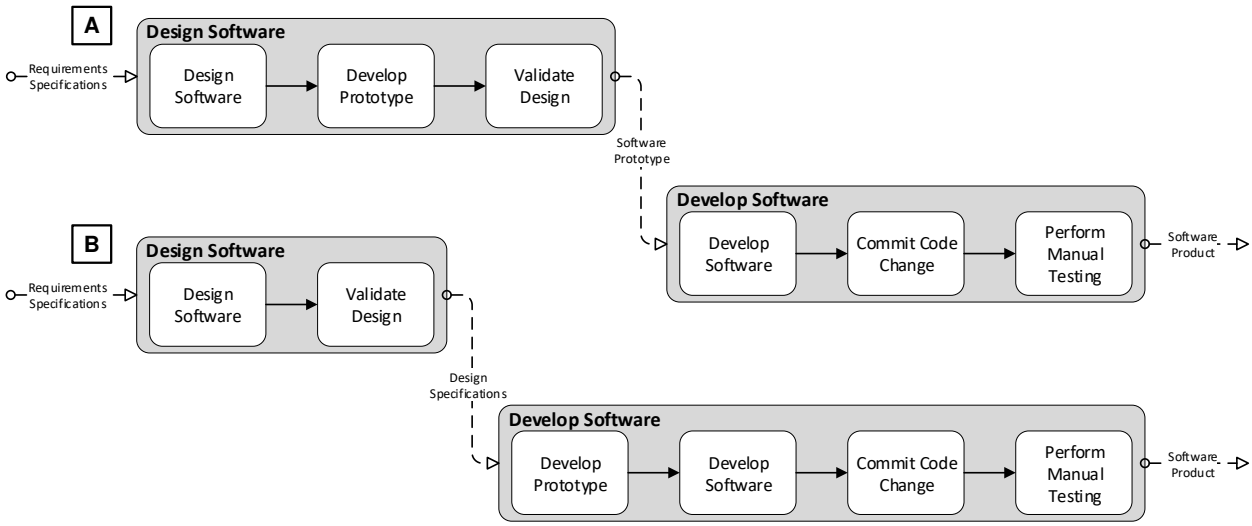

Fig. 6-8: (A) Two process stages connected through a sequence flow, (B) Same process stages but with a process element moved from one process stage to another

The design of a software can be prepared in the form of **Design Specifications**, as shown in Fig. 6-8(B), or as a working **Software Prototype** that is a manifestation of a validated design. Either one of these outputs can be attained by the placement of the **Develop Prototype** process element. If it

is placed in the upstream **Design Software** process stage (Fig. 6-8(A)) then the process stage produces a validated design in the form of a **Software Prototype**. If the process element is placed in the downstream **Develop Software** process stage (Fig. 6-8(B)) then the upstream process stage produces the **Design Specifications**.

Moving a process element to an upstream process stage would be useful if the decisions made and actions performed in that upstream process stage can be reused in multiple other process stages. It would therefore make sense to put these in a process stage that feeds into several process stages, thus saving time and cost during the execution of the initial process stage. An assumption here is that moving the process elements to this new process stage would still have available the necessary inputs required for successful process execution. Such an alternative hiBPM configuration, of course, brings about additional complexity by possibly introducing new process stages or changing the data flows, and the trade-offs to this new configuration would need to be carefully assessed.

6.4.3 Moving Process Elements across Process Phases

A similar situation would be for moving process elements across process phases, as shown in Fig. 6-9. The **Commit Code Change** process element moves within the process stage **Design, Develop and Test Software** to the **Test Software** process phase. This is to signify that the execution order of the process element is immaterial for the analysis of the model, and the model can be simplified by placing it alongside other process elements within the **Test Software** process phase.

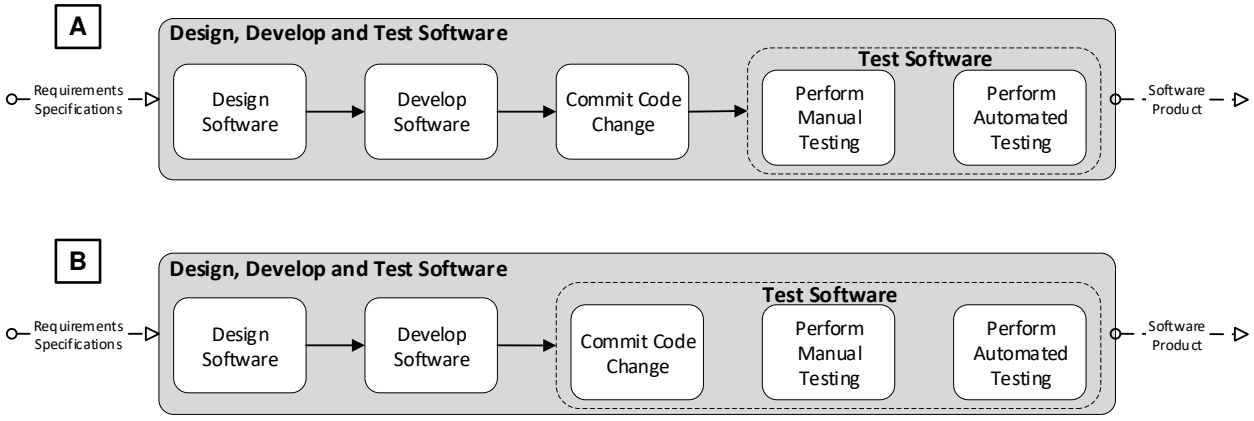

Fig. 6-9: (A) A process stage with several process elements and a process phase, (B) Moving a process element to a process phase within the same process stage

As before, moving process elements from one phase to the next may result in changes to softgoal(s) attainment for the process stage, including the quality of decisions made and the outcome of actions performed. However, the goals of the process stage would not change as overall the process stage still accepts the same inputs and generates the same output. From a scenario perspective, this is not so much different from moving process elements within a process stage, the only distinction here is to explicitly imply that the exact temporal placement with a process phase is not important.

6.4.4 Moving Process Elements within a Process Phase

Finally, it is important to discuss one more possible combination in the temporal change dimension, i.e., moving a process element within a process phase. This would have no difference with regards to the output of the process phase. The benefit of introducing the notion of process phases in hiBPM was to eliminate such a case by reducing the possible number of process element placement alternatives within a process stage, thus simplifying the analysis that needed to be performed. The process elements, **Commit Code Change** and **Perform Manual Testing** within the process phase **Test Software**, shown in Fig. 6-10, do not need to have a sequence flow associated with them as changing their execution order doesn't result in any change in the output of the process phase.

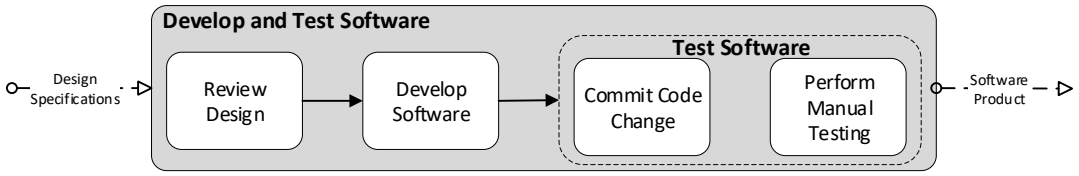

Fig. 6-10: Moving process elements within a process phase

6.5 Reconfiguring User Engagements

In user engagement reconfigurability, a large space of possible options for how users engage with information systems needs to be considered, with a variety of factors and the complexity of the domain to be taken into consideration. One should not expect to be able to "design" a solution where users engage through static business processes and it will just "work", as there may be many unknowns.

From an *enterprise* and *user* perspective, users are engaging with the new types of information systems being introduced while dealing with real-world business situations. Both sides (i.e., the users and the systems) need to adapt and adjust to each other and eventually converge to a workable state; the users learning to execute their assigned business processes while the systems undergoing cycles of iterative improvements to make them significantly more efficient and intelligent. Factors affecting the adoption success (of the systems by the users) may include the knowledge/skills of involved personnel, their aptitude for understanding the information systems' capabilities and limitations as well as their trust in such systems, willingness to learn and adapt, attitude towards and trust in automation in general, labour relations, reward structures, business domain regulations, etc.

From a *systems* perspective, how an information systems solution is accepted in an enterprise can be very specific to the situation in that organization. Nevertheless, even this situation will continue to evolve as the software systems get better or acquire new features. Employees would also gain experience or learn new skills on the side of the user organization. Thus, information systems should be capable of supporting a variety of process architecture configurations, with their roles ranging from assistive, to advisory, to complete responsibility for decision making. Designers of information systems cannot be expected to predict or prescribe exactly how the human side is going to use these systems.

From a *process* perspective, the impact of process-level user engagement is not limited to just direct system interactions but includes the related processes as well. By related (or surrounding) processes, we mean upstream processes that contribute in some way to the primary business process or downstream processes that benefit from the output of the business process. These surrounding processes too evolve in response to changes in information systems' capabilities. Thus, multiple processes need to be considered for analyzing and designing user engagement. Some of these processes may operate at the transactional level while others may execute infrequently.

These configurations need to evolve together with the changes in the above parameters as well as due to the feedback reflecting on how they meet their objectives. Thus, the relationship between

the processes where the systems are built, and where they are used cannot be static and needs to be managed concurrently with business processes. As explained in Chapter 5, we refer to these as user engagements. Therefore, we need to be able to characterize the space of alternative user engagement configurations reflecting the whole spectrum of such configurations for a given decision, their potential combinations, frequency and scope of their execution, and context, among other things. There would be transitions across these configurations due to changing enterprise requirements, contexts, etc. Finally, changes in these user engagements frequently affect related processes and give rise to new processes.

There can be multiple user engagement modes (UEM) at the **Manage Monthly Budget** process element. In Fig. 6-11, we show a user engagement between two process stages, **Select User Engagement** and **Manage Monthly Expenses**, that allows the process element to transition from one user engagement mode to another. Over time, the engagement between the system and the user may change based on evolving situation. This change is initiated based on the plan, **Selected UEM**, that is provided by the **Select User Engagement** process stage. **Select User Engagement** decides on transitioning to another user engagement mode based on the input being provided to it, and by the feedback **User and System Activity** that is the output of the **Manage Monthly Expense**. This output would indicate whether a specific state has been reached (e.g. where the user has high confidence in the outcome of the **Manage Monthly Expense** process stage), and there needs to be a transition to another user engagement mode.

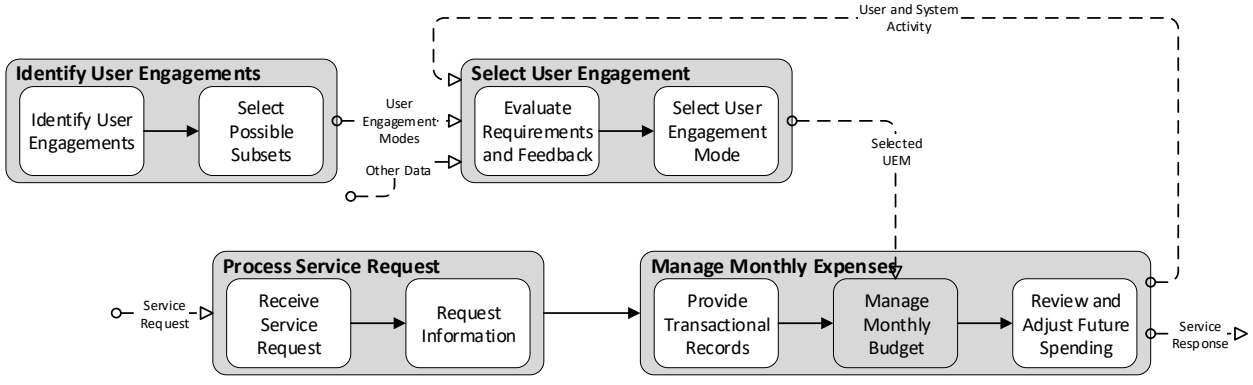

Fig. 6-11: Reconfigurability user engagement through selection of user engagement modes

Indicating user engagements in hiBPM models is essential as user attitudes towards automation in general, and the system they are interacting with, in particular, can change based on the accumulated history of these engagements and the quality of the output the system produces. Shifts in work allocation between both sides can be envisioned, which are triggered by well-defined conditions that focus on system performance, users' trust in systems' recommendations, and other main characteristics of user engagement. This engagement is not static. It varies over time-based on not just the evolving capabilities of enterprise systems but also on user requirements and enterprise context and objectives. This is not solely a design-time problem and needs to be continually addressed at runtime as and when changes are incorporated and implemented. There is also a lack of familiarity with how those adjustments and changes are going to happen in the future. As a result, a space of possible options for user engagement with systems would need to be considered, with a variety of factors and the complexity of the domain to be taken into consideration. By separately denoting such user engagements in hiBPM, we can indicate where the changes are to be introduced, with the possibility of having other supporting hiBPM changes, as and when such shifts in user engagements happen.

6.6 Reconfiguring Recurrence Relationships

Any enterprise needs to analyze how to configure the recurrence relationship between two adjacent process stages by balancing the cost, complexity and other factors. In the recurrence dimension, there are multiple possible ways in which changes in the recurrence relationship can be used to affect the behaviour and configuration of the hiBPM model. As explained in Chapter 5, within a recurrence relationship, the process stage that executes more frequently is referred to having higher recurrence (and exists at the higher recurrence side of the relationship) and the process stage that executes less frequently is referred to as having lower recurrence (and exists at the lower recurrence side of the relationship).

Table 8: The need and effect of reconfiguring recurrence relationships

Reconfiguration	Need for Reconfiguration	Example Effect of Reconfiguration
Moving process element to higher recurrence	A process element may be moved to a process stage having higher recurrence if the context required for execution of the process element is rapidly changing and the process element can provide improved output by using more recent input data.	Increase: Responsiveness, Context-Sensitivity Decrease: Stability, Efficiency
Moving process element to lower recurrence	A process element may be moved to a process stage having a lower recurrence if either the context required for the execution of the process element is not rapidly changing or if the process element doesn't require a more recent input data for execution.	Increase: Stability, Efficiency Decrease: Responsiveness, Context-Sensitivity
Moving process stage to higher recurrence	A process stage may be moved to a higher recurrence if the input context required for process stage execution changes rapidly and the process stage execution can benefit from a more recent execution for goal satisfaction.	Increase: Responsiveness, Context-Sensitivity Decrease: Stability, Efficiency
Moving process stage to lower recurrence	A process stage may be moved to a lower recurrence if the input context required for process stage execution changes slowly and the process stage execution at a lower recurrence does not result in goal non-satisfaction.	Increase: Stability, Efficiency Decrease: Responsiveness, Context-Sensitivity

Let us consider some examples of how recurrence reconfigurability would work between two process stages that are connected via a recurrence relationship.

6.6.1 Moving Process Elements across Recurrence Boundary

A process element can be moved from a process stage with a higher recurrence to one with a lower recurrence. Such a movement of the process element can change the non-functional properties of the process stage in various ways. We illustrate this using the model presented in Fig. 6-12. Having the process element **Groom Release Backlog** in the process stage **Plan For Release** that is at a lower recurrence saves cost as the same process element does not have to be executed repeatedly. Conversely, moving this process element from the lower recurrence process stage to **Perform Sprint Cycle**, which is at a higher recurrence, has the opposite effect. The cost of the process stage execution would increase, but this movement can assist with flexibility and adaptability as the process element is executed based on updated and current information.

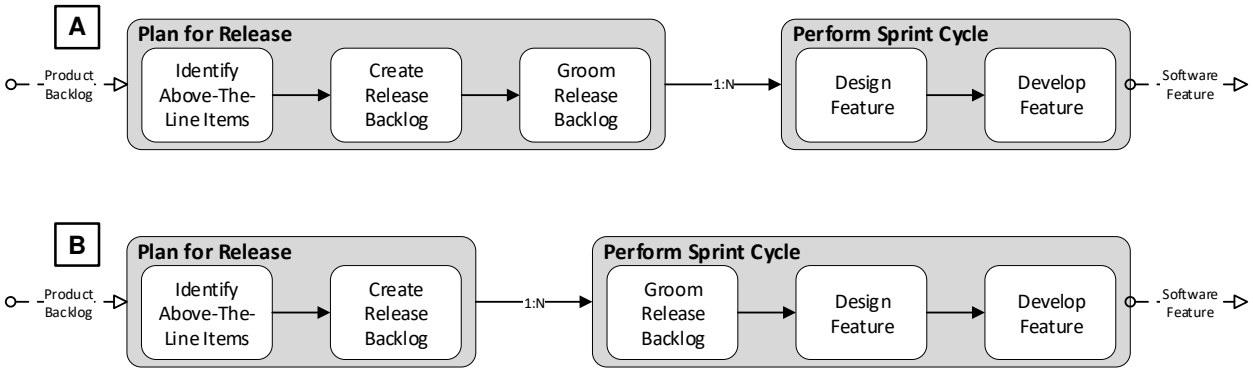

Fig. 6-12: (A) Two process stages connected through a recurrence relationship, (B) Same process stages with a process element moving across the recurrence boundary

6.6.2 Moving Process Stages across Recurrence Boundary

Similar to the previous case is one where the complete process stage is moved across a recurrence boundary. This could be in either direction, i.e., the process stage moves from the lower recurrence side to the higher recurrence side, or the process stage moves from the higher recurrence process stage to the lower recurrence process stage. In both cases, the process stage would adopt the recurrence of the side it has been placed in. Such a movement of the process stage is important as it helps attain different enterprise softgoals.

For example, the enterprise may decide to move from a phase-based software development methodology (such as Waterfall) to one that encompasses an agile mindset (such as Scrum). The hiBPM models for both these methodologies are presented in Fig. 6-13. In Fig. 6-13(A), we show that the **Design Software** and **Develop Software** process stages are performed once, with the resultant output being repeated tested. However, adopting the Scrum methodology requires iterative development. Thus, the **Perform Sprint Cycle** iterates repeatedly, with the high-level **Design Specification** as an input, for developing individual software features which are then tested in the **Test Software** process stage.

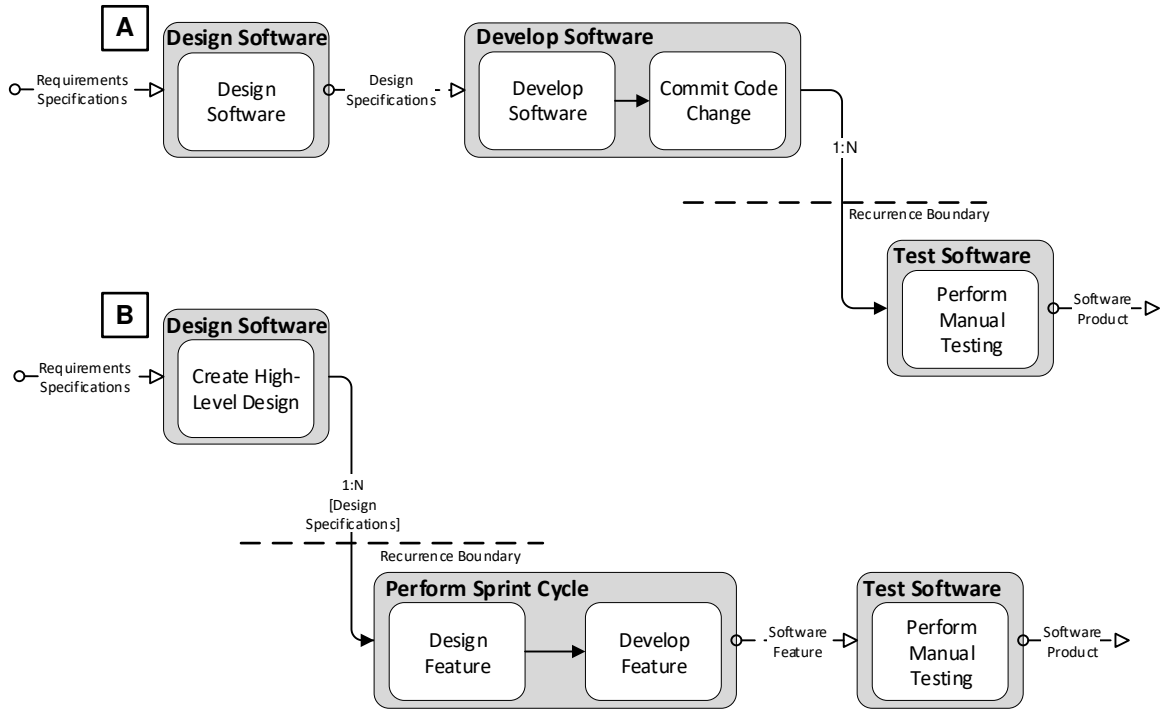

Fig. 6-13: (A) Process stages connected across a recurrence boundary, (B) Moving a process stage across the recurrence boundary resulting in increased recurrence

Based on goal-based analysis, it could be determined that certain process stages do not need to be executed as frequently as the process stage output does not vary significantly; thus, it would be better for them to execute at a reduced frequency. Another situation could be where, through automation, the cost of process stage execution is drastically reduced. Thus, it is possible to execute this process stage at a higher frequency now, as there is no significant added cost to the enterprise. It is obvious that such a temporal movement does not result in changing functional goal attainment, but rather, pertains to how this goal is attained by its impact on softgoals.

6.6.3 Changing Recurrence Relationships

A final case is where the recurrence between two adjacent process stages change relative to each other. In Section 5.4.2, we indicated that two process stages could have different recurrences and represented them as 1:N, N:1, and N:M. The relative recurrence between process stages can shift between these combinations, as dictated by softgoals. We use the same example as that of the

previous use case where an enterprise is transitioning from a phase-based development methodology, presented in Fig. 6-14(A) to a more agile one, presented in Fig. 6-14(B).

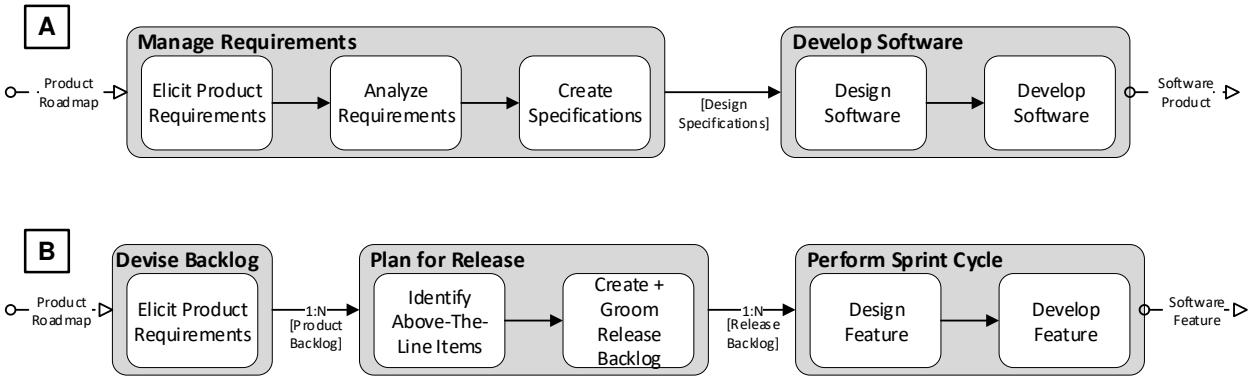

Fig. 6-14: (A) Process stages connected with a sequence flow and no recurrence, (B) Process stages reconfigured with recurrence relationships

It would also be prudent to consider any possible recurrence relationship with the ability of the enterprise to detect changes that are to be meant as inputs to the process stage. E.g., executing a process stage more frequently than the rate at which changes in the domain can be detected will not produce tangible benefits. Such a situation could change if new software systems are introduced that can do better and faster data sensing. In such a case, the recurrence relationship will change. We consider various cases for this below.

- Let us start with the simple case, where the frequency of both the process stages are tied to each other (say in a 1:1 recurrence relationship). The downstream process stage may move to a higher recurrence than the upstream process stage, i.e., to a 1:N recurrence. This may happen because the cost of repeatedly executing the process stage is reduced (through automation) or the need to sense data is urgent (because of improved sensing, data availability, or rapid response) than before.
- The upstream process stage may move to a higher recurrence than the downstream process stage, i.e., to an N:1 recurrence. This may be necessary to re-process or re-calculation some data input or plan that is sent down to the downstream process stage. Having the downstream process stage execute at a lower frequency than an upstream process stage is appropriate when the cost of process execution for the downstream process stage is high or rarely required.

- The recurrence relationship may change to one where the two process stages are operating at different degrees of recurrence (i.e., an M:N recurrence) or one that is on-demand (i.e., where a trigger is directed from the output of one process stage to another). Such an arrangement is accommodated to show that recurrence relationships in hiBPM are not always precise, and a higher degree of reconfigurability is needed that matches real-world situations.

6.7 Reconfiguring Design-Use Relationships

Design-use reconfigurability supports the identification and analysis of flexible design variations of the tool or capability. Generally, the tool or capability being produced are considered rigid in the sense that they are assumed to be developed elsewhere and cannot be modified during use. They are designed for certain functions and their use is not adaptable at run-time. However, evolving enterprises require flexible designs whose use can vary considerably resulting in different business outcomes. These designs need to be considered as evolvable objects, which can be easily redesigned at use time to accommodate changes in the external environments and business or system requirements. Here a focus is on the flexibility of the tool or capability being produced in the design process stage. The more single purpose (less flexible) the tool is, the simpler it is to use and the more optimized it can be, particularly for automating the execution of process stages. For a more flexible design (for supporting usage-time modifiability of the design), the design complexity may increase resulting in additional process overhead.

In design-use relationships, process elements can be placed in the design process stage or a use process stage. In the example presented in Fig. 6-15, we show how design-use reconfiguration is attained by positioning a process element on the design process stage or the use process stage, i.e., whether the process element is invoked as part of a design process, or is invoked during the use of that artifact, tool, or capability that is the outcome of the design. The **Make Environment Template** design process stage produces a complete design **Environment Template** in Fig. 6-15(A). A *complete* design is one where all decisions to be made during process execution have been made. Complete designs facilitate automated tool usage as no decisions need to be made in the use process stage for design use but they introduce rigidities and inflexibilities in the overall domain. This is because the decisions made during tool design may not match the execution context well.

Conversely, *partial designs*, as shown in Fig. 6-15(B), have unresolved design decisions, which are passed on to the use process stage.

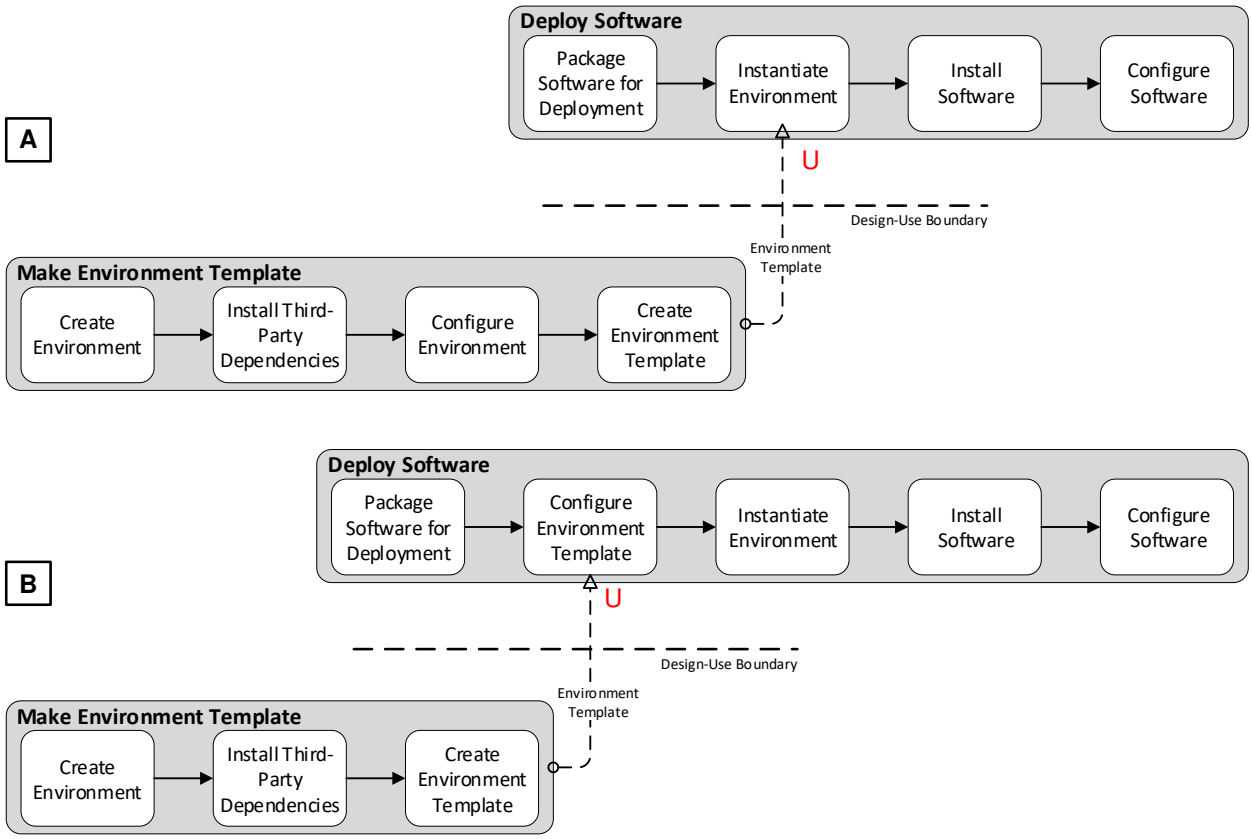

Fig. 6-15: (A) A design-use relationship between two process stages, (B) The same process stages with a process element moved from the design process stage to the use process stage

The **Configure Environment** process element has moved from the design process stage to the use process stage. Thus, the **Environment Template** produced by the **Make Environment Template** is no longer a complete design as the use process stage, **Deploy Software**, have to first configure **Environment Template** before it can be use. Here, we show that the process element responsible for completing the design has been moved to the use process stage.

Positioning a process element in the design process stage leads to increased artifact sophistication (through greater design), whereas placing the process element in the use process stage results in greater run-time customizability of the artifact (through manual control of a simpler design). If the process element is placed on the design process stage, it results in the following changes.

- In case of an activity, this means that the tool takes on more functionality, thus increasing the level of automation in the use process stage.
- In case of a decision, it means that it is bound in the design process stage and becomes fixed in the use process stage, thus reducing the customizability of the produced tool.

On the contrary, moving an activity in the other direction reduces the level of automation available to the use stage, while moving a decision increases the level of customizability of the tool since the decision is no longer built into the tool and can be changed during its use. When analyzing choices in a design-use relationship, we consider other factors in addition to increasing flexibility and evolving design capability. Having a pre-built design that is repeatedly used in other process stages greatly reduces the overall cost as the design is no longer built into the use process stage before it is to be used.

While partial designs may be beneficial when allowing for run-time incorporation and utilization of contextual data, they may reduce the automation of the use process stage because of the need for human involvement prior to processing. There may also be present process stages where there is no clear distinction between design and use activities and all process elements are collectively executed within that process stage. There is no design tool being produced and nothing that a downstream consumption process stage can use. Such a process architecture configuration offers advantages in terms of the preciseness of implementation, but at the expense of formal design and reusability in different contexts.

6.8 Reconfiguring Plan-Execute Relationships

An important consideration for enterprise agility is deciding whether activities should be part of the planning process or to be left later for the executing process. In hiBPM, we allow for the explicit representation of planning activities separately from the execution of these planned activities. The plan-execute reconfiguration dimension supports the identification and analysis of variations of the completeness of plans being produced, through reasoning about the possible placement of a process element on either side of a plan-execute boundary. The main focus is on analyzing how much to pre-plan in the planning stage and how much to leave to the execution stage to achieve the desired level of enterprise flexibility. Through this, we introduce the ability

for enterprises to evolve processes in the face of changes, which is crucial in highly volatile domains.

A process element can move from an executing process stage to a planning process stage (and vice versa) based on the goal-driven analysis of their contribution to relevant non-functional goals. Such movements create variations in the plan-execute behaviour and allow for either increased pre-planning (by moving a process element to the planning stage) or shifting more responsibility to the execution process stage (by moving a process element to the execution stage). Moving a process element to the execution stage results in decreasing plan completeness. This brings about flexibility and allows for handling of change at the time of execution, yet this also burdens the execute process stage with monitoring for change and processing additional data to best complete the partial plan that is provided to it. Thus, there has been a careful analysis about the degree of plan completeness.

A plan produced by a process stage either fully specifies the execution in advance or partially constrains the behaviour of the subsequent stages. We show how complete and partial plans are attained by positioning a process element on the planning process stage or the executing process stage in Fig. 6-16. In Fig. 6-16(A), we show the **Plan for Test** planning process stage and the **Test Software** executing process stage, with the plan process stage producing a complete plan, **Software Test Plan**. A *complete* plan consists of primitives that can be executed by the execute process stage without needing to make any decisions or figure anything out. Such a plan would be inflexible and restrictive, as it fully describes the execution of the downstream process stage. This is contrary to the concept of partial plans, shown in Fig. 6-16(B), where a *partial* plan is one containing unbound decisions and/or abstractions that are intentionally left open in the planning process stage. This is attained by moving the **Define Execution Sequence** process element to the executing process stage.

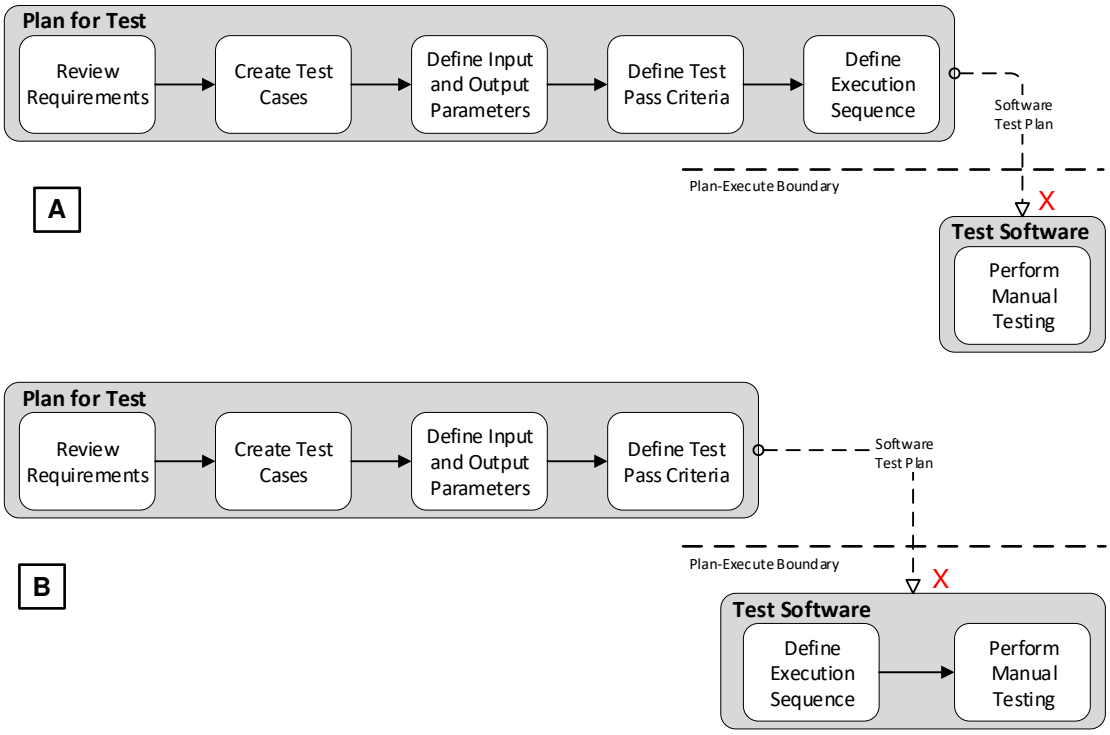

Fig. 6-16: (A) A plan-execute relationship between two process stages, (B) The same process stages with a process element moved from the plan process stage to the execute process stage

Partial plans are not executable until all decisions are made. They are required in situations where partial plan reuse is needed or having complete plans is not possible because of dependency on up-to-date contextual data. With partial plans, we can separate the confirmed activities from those that are uncertain or undetermined when the plan is being devised. As an example, consider the possibility where (depending on where an enterprise is in their software product release cycle), they want to have control over the test cases that are executed, and the order in which they are to be executed. Having a partial plan for software testing that leaves out the determination of the test case execution sequence for later is useful to have. Thus, the planning process stage generates a partial plan which permits the execute process stage a certain degree of freedom in its execution.

Complete or partial plans are considered based on the needs of downstream process stages. E.g. complete plans may be produced on a per-instance basis, fully customized for the needs of a particular process instance and therefore to be executed just once. Alternatively, partial plans may be general enough that downstream execution stages may have the flexibility of adjusting these

partial plans at execution time through binding different execution process stage decisions. Decreasing plan completeness increases flexibility and ability to handle change when executing the plan. It allows separating stable and volatile portions of specifications. Conversely, this puts pressure on the execution stage to monitor for change (which might incur data collection and processing costs) and to complete the partial specification provided to it by the planning stage based on the current context.

6.9 Reconfiguring Sense-and-Control Relationships

Enterprises need to develop business processes that are nimbler and more responsive to change. Traditionally, sense-and-respond portions of the enterprise may be situated at different locations in a hiBPM model. The sensing portion may be focused on operational process execution while the responding area may be more strategic. From a hiBPM modeling perspective, we considering differentiating between these two enterprise areas by situating them in different process levels. The structural elements in these process levels would have different types of relationships spanning process boundaries, including recurrence relationships, design-use relationships and plan-execute relationships. Under certain conditions, it may be desirable to have both sense and control activities at the same level. Therefore, there has to be a means to depict such feedback paths in the enterprise and determine what the optimum configuration should be with regards to the placement of process stages that are responsible for sense and control activities. Through feedback path reconfigurability, we can show and analyze the different ways of configuring the sense-and-control relationships in a hiBPM model. Some of these possible ways of reconfiguring sense-and-control paths are given in Table 9.

Table 9: The need and effect of reconfiguring sense-and-control relationships

Reconfiguration	Need for Reconfiguration	Example Effect of Reconfiguration on Softgoals
Adding process stages to the adaptive loop	A process stage can be added in either the sensing path or the controling path. Adding process stages to the adaptive path can result in better analysis of the sensed data or improved determination of controlling actions.	Increase: Methodicalness, Cost Decrease: Responsiveness
Removing process stages to the adaptive loop	Process stages can be removed from the sensing path or the controlling path. Removing process stages from the adaptive path results in faster enteprise response with the removed process stages becoming planning or designing process stages.	Increase: Configurability Decrease: Simplicity
Moving process stages to higher recurrence	Having the adapative path entirely contained within a higher recurrence results in autonomous sensing and responding to context where human intervention is not needed.	Increase: Perceptability Decrease: Scriptability
Moving process stages to lower recurrence	Having the adaptive path cross a a recurrence boundary makes possible for strategic planning and human intervention. The sensing part can be done at a higher recurrence with the analyzing part at a lower recurrence.	Increase: Automation Decrease: Customizability

Feedback paths reconfigurability can take one of two forms. In the first, only structural elements are relational elements are involved, whereas in the second, the boundary between the sensing and control is also a factor.

6.9.1 Modifying hiBPM Structural and Relational Elements

Let's consider a hiBPM model that contains a sense-and-control adaptive path, as shown in Fig. 6-17. As evident in Fig. 6-17(A), there are multiple process stages involving both the sensing path as well as the controlling path of the hiBPM model. The sense flow is between the **Monitor Environment** and the **Review Environment Metrics**, where the metrics captured from the production environment are analyzed and studied. The output of the **Review Environment Metrics** is a controlling flow that provides input to the subsequent design and development iteration of the software.

This hiBPM model can be modified to better handle the feedback cycle by adding process stages to or removing them from the adaptive path, as shown in Fig. 6-17(B). Here, the control flow is

directly connecting to the **Setup Software** process stage, thus by-passing the **Design, Develop and Deploy Software**. This is to have faster control input to modify the production environment based on analyzed production metrics.

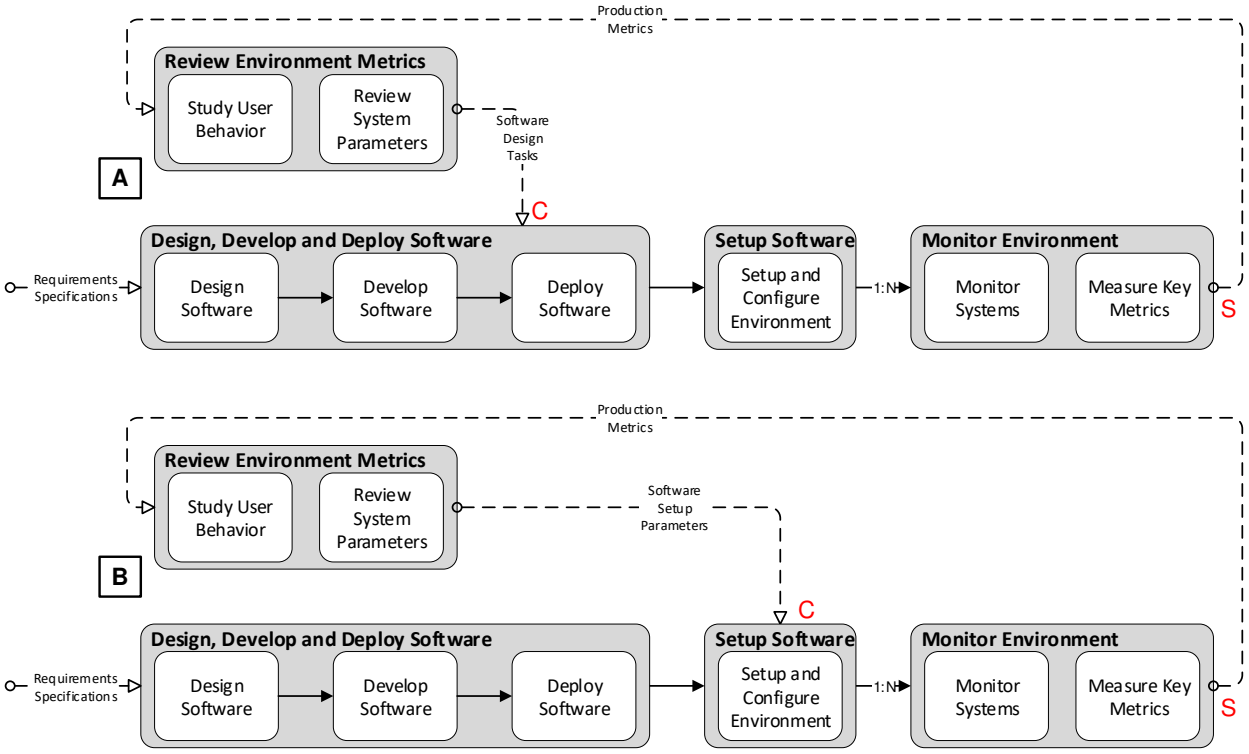

Fig. 6-17: (A) Process stages in a sense-and-control adaptative loop, (B) Reduce the process stages in the sense-and-control adaptive loop

The need to add or remove process stages to the sense and control path arises to simplify particular parts of the adaptation loop or to add processes for additional planning and designing. In order to appropriately configure the hiBPM adaptation loop, the softgoals are first determined. In some cases, the primary softgoal may be for faster responding to sensed data; in others, it may be a more accurate determination of what needs to be done,

- For the former, improving execution of the process stage(s) could be done by selectively reviewing and removing process elements from within the loop execution path. These process elements could be eliminated from the hiBPM model or moved elsewhere to other process stages, and whose pre-processed output would be used by process stages within the loop.

- For the latter, more accurate determination could be done by adding additional structural elements that provide additional validation to the output of the previous process stages. This can be done by executing test plans that are prepared outside of the loop.

6.9.2 Moving Structural Elements across Process Boundaries

In some cases, the sense-and-control paths can cross process boundaries. The sensing process stage **Monitor Environment** could be automated and executing rapidly, based on the capability of underlying software systems to detect and sense change. This sensed data would be sent to the **Review Production Metrics** process stage (after spanning the recurrence process boundary) that executes at a lower recurrence, possibly due to humans requiring to review the data and come up with a response for a future course of action. We show this in Fig. 6-18(A) where both the sense flow and the control flow cross a process boundary. We use the generic process boundary here as the arguments apply equally to recurrence boundary, design-use boundary or plan-execute boundary.

If additional capabilities are built in the hiBPM model, so that the responding side can process the incoming sensing data effectively, and suitably produce a course of action, then the adaptation can occur within the same process boundary, without going across process boundaries. This, for example, can be done through automation through which the **Review Production Metrics** process stage executes at the same recurrence level, shown in Fig. 6-18(B). This enables a dramatic increase in efficiency and effectiveness in both sensing and interpreting the environment and allows for real-time adjustments to the production environment.

In hiBPM models, such adaptive loops are used for revealing unique adaptation relationships between two "orders" of processes. A higher-order process stage is considered to be a design or plan process stage to its lower-order use or execution process stages, respectively. It senses how well its lower-level process works and may change the way it operates. The change is either through a "control" flow constraining the possible options for the target process at runtime or through an "execute" flow that changes the space of options for its target process by creating new capabilities. Hence, there is a hierarchy among processes that reflects their relative control order. The execution frequencies of both these levels typically differ as well.

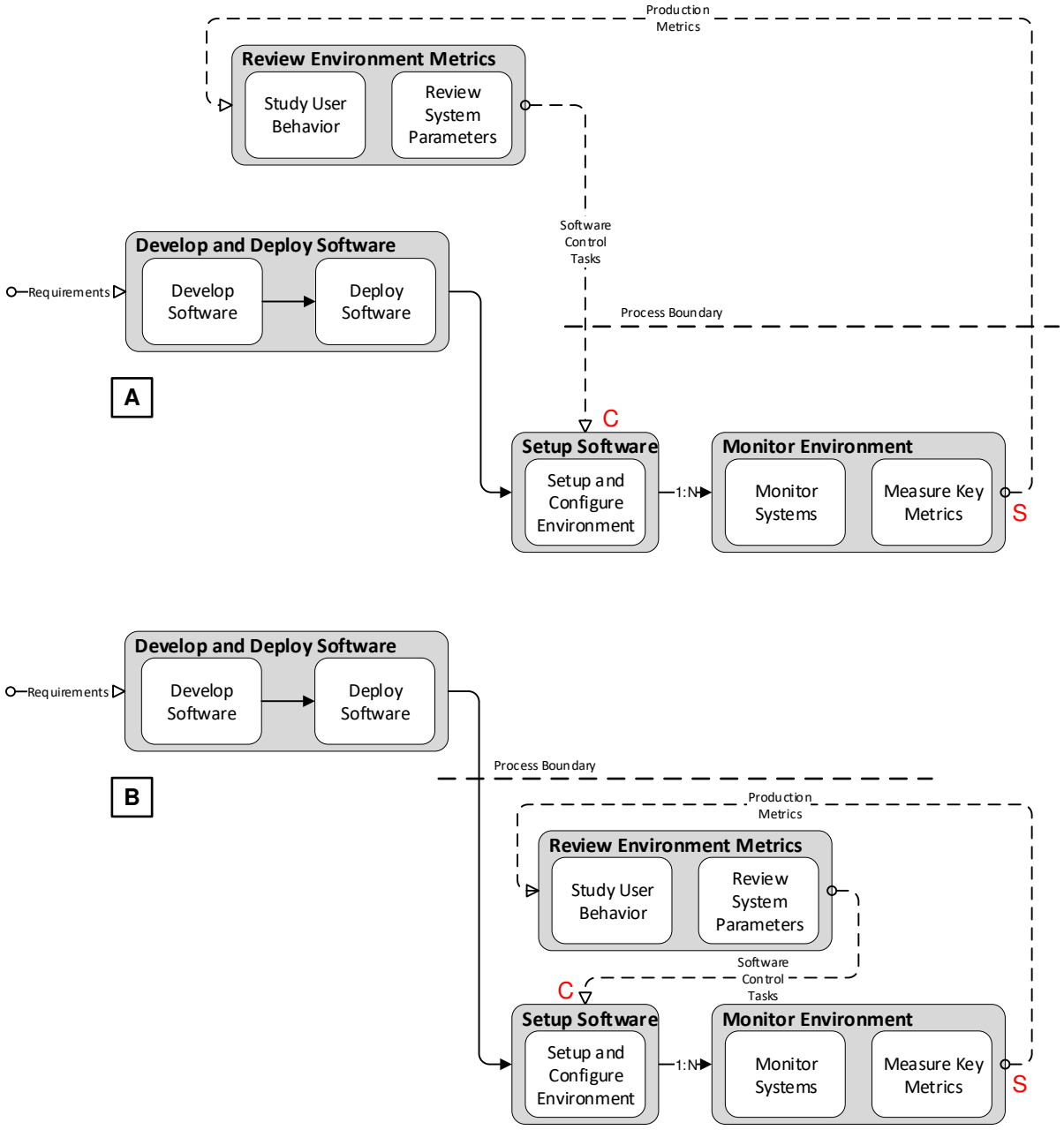

Fig. 6-18: (A) Sense-and-control adaptive loop spanning a process boundary, (B) Sense-and-control adaptive loop within a process boundary

6.10 Conclusion

Enterprises tend to optimize their processes to ensure maximum efficiency and speed of execution. This is possible if the enterprise is operating in a static environment with unchanging circumstances or evolving requirements. Here the processes can be designed once and contain little to no design variation. However, most enterprises in dynamic and changing environments, particularly those that are software-enabled, need to rapidly adapt to, and innovate in response to, evolving requirements. Thus, the hiBPM model needs to be modifiable to support the enterprise. In order to do this effectively, flexibility is introduced at appropriate places so that the enterprise can add necessary design reconfigurations in a low-cost manner.

For managing enterprise transformation, we need to understand the possible space of design options that are necessary and permissible, which allows for contemplation of possible alternative hiBPM model options. Navigating the space of such possible hiBPM model configurations while considering goals and softgoals is difficult and may result in trial-and-error practices being employed without convergence to an acceptable solution. Enterprise architects need to be prepared to explore this design space of hiBPM configurations by employing techniques from their repertoire to point out the design decisions and possible alternative configurations, while considering aspects such as trust and confidence in the systems' decision-making ability, effort required to help with decision making, compliance with industry regulations and company rules, the cost of deployment, etc.

In this chapter, we discussed how a hiBPM model can reflect the domain characteristics while considering the dynamics and volatility that is present in an enterprise. It should help organizations determine the appropriate level of flexibility and rigidity necessary to meet the various functional and non-functional goals. We presented several possible hiBPM process architectural configurations that exist in the design space. However, the configurations presented are not meant to be exhaustive and there could be additional ways in which the hiBPM model could be configured.

7 Case Study – Enterprise Process Innovation

Acknowledgement: This chapter is partially based on the following paper;

- *Babar, Z., Yu, E.: Integration of Software Applications into Evolving Enterprise Business Processes. In Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS), Vol 2, pp. 778-786 (2020)*

7.1 Background and Context

The first case study was conducted at a large Canadian food retailer and general merchandise provider. The company has hundreds of retail stores located in several provinces across Canada. In addition to these stores, the retailer has several distribution centers and a trucking fleet that is part of its supply chain service. The retailer is in the midst of an overall industry transformation where companies in this space are increasingly relying on digital technologies to transform key business processes to improve operational efficiency and customer-centricity. To ensure the anonymity of the retail organization, this case study abstracts away from specific details and the organization context is described in a more generalized manner.

The company was continuously looking to improve the efficiency of the entire demand and supply chain to better serve its customers. An essential operation in the supply chain is that of planning and forecasting the sales volume of various products at individual stores across the country. Such a planning exercise requires several crucial business, technology and software processes to come together to be able to calculate predicted sales volumes for several future periods at different levels of granularity (daily, weekly, and monthly).

As part of the case study research activities, we studied multiple business, IT and software processes that were part of the sales forecasting and promotion management area, and the usage of data analytics applications within them. Through these business processes, knowledge workers were able to forecast future product sales across multiple stores for geographical regions. Through software processes, the organization developed and delivered in-house and vendor-provided retail applications that are used during the execution of the business processes. Operations and maintenance of these retail applications, data sourcing and ingestion, and ad hoc data analysis were part of the information technology (IT) processes. Being a large retail organization, there were

several enterprise applications that were part of our study domain. These included software systems in the data ingestion pipeline that sourced software from retail stores and stored a transformed form into the enterprise data warehouse, and software systems through which data evaluation and analysis could be performed.

The enterprise required a software solution that would not only provide more accurate sales forecasting numbers (using a data analytics application) but also have an approach that would allow for evaluating about how to best integrate the analytics solution across several enterprise businesses, software and information technology processes while comparing and analyzing between multiple alternative configurations of these processes. Such an approach would help the enterprise decide on the optimum business process design for that data analytics solution.

7.2 Case Study Investigation Parts

There were three parts to the case study,

Enterprise data analytics: The organization wanted to use a data analytics application to help with sales forecasting and promotion planning while considering the multiple types of promotions could exist, and their effect on sales uplift. In the first part for the case study, the aim was to deliberate on the design of a prototype data analytics solution that would provide improved accuracy for the forecasted sales orders. The solution would also facilitate coordination among the various stakeholders that are part of the existing sales forecasting and promotion planning operations, while enabling faster solution development and shortening cycle times for simulations performed for promotion planning and sales ordering.

Enterprise process innovation: The data analytics application, while providing the necessary benefits in forecasting accuracy, would need to be well integrated with organizational business, technology, and software processes. The existing constraints in the process design could also influence the design of the data analytics solution. In the second part for the case study, the aim was to provide guidance into how such a data analytics application could integrate into a complex enterprise process environment while ensuring that processes are configurable and amenable to

change without significant redesigns, particularly as the data analytical solution could evolve, or new data analytics applications could be adopted.

Building up enterprise capabilities: All enterprises have inherent capabilities that allow them to develop and produce goods and services. For the third part, the aim was to analyze the dependencies between business capabilities, other organizational capabilities, and advanced technology capabilities. The intention was to determine presently enabled enterprise capabilities, including capabilities that constraint technology and process innovation. Through capability modeling, we could determine the required capabilities over both short-term and long-term timescales, including those needed to support ongoing attainment of the enterprise objectives for this data analytics application, and determine effective designs for introducing agility in critical areas of the enterprise.

7.3 Objective

In the case of this thesis, the second investigation part was more relevant and this researcher approached the case study from the perspective of modeling and reconfiguring the process architecture to support the needed enterprise innovation using the hiBPM framework. Thereby, the research objective was to evaluate the effectiveness of the hiBPM framework by demonstrating its use in the context of this case study. This objective was carried out through the following,

Modeling and analyzing a set of interrelated processes: There were changing requirements to support a data analytics solution, in the form of the changes to the software processes that develop the solution, the technology processes that provide the necessary data that help with training the solution, and finally the business process where the solution is finally used. The interrelated processes would need to be designed around variation points to accommodate the final solution. We needed to validate the hiBPM framework's ability to visually model and analyze a multitude of business, technology and software processes while accommodating design uncertainty.

Analyzing integration of data analytics application in existing processes: There were bidirectional dependencies between the design of the data analytics solution and the business processes. Thus, the data analytics application that integrates into the business process also needed

to be designed to accommodate existing process design and the expected usage of the software. We needed to validate the effectiveness of the hiBPM framework in influencing the software design while considering the interplays of software and process redesign.

7.4 Activities

In Chapter 1, we presented a method for attaining research structure and rigour in our case studies with three distinct areas. We detail the specific research activities performed for each of those areas below.

7.4.1 Area 1 – Research Design

A well-defined business problem was presented to the research team by a senior member of the company at the initiation of the case study. This was in the form of a one-page document that defined the scope of the study and a desired outcome. The business problem also defined the primary business processes that needed to be studied, thus limiting and providing a context for the case study. The problem statement was then decomposed into the three distinct parts, as described in the previous section.

The research team consisted of members from both the company and the university. From the organization, the team members included individuals from the IT department and business department with senior management staff periodically reviewing the progress of the project. There was one main contact person from the company side, a manager in the IT department, who was responsible to provide the necessary data and to review the produced hiBPM models and contributing to the direction of future investigation.

From the university's side, the team consisted of three PhD students and their supervisor. Each of the above investigation parts was assigned to an individual based on their research interests and applicable background. As mentioned in Section 7.2, the second part (pertaining to enterprise process innovation) was more relevant to this thesis and was this researcher's primary focus and the research objective was defined accordingly. However, to ensure that the case study findings across the different areas supported each other, there were collaborative discussions and iterative refinements in both the data collection and data analysis research activities.

7.4.2 Area 2 – Data Collection

Research activities were defined for data collection early in the case study. On establishing a frame of reference and research guidelines, various individuals across the organization were identified who could help with the gathering of data and review the outcome of the data analysis. These individuals were selected from both the technology and business side of the organization. The project duration was 12 months. At the beginning of the project, this researcher spent several hours a day, and multiple times a week, on-site at the organization's corporate headquarters to understand the business domain, including the technology and business processes in the defined research context, and to study documentation that was only available for access through the corporate network.

The data collection area involved reviewing corporate documents, understanding the company's use of different software artifacts and tools, and existing conceptual diagrams of business processes and enterprise architecture. Staff members from the organization were available to both explain and supplement the information provided in the documents and conceptual models. A case study database was maintained where a repository of the collected data was stored for later review and analysis. The data collection was iterative, with the outcome of a previous data analysis step determining the next set of data to be sourced for analysis.

7.4.3 Area 3 – Data Analysis

The provided documents were supplemented with field notes that captured the verbal discussion for later analysis. The data collected from these multiple sources were reconciled with the actual process of data analysis following either a logical chain of evidence or explanation building. In logical chain of evidence, we established a connection between the problem presented earlier in this chapter to subsequent evidence trails that were apparent in the data collection. This led us to hiBPM modeling constructs that aided in analyzing the situation under study. By reviewing our notes, we could associate which hiBPM construct could be used to model an identified item from the notes. In explanation building, we looked at the collected evidence (as explained in the data collection area) and attempted demonstrate the capability of hiBPM modeling constructs to help in both visualizing and analyzing alternative configurations for the situation.

As the findings from data analysis activities were verified, they were shared with the broader research group, usually once a month. These team meetings included members from both the organization and the university, including the project facilitators on the organization's side. The research findings and feedback received during these meetings were consolidated in a final report. This shared at the conclusion of the case study with certain members on the organization's side for confirmation of its accuracy.

7.5 Modeling the Domain

To solve the presented problem, we focused on how hiBPM constructs can be used to model and analyze the software artifacts and their integration into the overall process architecture. For this, we take individual examples and explain them further in this section. As part of this case study research, there was not a significant emphasis from the participating organization on comparing and contrasting between alternative configurations of the hiBPM model on the basis of non-functional requirements. Rather, these reconfigurations were considered more from the standpoint of the possible ways to configure the existing processes, with a cursory understanding of the benefits (and drawbacks) offered by each variant. Thus, the hiBPM models below do not have detailed accompanying goal models, as these were not developed significantly for this case study.

7.5.1 Evolving Design Capabilities

In the case of software applications, components are designs built by different software processes which are then used during the execution of the business (or another software) process. The hiBPM model was able to capture the relationship between both sides, i.e., where the software component is produced and where it is used. Through design-use relationships, we can show the process stages that result in the building of a design (i.e. software artifact) and using of that design.

We show a simplified model with two process stages in Fig. 7-1, each of which has multiple process elements. The process stage **Develop Analytical Model** is responsible for creating the designed artifact (i.e., an **Analytical Model**) and **Perform Weekly Analysis** is responsible for (repeatedly) using the designed artifact. Here, the **Analytical Model** is a design that is used by another downstream stage. Thus, a design-use relationship exists between these stages.

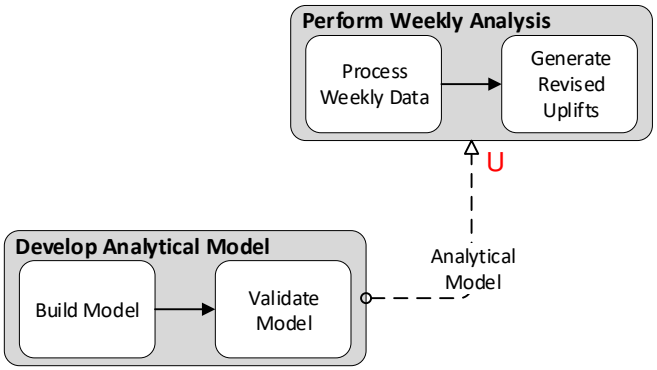

Fig. 7-1: Design-use relationship between the process stages that are part of the weekly sales revision function

In Fig. 7-2, we show how variation in design-use behaviour is attained by positioning a process element on the design side or the use side of a process boundary, i.e., whether the process element is invoked as part of a design process stage, or is invoked during the use of that artifact, tool, or capability that is the outcome of the design. In Fig. 7-2(A), the use of the **Analytical Model** is shown to be fully automated to simplify the process of creating new analytical models with minimum effort or deliberation; here, the design plus build process stage takes on more functionality, thus increasing the level of automation in the use process stage. By automation we mean that the **Perform Weekly Analysis** process stage can be entirely scripted for execution without needing any human intervention.

Conversely, having a partially designed **Analytical Model** allows for customizing the use of the **Analytical Model** to fit specific needs, in this case being able to change various model attributes to ensure that the sample analytical model can be used repeatedly. For this, Fig. 7-2(B) shows a partial design that is then used during the **Perform Weekly Analysis** process stage. Recall in Chapter 6 that reducing process elements in, or moving process elements across, the design process stage to the use process stage reduces the level of automation available and increases the customizability of the tool. At use time, through human intervention, the **Analytical Model** behavior can be modified through assigning different model attributes. Such customizability may be necessary based on changing enterprise objectives.

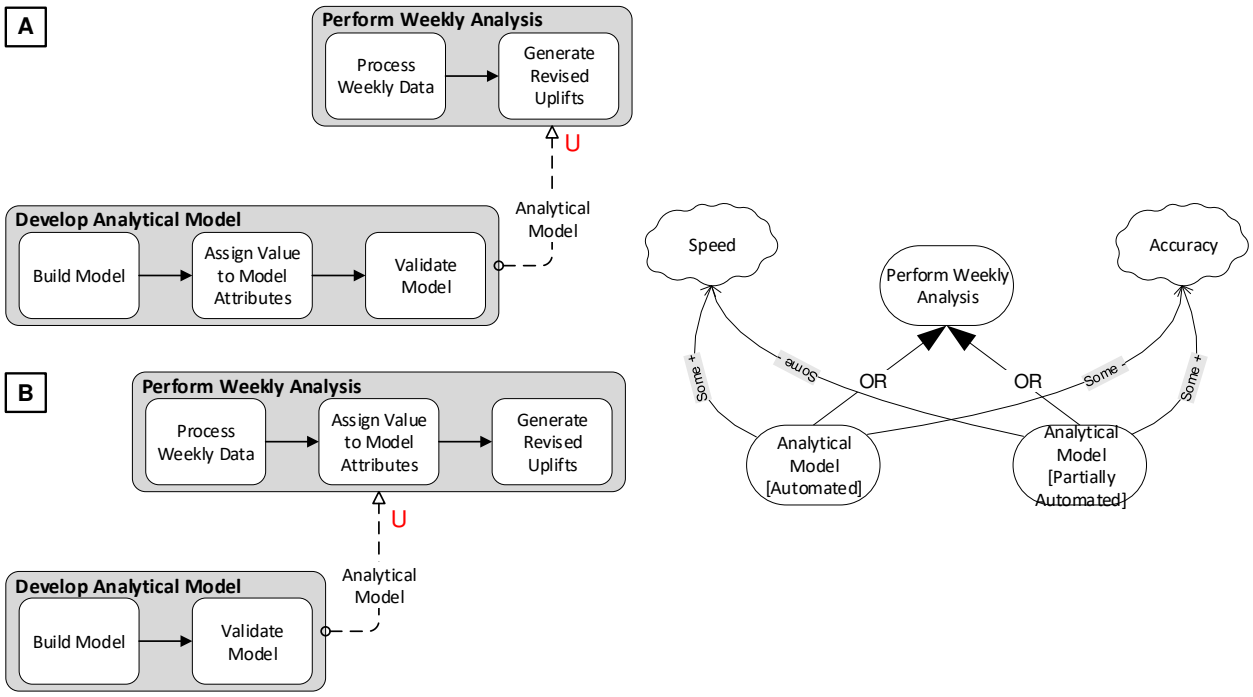

Fig. 7-2: Evolving capabilities through design-use relationships for the weekly sales revision function

7.5.2 Flexibility of Process Execution

Fig. 7-3 shows a plan-execute relationship between two process stages, **Plan for Analytical Model** that provides instructions on how to go about with the building of an **Analytical Model**, and **Develop Analytical Model** where the provided plan is executed to design and build the actual model. The **Plan for Analytical Model** process stage determines the goal and purpose of this model, including the attributes that are to be contained within this template. These are codified as a plan that is then executed by the downstream process stage during its execution.

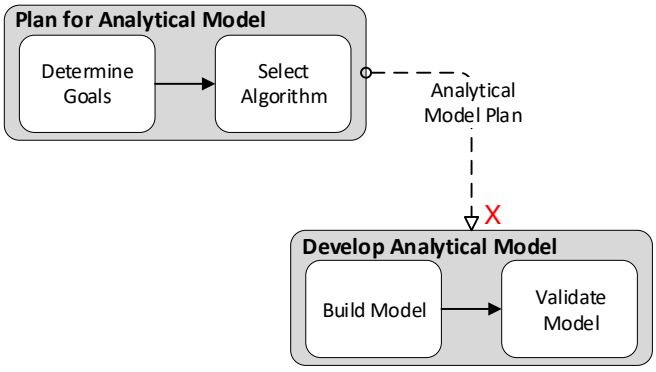

Fig. 7-3: Plan-execute relationships between the process stages that are part of the weekly sales revision function

In **Plan for Analytical Model**, the plan on how to build the **Analytical Model** can be fully elaborated or certain design decisions (such as the attributes to use when building the instance) left for later determining. Locking the selection of the attributes is helpful when there is no uncertainties when building the **Analytical Model** and the same instantiation would be repeatedly required. Alternatively, leaving these parameters open is beneficial when a model may be used across different settings and custom values provided during the building process. Both of these alternatives are shown in Fig. 7-4(A) and Fig. 7-4(B).

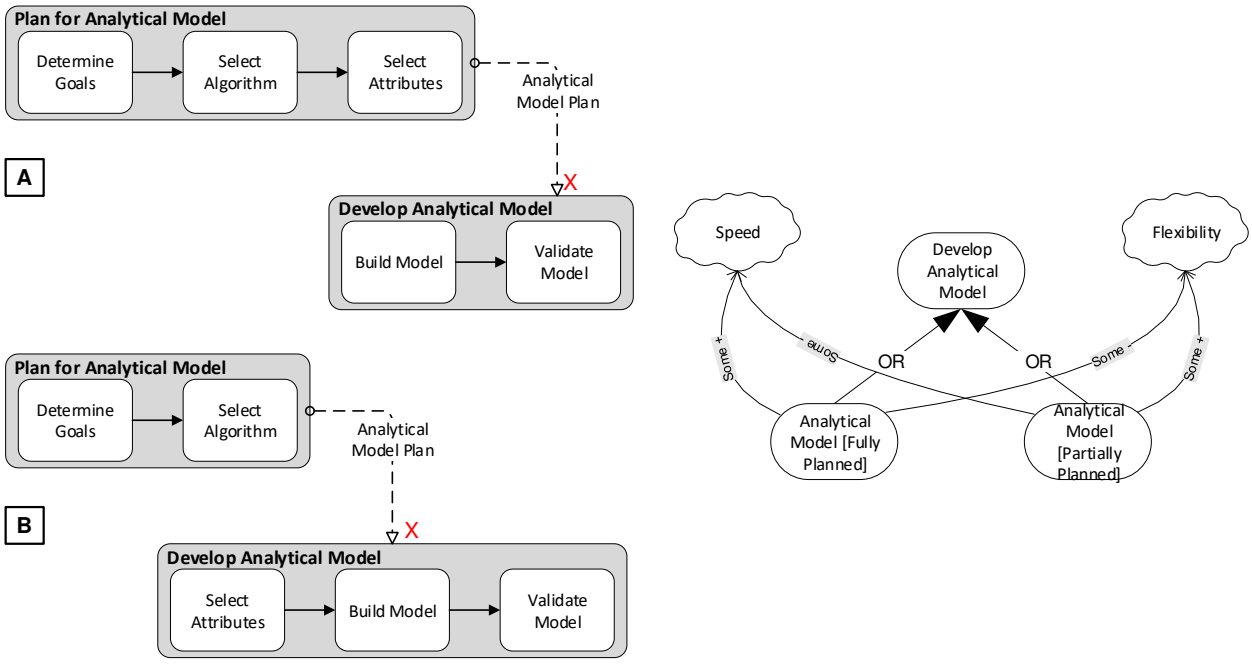

Fig. 7-4: Execution flexibility through plan-execute relationships for the weekly sales revision function

7.6 Enterprise Agility through Flexible Planning and Evolving Designs

Generally, software systems, artifacts or tools need to be designed for automation or re-usability. In practical terms, such software architecture designs are produced through well-defined interfaces, class structures and hierarchies, and code components, frameworks and libraries. While such software can be designed in a manner that allows for reusability by other software systems or users, crucial design decisions need to be made regarding their deployability and usage over a range of conditions and settings. On the other hand, plans can be used to inform and guide the execution of processes that use software systems and tools. Through the collective use of design-

use and plan-execute relationships, we can provide the ability to express and analyze these situations.

The design-use relationship allows for handling various kinds of situations; however, it does not indicate how to have flexible execution at runtime. This needs to be shown separately from the design-use relationship in the form of plan-execute relationships. Thus, using both plan-execute and design-use relationships can allow for introducing flexible design capabilities, simultaneously helping to understand when and how to introduce change in execution behaviour. Here, a plan can influence how a design is used by providing varying instructional input to the process stage that is using the design. We show how both relationships come together to bring about both flexible planning and evolving design capabilities in the enterprise in Fig. 7-5.

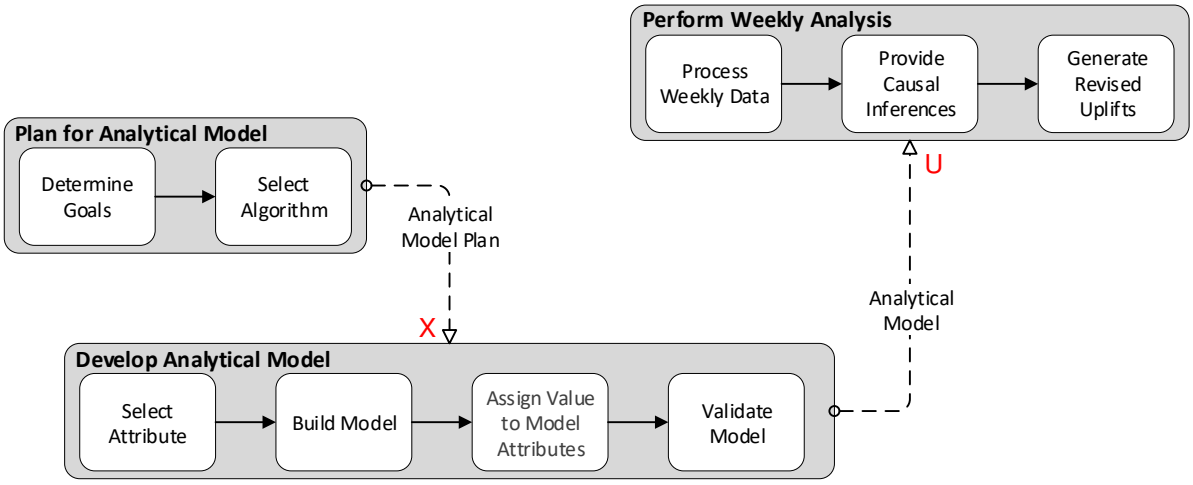

Fig. 7-5: Combining design-use and plan-execute relationships for flexible plans and evolving designs

Introducing design-use and plan-execute relationships to the process architecture may require additional changes as well, particularly in the case of supporting evolving design capabilities. Representations of design, development or other tool acquisition processes need to be integrated into a process architecture to allow modeling of evolving capabilities available to the enterprise while supporting continuous design. For instance, being able to evaluate tool redesign cycles relative to other changes in the enterprise allows the identification of rigidities in organizations and the evaluation of cost-effective ways to remove them. Often, these necessitate the addition of supporting process stages that surround the locations at which the design-use and plan-execute relationship are introduced. These are done to ensure that the use process stage or the execute

process stage has all the requirements available to process design or plans. For example, in Fig. 7-5, the **Plan for Analytical Model** process stage provides the necessary information for the execution of the **Develop Analytical Model** process stage, specifically the need for having an **Analytical Model Plan** that would be used during the execution of the **Develop Analytical Model** process stage.

There is an essential distinction in how designs and plans are conceptualized in hiBPM. As was discussed in Chapter 5, designs are considered to be pre-built black box artifacts that can adopt different forms; they may be physical objects, a digitized entity or even be informational. The designs are black-boxes as a user of the design should not care (neither is informed) about the (internal) structure of a design artifact or how it is built, and is only concerned about whether the functional and non-function objectives are achieved when invoked during using of the design. Contrary to this, a planning process stage devises the plan irrespective of how the execution process stage will execute it. The execution process stage is aware of the plan internals to interpret and best execute the plan based on requirements and trade-off analysis that can be done as part of its execution.

This is important to understand in the context of this case study. As part of the university's engagement, a prototype **Analytical Model** was to be developed. Based on the merits of this exercise, the organization could decide to further develop the **Analytical Model** in-house or source it from a vendor. The aim was that despite the uncertainty in the form of the final data analytics application, the organization's processes should not have to substantially change from what was designed and determined in the case study. Through the design-use and plan-execute relationships, we were able to show what designs are needed, and what plans are to be devised, so as to support different possible forms of data analytics applications.

Driven by changing business needs and external environments, as well as based on the feedback from the use of the current version of the tool, the design process stage can be re-executed when appropriate. This will produce or acquire new versions of the capability, thus evolving the enterprise and its systems. Therefore, product innovation and process innovation both can be considered simultaneously, including how they are socially integrated into the enterprise.

7.6.1 Fully Automated Forecast Adjustment

Additional possible solution configurations were identified by the research team during the deliberation process. Analyzing and adjusting the product sales number every week is less than optimum as it does not factor in the daily fluctuations. However, that is a necessary compromise as doing the same operation on a daily basis against daily sales data would be additionally time-consuming and computationally intensive.

In Fig. 7-5, we presented how the sales forecast was either positively or negatively adjusted in the **Perform Weekly Analysis** by forecast planners in the business department after a round of sales review and using their collective experience. This adjustment was an automated activity performed by a data analytics system that automatically generated revised uplifts. However, this is a simplistic solution and merely mimics the current process behaviour present at the enterprise (albeit automating critical aspects of it). The **Analytical Model** designed adjusts uplifts on a weekly level without considering the daily variations in forecasted sales and actual sales. A more sophisticated solution would have both daily and weekly forecasted sales adjustments, with the process architect adjusting the workflow as needed.

Such a solution would require a redesign of both the analytical model *and* the processes where the analytical model is used. The redesign trigger is the **Monitor Context** process stage, which actively determines on a daily basis if the forecasted sales and actual sales numbers are sufficiently different. Once it detects that the deviation is statistically significant, it would initiate a redesign by calling the **Plan for Analytical Model** and **Plan for Process Config** process stages with the appropriate data. Note, this reconfiguration of both the **Analytical Model** and the hiBPM model is not done on a per-instance level. Rather it is meant to be an infrequent reconfiguration when there are sufficient changes in context to warrant such an expensive operation. The hiBPM model snippet showing the new process architecture configuration is shown in Fig. 7-6.

Both these processes accept the **Filtered Context** and initiate replanning activities that result in a redesign plan - **Analytical Model Plan** in the case of **Plan for Analytical Model**, and **Process Reconfig Plan** in the case of **Plan for Process Config**. As before, the **Develop Analytical Model** takes the **Analytical Model Plan**, along with accepting a **Model Catalogue** consisting of analytical model

design patterns, to produce an **Analytical Model**. This **Analytical Model** can generate uplifts either at daily or weekly periods. Similarly, the **Plan for Process Config** generates a **Process Reconfig Plan** that is then processed by the **Execute Process Config Plan** to reconfigure the internal process elements of the **Perform Sales Analysis** stage. Note, the process stage name is different as it is no longer just dedicated to weekly analysis but the sales analysis can be done on either a daily or weekly, depending on how it has been reconfigured for execution.

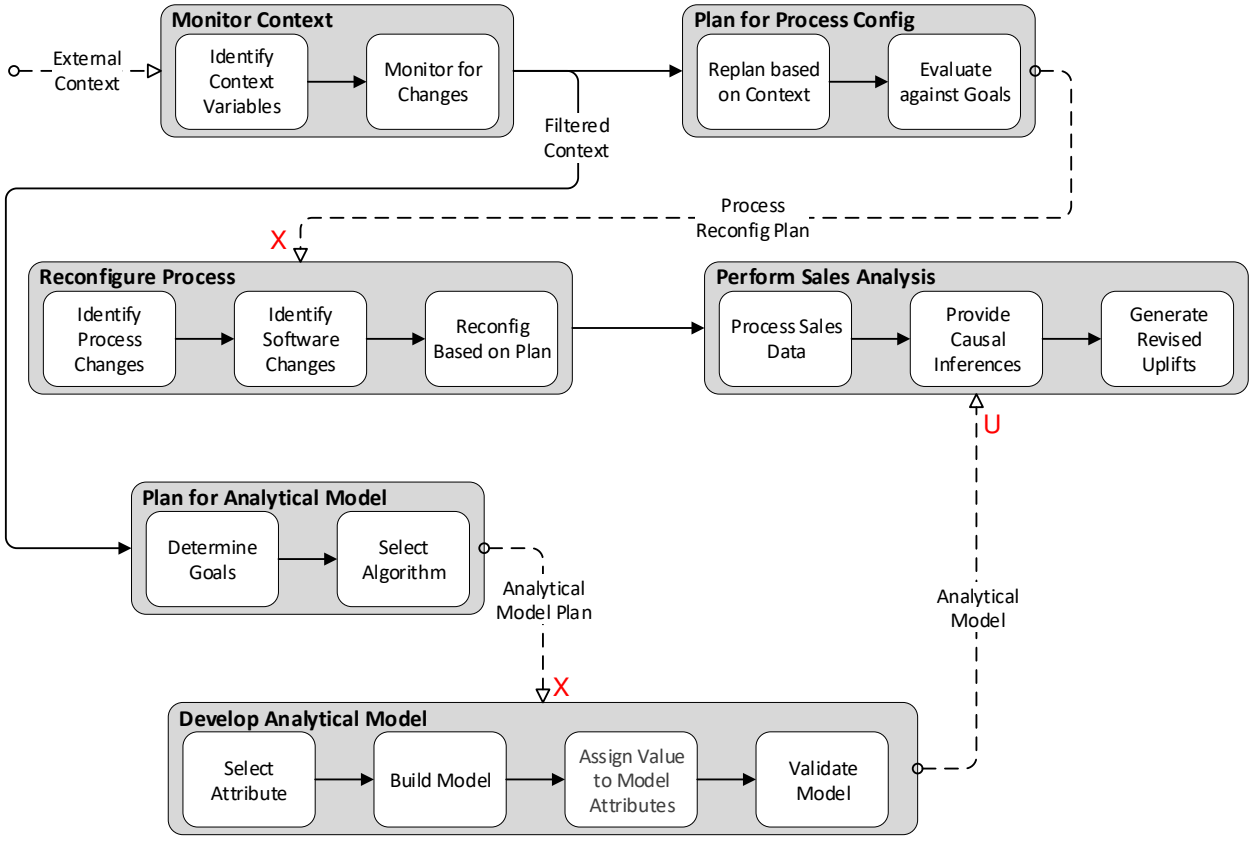

Fig. 7-6: Introducing surrounding process stages for full automated forecast adjustment function

7.6.2 Partially Automated Forecast Adjustment

The hiBPM model in Fig. 7-6 provided full autonomy of sales forecast adjustment at the cost of manual control. Another possible solution configuration was one in which the forecast planners in the business team would still have some manual control on the adjustment process and be able to use various "levers" to adjust the sales uplifts manually and evaluate the simulated forecasted numbers for the next several daily and weekly periods. This was important as the individuals from

the business team wanted to adjust the forecasted numbers based on their extensive experience and tacit knowledge that they brought from field operations. Some of the causal factors that affect the sales orders were not captured in the data warehouse, and thus the **Analytical Model** could not be trained against them. Having manual control to simulate and adjust the forecasted sales allowed for improved accuracy beyond what the **Analytical Model** could provide.

Fig. 7-7 shows the hiBPM model configuration for such a scenario. To simplify the scenario, here we consider weekly analysis and adjustments. The process element **Generate Revised Uplifts** now accepts a set of input parameters that influence the calculation of revised uplifts. The forecast planner may manually modify different variables that have causal relationships with sales activity for a particular product or location. Examples of such variables may be weather patterns, seasonal holidays, competitor activity, etc. **User Assess Weekly Data** is the process stage that reviews the weekly sales numbers and attempts to simulate new sales forecast by providing different values for the causal variables than would have been otherwise provided to the **Perform Sales Analysis** process stage from elsewhere in the system.

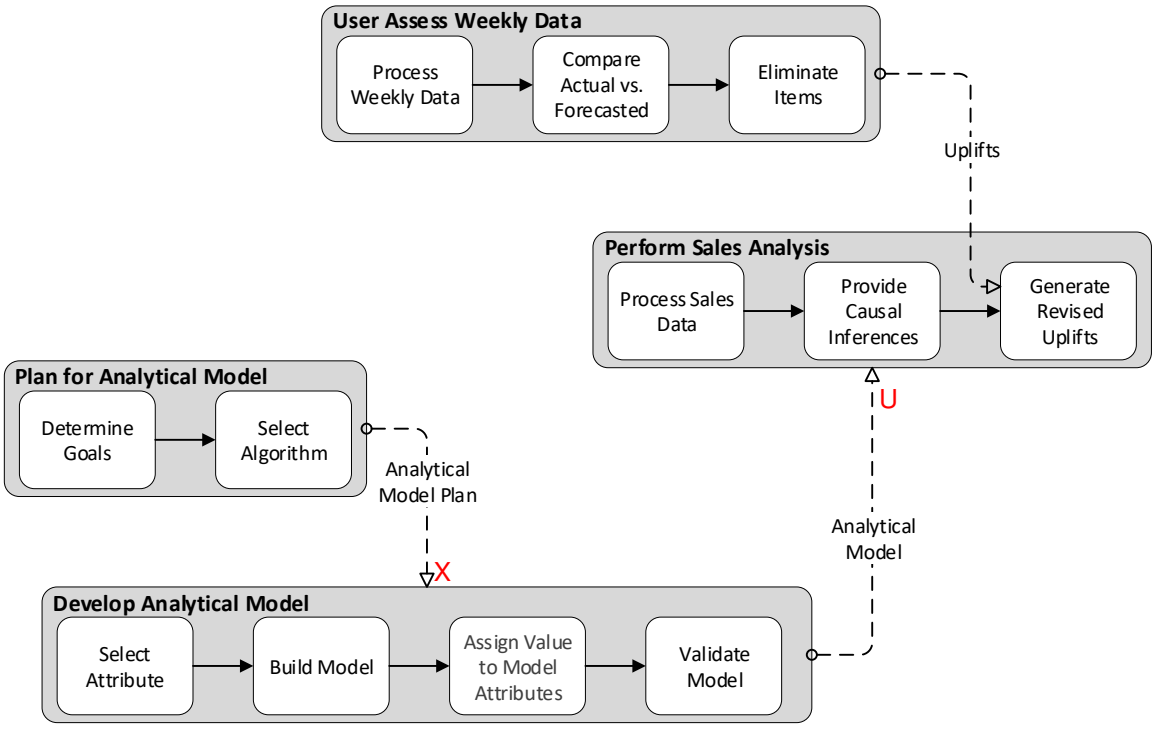

Fig. 7-7: Manual control for partial automation of the forecast adjustment function

7.7 The Complete hiBPM Model

We show the complete hiBPM model with several business, technology and software processes in Fig. 7-8 that collectively help the organization accurately predict product sales and calculate orders for its many stores while keeping in mind customer product purchase demands and any associated promotions that influence this demand.

The information technology process stages **Collect Sales Data**, **Preprocess Sales Data**, and **Aggregate and Load Data** collect, aggregate and transform data to be used for forecasting future sales for product items across various product categories that are to be sold in multiple retail locations. The **Forecast Future Items Sales** process stage predicts how many items are going to be sold by day-of-week for the next several weeks, across all retail stores. The promotion planning department devises promotions to nudge customers to purchase particular products more than they would otherwise; these promotions are planned and determined in **Prepare Merchant Promotion Plan** and an adjustment factor (called uplifts) that would indicate how much higher the sales would be because of the promotion, are computed in **Calculate Promotion Uplifts**. For more personalized recommendations, the loyalty department generates individualized product offers through loyalty campaigns in the **Prepare Loyalty Mass Promotion** process stage. The loyalty offers for each item are mapped to targeted loyalty plan members, and an uplift calculated for each unique customer-item mapping to **Generate Loyalty Targeted Offers**.

Calculate Product Order process stage calculates the product order (at a product-location level) for the next several weeks, which are sent to the suppliers. Despite all the effort put into forecasting future product sales, there is still some difference between the forecasted numbers and the actual sales that happened. Hence, in **Perform Weekly Analysis** process stage, a review of the previous week's actual vs. predicted performance is done, and if required, uplift adjustments are made to future week forecasted sales to align the forecasted sales to the actual sales expectations better. Uplift values greater than 1 indicate an upward sales adjustment, while values less than 1 indicate a downward sales adjustment.

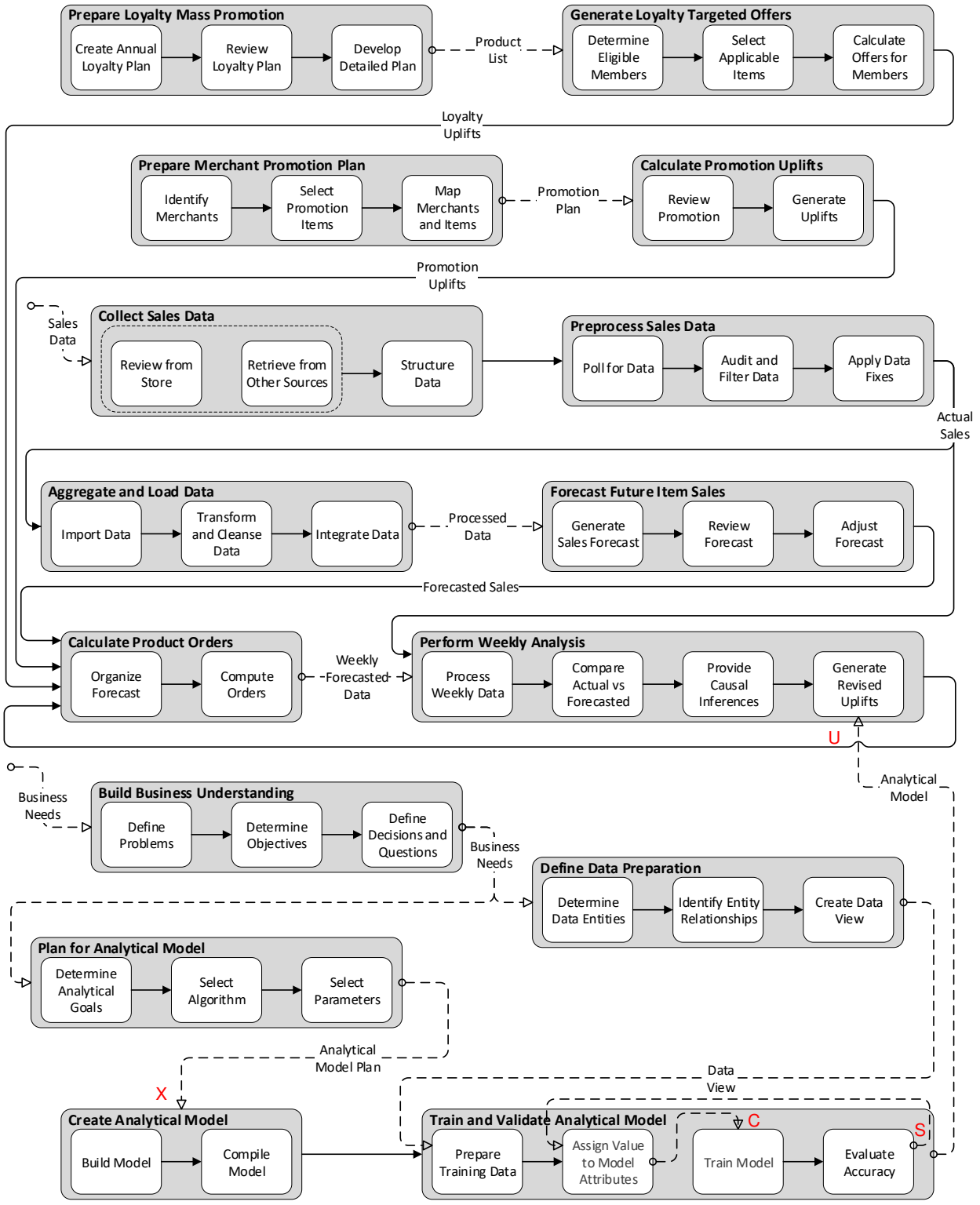

Fig. 7-8: Complete hiBPM diagram for the Enterprise Process Innovation case study

7.8 Data Analytics Solution

As mentioned earlier in this chapter, there were bidirectional influences between the design of the process architecture and the design of the software architecture. Any software architecture that was determined would need to integrate into the existing processes while adhering to design constraints placed by these processes, and still satisfy the functional and non-functional goals. Through componentization, the solution should also be adaptable to change.

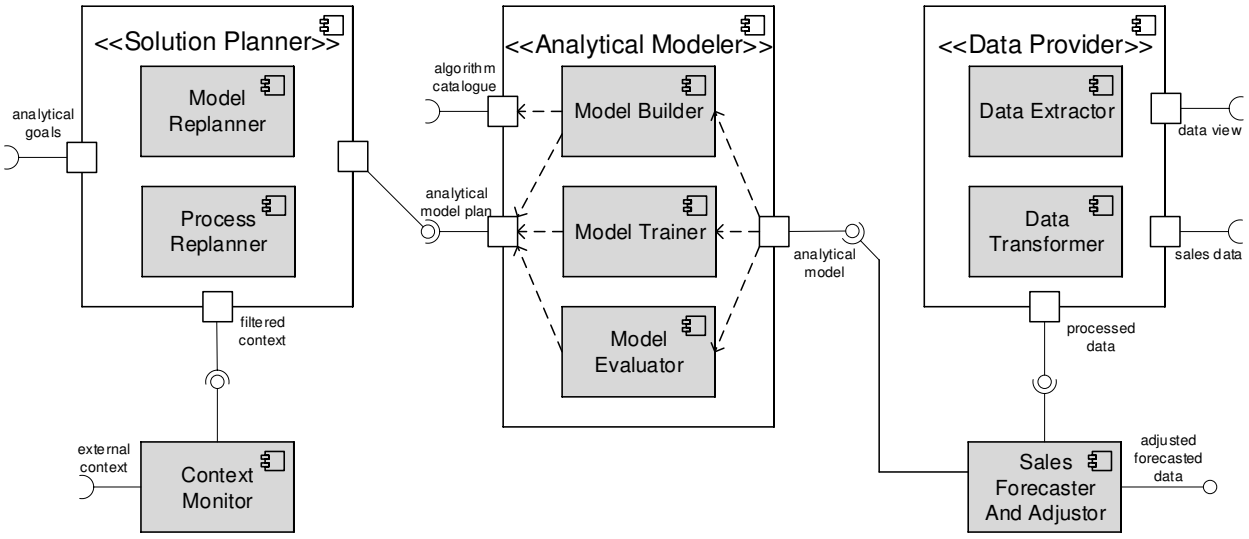

Fig. 7-9: UML component diagram for the prototype data analytics application

The prototype architecture of the data analytics solution is partially illustrated in Fig. 7-9 as a UML component diagram. This diagram shows the primary software components and the necessary data exchanges between them. Here we emphasize the primary logical UML deployment artifacts (enclosed with the double arrows <<>> in the diagram). These components were determined through an analysis of the hiBPM models where we reviewed the hiBPM models to determining designs, plans, and data flows that were present in those models. These were abstracted out as software components.

We describe each component, and how they were determined, in the bullet points below.

- **Context Monitor** monitors and evaluates for external context. Evolving enterprise requirements and environmental circumstances causes changes in external context. Not all context is useful

therefore this external context is first processed and a **Filtered Context** is then passed to the **Solution Planner** for triggering either process redesign, software redesign or a data preparation redesign. This component is determined based on the **Monitor Context** process stage shown in Fig. 7-6.

- **Solution Planner** component is triggered on context change and determines a suitable plan to modify the analytical model design or the process design. The **Solution Planner** evaluations the filtered context and produces an **Analytics Model Plan** that provides instructions on how to select appropriate machine learning algorithms to form a solution design. This is then used by the **Analytical Modeler** as an input. The **Solution Planner** relies on **Analytical Goals** to provide necessary details, such as the business goals and enterprise strategies, which guide the asking of suitable analytical questions through which the **Analytics Model Plan** is produced. The **Solution Planner** component is based off the **Plan for Analytical Model**, **Plan for Process Reconfig** and **Reconfigure Process** process stages.
- **Analytical Modeler** builds, compiles and trains an **Analytical Model** that is used to adjust the sales forecast that was previously calculated elsewhere. **The Analytical Modeler** receives as the input a wide range of machine learning algorithms **ML Algorithms**, and the instructions provided in the **Analytical Model Plan**, to come up with a design for the machine learning solution, i.e., the **Analytical Model** to solve the business problem. The **Analytical Model** is expected to change and evolve based on context changes. This may require periodically retaining the model, tuning model parameter to ensure forecasting accuracy, or a complete rebuilding of the model. This component was determined by looking at the **Create Analytical Model** and **Train and Validate Analytical Model** process stages.
- **Data Provider** is responsible for retrieving, cleaning, and transforming the raw sales data. This processed data is then used by other components. The **Data Provider** included a wide range of tasks and techniques that pertain to data preparation, including activities such as data cleansing, data transformation, data manipulation, etc. This component requires certain inputs; the first is the actual sales data that needs to be processed, the second is a plan, **Data View**, that provides the data preparation workflows that are needed to select, transform and pre-process the input

data into an appropriate format. The **Data Provider** component is determined based on the **Define Data Preparation** and **Aggregate and Load Data** process stages.

- **Sales Forecaster and Adjustor** component is the component that produced the final predicted sales orders by using the **Analytical Model** as an input. Further, it can also positively or negatively adjust these forecasted numbers based on the previous week's actual sales order. This is necessary as there may have been inaccuracies in previous forecasts (by either over or under forecasting) that need to be rectified in the current forecasting cycle. Finally, the sales forecasts can be adjusted by human users to simulate various scenarios by triggering different user controls. This component is encapsulates functions from several process stages, including **Forecast Future Item Sales**, **Calculate Product Orders**, and **Perform Weekly Analysis**.

7.9 Evaluation

The evaluation of the hiBPM models was performed both during and at the conclusion of the research project. During the course of the case study, periodically team meetings were held where the hiBPM models were presented to show the ability of the hiBPM framework to visualize and analyze portions of the domain that were being studied in that period, with feedback being solicited. These meetings were held approximately every month, however the schedule could vary based on the availability of the team members. The result of these periodic and iterative evaluations guided the next round of study and modeling. Additionally, this researcher would also periodically (i.e. 2-3 times a month) meet the designated contact person from the company's side to review the produced hiBPM model for providing feedback and guiding the direction of investigation for the next iteration.

At the end of the case study, two activities were performed. Members of the company qualitatively evaluated the hiBPM models, including their quality and ability to capture the domain properties to understand the stated problem. Also, a concluding questionnaire (presented in the Appendix) was also filled out by the main contact person from the company's side where the effectiveness of the hiBPM framework was evaluated. This individual was best qualified to evaluate the effectiveness of the hiBPM framework based on their frequently review of the hiBPM models. The questionnaire attempted to find out the effectiveness of the hiBPM framework by asking specific

questions on the ability of hiBPM models to characterize the domain, the usefulness of the hiBPM modeling notations for analyzing the domain, and if hiBPM aided in determining at an optimum design faster compared to a situation if hiBPM had not been used.

7.9.1 Evaluation against Research Objectives

As mentioned previously, the research objective was to evaluate the effectiveness of the hiBPM framework by demonstrating its use in (1) modeling and analyzing interrelated processes, and (2) analyzing the integration of data analytics application in existing processes. A summarization of the feedback received as part of the questionnaire is given below.

Modeling and Analyzing Interrelated Processes: As per the responses provided in the questionnaire, the hiBPM framework was able to capture the essential activities across several software, technology and business processes. This was necessary to be able to understand the overall structure of how the data for the sales order generation was initially retrieved (from the PoS machines at retail stores) and transformed before being stored in data repositories (i.e., data warehouses), to subsequent processes where this data was then used to generate sales orders for future weeks. Of particular interest was understanding the relationship between the processes that review and alter the forecasted sales order numbers, and the processes that adjust orders due to promotion planning.

Overall, the ability for the hiBPM framework to capture and analyze the domain from a process architecture perspective was positively evaluated by the questionnaire respondent. The ability to bring into focus only those activities that are meaningful to the analysis was further appreciated, without having to be burden by capturing all the necessary details for the sake of an accurate domain depiction. Organizing the processes around goal achievement made it easy to comprehend the reason why the processes existed in the first place.

Integration of Data Analytics Solution in Existing Processes: As per the responses in the questionnaire, the organization could visualize design-use relationships where software artifacts could integrate into existing business processes. This was needed to see the changes that would need to be introduced to accommodate a data analytics application in the existing business

processes, and the modifications that would need to happen to the technology processes that provided the data for the data analytics solution. The addition of supporting processes to facilitate the adjusting of the sales order forecasts by the data analytics application through the hiBPM framework was also confirmed.

The software components were presented as designs in the hiBPM model with the design completeness providing guidance to supporting evolvability requirements for those software artifacts, particularly as the organization moved to a different (and final) form of the data analytics solution. Flexibility in process execution was provided through the creation of plans that would guide the execution.

7.9.2 Shortcoming of the hiBPM Framework

While the questionnaire responses confirmed the effectiveness and usefulness of the hiBPM framework in the understanding of the stated problem and determining suitable configurations of the processes to solve them, there were some shortcomings raised as well by the individual when filling out the questionnaire.

A primary limitation was that the hiBPM modeling notation did not provide sufficient expressiveness to help go beyond very abstract software artifact visualization in the models. It was also not possible for the hiBPM models to provide detailed requirements for onwards software systems design. Further, the relationships between the software components were not apparent, and they appeared to be disparate components with just data flows between them. Such limited support for visualizing the software components that are either produced or used in the hiBPM model made the range of analysis difficult beyond the support provided by design-use and plan-execute dimensions. These expressions could not be used for developing requirements for any software application. Further, there was limited traceability of design configuration between the process-side and the software-side, where design reconfigurations introduced on the process architecture side could not be conclusively traced to corresponding changes at the software architecture side. Hence, there was no surety that the determined software architecture was indeed compatible with the design of the hiBPM model.

7.9.3 Learnings from the Case Study

There was a general perception by the questionnaire participant that the hiBPM model was able to determine possible configurations of the business and technology processes that would support the integration of the data analytics solution. The individual further agreed that determining such a to-be state of the processes would have taken longer without the use of hiBPM as the processes themselves were separately designed, with separate departments overseeing the management and design of these processes. The to-be state would have required significant study, including cycles of experimentation and review, to understand how the technology processes would need to change to support the data requirements for the data analytics application, which would then be used in the business processes. The hiBPM model helped iterate through different design alternatives, and to narrow down the possible options to ones that would be more appropriate considering the problems that were to be solved. The participant also appreciated the limited number of hiBPM modeling notations in helping understand the models quickly, including the rationale for including the process elements that are only needed for analysis.

In this case study, not all hiBPM modeling notations were required or used for actual model construction and analysis. Generally, there was a greater emphasis on the hiBPM relational elements (such as design-use and plan-execute, along with the other relationships) due to a need for introducing evolvable software design artifacts and a certain flexibility in the process architecture. Some relationships, like user engagement, were not considered as the software artifacts were assumed to be immutable over time. Another hiBPM construct that was not heavily utilized in the analysis was process phase. The participant in the study assumed that there was some temporal sequencing of activities and moving them out-of-sequence was not necessary.

7.10 Conclusion

In this chapter, we present a case study where the hiBPM framework was applied to a large retail enterprise to understand better how to design the integration of a data analytics solution to existing business processes while considering that both the business processes themselves would evolve, as may the data analytics application.

This study was particularly relevant to this organization as the retail industry is fast-moving, and enterprises in this space are increasingly incorporating data analytics software to help with decision making, sales forecasting and product ordering [190]. The retail domain has high rates of evolution and change, can have sections in the enterprise which change at different rates, can be software-enabled or rely less on technology etc. There is a strong emphasis on enterprise agility and flexibility, customer focus and centricity, data-driven decision making, and automation of business process execution. Despite this, some decision making was still based on practical experience gained through field exposure. A primary reason for this is that it is difficult for the enterprise to capture all the parameters that affect product sales, thus requiring individuals with retail store-level experience to make minor and ongoing adjustments to the calculated sales forecast.

The organization was already using conceptual modeling techniques to capture the design of individual business processes in detail. However, hiBPM was appreciated for its ability to show the relationships between several processes, spanning multiple organizational units, as a single model. This hiBPM model proved useful in capturing alternative hiBPM configurations, highlighting the varying degrees of plan and design completeness suitable to different contexts and situations within the enterprise. This was done using design-use and plan-execute relationships, as these relationships not only show how an analytical application can be introduced within the hiBPM model but also help understand the changes in process architecture configuration that can be possible to ensure flexibility of process execution.

8 Case Study – Cognitive Business Operations

Acknowledgement: This chapter is partially based on the following papers;

- *Babar, Z., Yu, E., Carbajales, S., Chan, A.: Managing and Simplifying Cognitive Business Operations Using Process Architecture Models. In International Conference on Advanced Information Systems Engineering (CAiSE), pp. 643-658, Springer Cham (2019)*
- *Babar, Z., Lapouchnian, A., Yu, E., Chan, A., Carbajales, S.: Modeling and Analyzing Process Architecture for Context-Driven Adaptation: Designing Cognitively-Enhanced Business Processes for Enterprises. In Proceedings of the 22nd International Enterprise Distributed Object Computing Conference (EDOC), pp. 58-67, IEEE (2018)*
- *Lapouchnian, A., Babar, Z., Yu, E.: Designing User Engagement for Cognitively-Enhanced Processes. In Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering (CASCON), pp. 227-233 (2017)*
- *Lapouchnian, A., Babar, Z., Yu, E., Chan, A., Carbajales, S.: Designing Process Architectures for User Engagement with Enterprise Cognitive Systems. In IFIP Working Conference on The Practice of Enterprise Modeling (PoEM), pp. 141-155, Springer Cham (2017)*

8.1 Background and Context

IBM Business Automation Workflow (previously known as IBM Business Process Manager) is a software product offered by IBM Corporation to help with business process design for improving enterprise operations productivity by coordinating activities between tasks performed by (human) knowledge workers and automated software systems [191]. This product is needed to manage the increased complexity of designing and maintaining business processes in large enterprises. The IBM Business Automation Workflow software provides a user interface for creating visual models through which business processes (also referred to as workflows in the product) can be created and configured for a given enterprise. Work activities are sequenced together for execution and delegation in the form of workflows through this user interface. Accompanying tools are used to evaluate the design of the workflows and improve upon them as applicable, based on rule-based automation and process optimization analysis.

Enterprises often use intelligent business process management platforms (of which IBM Business Automation Workflow is an example) to understand and automate key activities in situations where information processing is needed for some business process execution [192]. Business processes often require a human knowledge worker to use tacit knowledge and decision-making

abilities (built over years of experience) to come to a optimum decision when executing a specific activity. An example could be that of a bank employee who is processing a loan application based on provided information, while supplementing the loan information with additional enterprise policies, guidelines and personal experience, before deciding on the approval or rejection of the loan application. Through IBM Business Automation Workflow, enterprise architects and process designers can capture such scenarios, including the information needed for decision-making, the act of decision-making itself, and the outcome of this decision-making.

Recent years have seen rapid advancement in cognitive computing technologies and artificial intelligence that leverage growing volumes of readily-accessible data, increasingly powerful data analytics and machine learning algorithms. The IBM Business Automation Workflow product team wanted to consider how to automate key decision-making activities that presently required human decision-making to one where the decisions were performed using cognitive software systems (also referred to as cognitive agents). By cognitive systems we mean applications that have certain critical characteristics that help differentiate them from other enterprise information systems [193]; these include, being able to function with a degree of autonomy while demonstrating continuous learning behaviour, perceiving events within the surrounding environment and respond appropriately, and showing ongoing adaption in response to evolving circumstances and changing environment. Such tasks performed by these cognitive systems previously relied on human experience and judgement; these are difficult to automate simply using rule-based algorithms or logical operations.

Traditional human-based decision making relies on the intuition and experience of human knowledge workers which are subject to human biases and have variations in the accuracy and speed of decision-making, whereas the expectation from incorporating cognitive agents in existing business processes was the enabling of faster, more uniform, and consistent decision-making despite changing business contextual and situational factors, and staff rotation. There has to be a management of ongoing changes without re-engaging these knowledge workers or undergoing expensive redesigns at both a system and process level. As the enterprise is dynamic, there could be changes in engagements between the knowledge workers and the cognitive agents, while

including self-learning requirements from cognitive agents and managing evolving contexts and adaptation.

A team of researchers from the University of Toronto and IBM Canada participated in a multi-year case study to study the design of cognitive-enhanced business processes. In this project, the phrase Cognitive Business Operations (or CBO) was used to collectively cover the spectrum of enterprise business operations and their involved processes, decision-making activities using insights from available data, and the engagements between business workers and cognitive agents. The term "cognitive" was used in the product as part of the positioning of a broader strategy for this vendor; similarly, the term “cognitive agent” was used to indicate a software system that relies on machine learning and deep learning techniques to aid and assist in decision-making as part of business process execution.

8.2 Case Study Investigation Parts

The case study had three investigation parts. These were as follows,

Requirements for Simplifying CBO Adoption: Understand how to simplify the adoption and integration of cognitive services as part of an enterprise’s routine business operations, thus moving it towards “Simplified CBO”. The challenge in designing and deploying CBO in any enterprise is in how to minimize the overall cost of the solution, reduce the need for individuals with specialized skill sets (such as data scientists and business analysts), and shorten the overall duration of any project engagement. Cognitive business processes do not have simple success criteria as there may be a possible spectrum of processing outcomes by these processes, with these outcomes changing over time as the cognitive agents evolve. Thus, a support structure of processes may be needed to ensure the quality and accuracy of cognitive operations across different conditions, along with minimizing the range of possible changes and reduce the impact and cost of these changes.

Business Process Model with Continuous Improvement: Design a business process model that integrates continuous improvement in the outcome of the business process execution through ongoing training and learning for the cognitive agents. The learning draws on data analytics of past instances as well as dynamic contextual data. The availability of context helps with improved

decision-making and involves both human users and cognitive agents while ensuring continuing satisfaction of enterprise objectives. The ability to redesign sections of the business process, where cognitive agents are integrated to aid with human decision making, in response to evolving domain context is needed to ensure ongoing satisfaction of objectives. Such a conceptual model will support improvement analysis that results from ongoing training and learning.

Cognitive Solution Catalogues and Design Patterns: Develop a knowledge-based method for simpler and faster development and adoption of cognitive agents in business processes. This method would reduce the high cost and skill level requirements of developing and deploying cognitive solutions (due to scarcity of data scientists) through reusable business domain knowledge that characterizes client business problems and objectives, thus overcoming a significant adoption barrier. On the solution side, a method for creating and maintaining cognitive solution catalogues was to be provided where the catalogue contains multiple design patterns for each business problem along with a selection criterion for choosing between the design patterns for meeting the design objectives of the cognitive solution. The design patterns would be solutions to frequently occurring business problems and would be close to being adoptable in CBO with some customization and adjustments, subject to the target environment.

8.3 Objective

The research objective was to evaluate the effectiveness of the hiBPM framework by demonstrating its use in the context of this case study. This objective was carried out through the following,

Use Goal Model to Design hiBPM Model: The NFR framework was used to determine and evaluate key tasks and goals (through a goal model) to consider during any CBO adoption exercise. We needed to validate if the hiBPM model for a target organization could be designed by using goal model(s) as a starting point while also uncovering additional processes that would need to be introduced to the enterprise for supporting Simplified CBO objectives.

Incorporate Context for Continuous Reconfiguration: Changing enterprise context would result in changes to the design of both the business processes and the cognitive agents that involved

in the execution of the business processes. We needed to validate the ability of the hiBPM framework to model and analyze business processes to allow enterprises to better incorporate contextual data into the overall hiBPM process architecture design, including the interactions between these processes and cognitive agents.

Develop Pre-Built hiBPM Design Patterns: Various solution patterns for business process design could be pre-determined that offered a systematic way of attaining different design objectives for Simplified CBO adoption. We needed to validate if hiBPM models could offer visualization of how design knowledge could be expressed in a reusable form that would address a wide range of business problems, particularly around reducing the complexity of CBO adoption.

8.4 Activities

In Chapter 1, we presented a method for attaining research structure and rigour in our case studies with three distinct areas. We detail the specific research activities performed for each of those areas below.

8.4.1 Area 1 – Research Design

The principal architect for the product provided a clear business problem to the research team. This business problem was then decomposed into the distinct parts, as mentioned in the previous section. A research team was formed that consisted of participants from both the university and the company. The university research team consisted of two PhD students, one post-doctoral researcher, and their research supervisor. From the company, the team included the principal architect for the product, the software development manager, and several members of the technical staff. The research project took three years to complete.

Each investigation part was assigned to a primary individual. The first part (defining the requirements for CBO adoption) and the second part (adoption process models with continuous improvement) was a primary focus for this researcher. For the third part (cognitive solution catalogues), there was collaborative work with this researcher focusing on the design of process architectures and another PhD student focusing on the design of the cognitive agent. In this chapter we include the findings from all three parts for which this researcher was responsible. To ensure

that case study findings from each investigation part supported each other, there were periodic in-person team meetings, both with just the university team and with the broader research team.

8.4.2 Area 2 – Data Collection

Research activities were defined for data collection early in the case study. The research team (with participants from both the university and the company) usually met once a month, based on need and research progress, to review the stated problem and to apply constructs from each individual conceptual modeling approach (based on their respective areas) to see how the models could help better build understanding of the stated problem.

Apart from the initial problem document, additional documentation was provided that helped with a better understanding of the product; these included high-level product design documents and architectural diagrams. Access to the product was also provided for allowing the university researchers to experiment with and understand the product's existing features and capabilities. These documents (including the produced conceptual models and meeting notes) were stored in a company-provided online cloud storage platform that was used by the entire research team for project collaboration.

8.4.3 Area 2 – Data Analysis

Notes were taken after each meeting for later analysis and record keeping. We used logical chain of evidence to establish a connection between (a) the research problem presented earlier in this chapter, (b) to uncovering the requirements for Simplified CBO for an enterprise, and (c) to determining the hiBPM modeling constructs that would help us in analyzing the particular situation under study. This was an iterative exercise as each uncovered requirement result in a hiBPM model snippet with the models then ideated and subsequent models being produced, and their effectiveness determined.

This process was repeated for each of the main investigation parts of the case study that this researcher was responsible for, and the results shared with the broader research group in the periodic team meetings. The meetings were often held in-person at the IBM Software Labs office in Markham, Ontario. On occasion, the meetings would be held virtually if some individuals could

not be present in-person. On project conclusion, a case study report was shared with the senior team member (and main contact person) from the company for confirmation of its accuracy.

8.5 Understanding Simplified CBO

To illustrate and discuss some key concepts of this case study, we consider an enterprise about to adopt cognitive computing capabilities to support its routine business operations, thus moving towards Cognitive Business Operations (CBO). A simple repetitive business process, loan application process at a financial institution, was selected by the research team that could be used for analysis purposes. Key decisions in this business process are made by human knowledge workers based on their experience and information presented to them by information systems. In a cognitive-enhanced business process, a Cognitive Business Agent (CBA) (also simply called the cognitive agent) leverages recommendation systems (also known as recommenders) [194] to assist humans knowledge workers with decision making. A recommender is a software system that tasked with providing advisory services and personalized product recommendations. Recommenders have gradually evolved from generating non-personalized to increasingly personalized recommendations by adopting progressively more complicated approaches, ranging from probabilistic and mathematical modeling to machine learning and deep learning algorithms. In our case, the cognitive agent utilizes a recommender to provide an action (e.g., “approve loan”, “reject loan”, etc.) to the human worker. Increasingly sophisticated cognitive agents can initially help their human workers with recommendations for key decisions and later take complete responsibility for decision making, without the knowledge workers involvement. Usually, recommendation personalization is provided to the end-user (who may also be the enterprise customer) but in this motivating example, it is for the company staff who are responsible for or involved in the repetitive business process operations.

In our experimental system, the recommender considers evolving (business and technological) contextual and transactional process data before generating a recommendation. The analysis of that experience identified several design challenges about the recommender’s adoption and use in enterprise business processes. These include:

- Changing business and/or regulatory environment. The recommender needs to detect and incorporate business parameters while generating recommendations during process execution. This way, recommendations are more aligned with enterprise objectives, despite these objectives evolving.
- Retraining recommender models. The recommender needs to be retrained when there are significant changes in data, such as switching from batch to streaming data, shifting data trends, or data availability (e.g., whether customer credit scores or customer profiles are made available for loan decisions). These result in ongoing relevance of recommendations.
- Decreasing accuracy of the analytical model. Various measures, such as precision, recall, etc., are used to measure model accuracy. Accuracy below some threshold results in the recommender model being retrained (with new datasets or different parameter settings) or being rebuilt (utilizing a different set of algorithms or approaches).

The above situations align with realistic business situations where changes in business, technology or data context invariably result in changes in how the cognitive agent is designed and utilized in the business process. Here, both the knowledge workers and the cognitive systems need to adapt and adjust to each other as progressively more advanced and accurate advisors take on increased responsibility and autonomous decision making.

8.5.1 The As-Is Situation for CBO Adoption

We present the As-Is hiBPM model of the loan approval scenario in an enterprise that is adopting Cognitive Business Operations (CBO), with the loan approval/rejection decision being the focus of analysis. As explained previously, this business process was selected by the research team as an example representation of the business processes that the company's product would be expected to optimize for Simplified CBO. In the As-Is situation, the enterprise relies on highly skilled individuals for the adoption of cognitive capabilities and utilizes a traditional analytical model-based solution. In Fig. 8-1, the domain-specific process stages **Process Loan Application**, **Setup Loan Repayment**, and **Repay Loan** are centrally shown as process stages, with the cognitive system-specific process stages **Create Analytical Model**, **Tune Analytical Model**, and **Validate Analytical Model** producing the software artifacts necessary for enabling cognitive decision making. **Process**

Loan Application contains a single process phase that pertains to the collection of decision-making process elements that produce and present a recommendation.

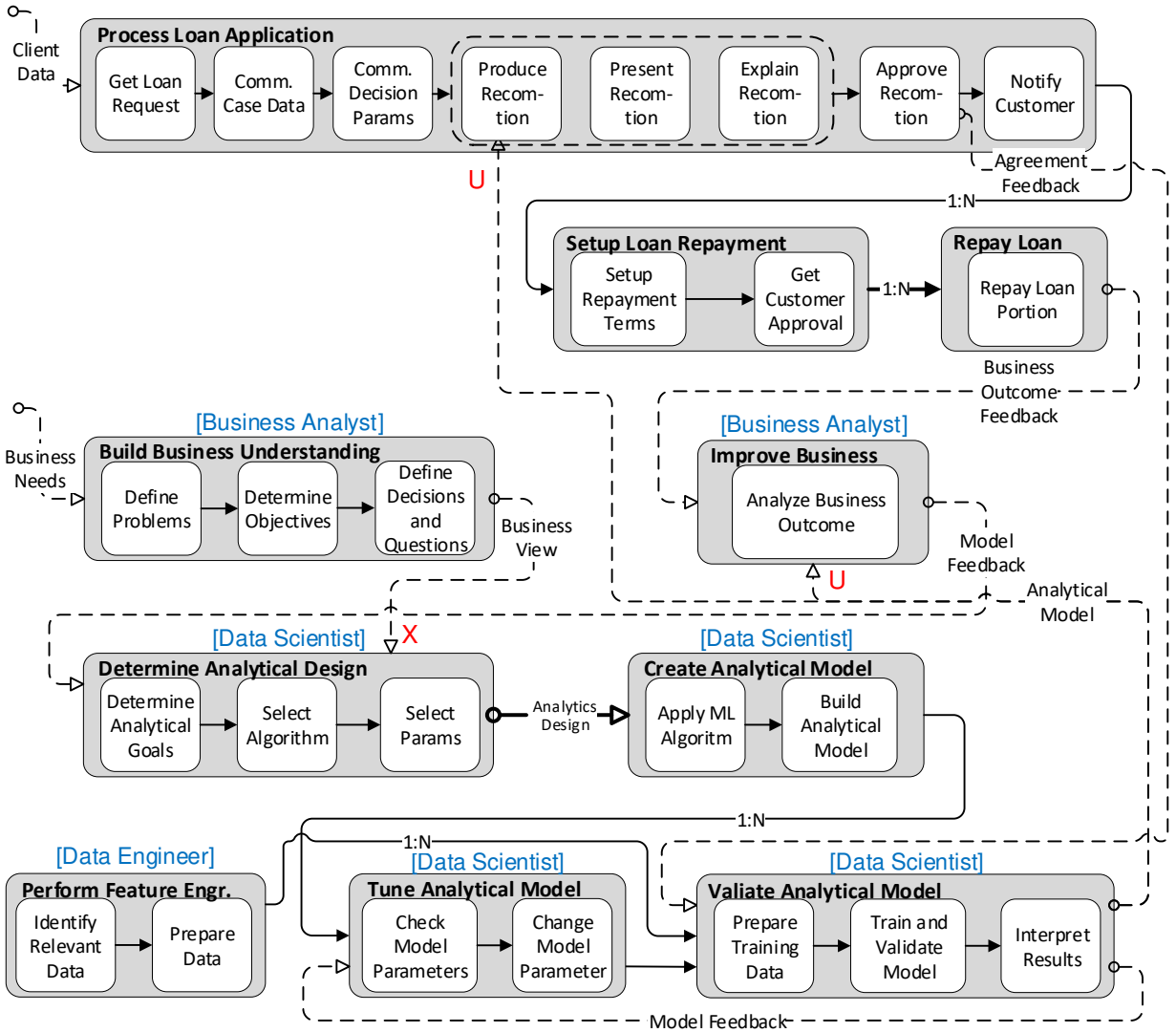

Fig. 8-1: As-Is Process Architecture Model for the Loan Application Process

The domain-specific business processes rely on different systems artifacts to help with processing a loan application. **Analytical Model** is a design artifact that is "used" as part of the **Process Loan Application** processing. Thus, there exists a design-use relationship between the **Validate Analytical Model** and the **Process Loan Application** process stages. **Build Business Understanding** produces a plan that is executed by the **Determine Analytical Design**. Similarly, there exists a plan-execute relationship between these two process stages. While not part of the hiBPM modeling notation,

some process stages are annotated with process participants responsible for their execution to indicate different domain actors and their involvement. Through these annotations, we emphasize the need for highly skilled individuals for any implementation project. Several process stages require the involvement of **Data Scientist** who performs manual operations to execute the activities within that process stage; this incurs time and cost, with the **Analytical Model** produced generally being custom developed for a specific environmental context.

8.5.2 Analyzing CBO using the NFR Framework

We use the NFR Framework [94] for determining the essential softgoals that are relevant to the particular enterprise adoption of cognitive agents in their existing business processes. Each enterprise may have different softgoals based on their definition of Simplified CBO. E.g., some enterprises may prioritize ease of deployment whereas others may prioritize the minimum maintenance of the overall system. These identified Simplified CBO softgoals for an enterprise can then help with determining a suitable hiBPM model for that enterprise, we can characterize the CBO adoption of our target enterprise using the following softgoals. An explanation of these softgoals is also provided, as they apply to the target enterprise.

- **(High) Learnability:** Learnability implies feedback and higher-order analysis of options by applying previous experience. Considering this, business operations powered by cognitive agents should be dynamically reconfigured to respond differently based on learnings from prior instances. The changes in response could take many forms, from how knowledge workers engage with cognitive agents, to how to respond to external stimuli, etc.
- **(High) Reusability:** A fundamental premise in cognitive solutions is the ability to have reusable knowledge artifacts that provide best practices and patterns-based solutions to commonly occurring problems. Such a reusable knowledge base is built over time in the form of knowledge catalogues with the cognitive solution being designed to leverage these to reduce the effort of solving known problems and handling situations.
- **(High) Configurability:** Ongoing reconfigurations of any cognitive-enhanced enterprise solution to support evolving enterprise requirements and changing circumstances. Thus, the ability to reconfigure aspects of the solution (such as the processes, the systems, or the

engagement between the knowledge worker and the systems) is essential to reduce continuous project cost and human involvement at both deployment and post-deployment time.

- **(High) Developability:** Any cognitive-enhanced solution needs to be customized to intelligently handle different environments, requirements, and changing circumstances. The product should allow for some form of development to extend, enhance or modify functionality after it has been released. Due to the variations in the business and technical requirements and contexts for cognitive agents, it is impractical to identify all possible configurations required for different organizations.

There are two possible mean to achieve Simplified CBO objectives which are presented as two alternatives using the NFR Framework. While each alternative will satisfy the CBO functional goals, there may be trade-offs when it comes to satisficing the softgoals. These alternatives (represented as Alternative A and Alternative B) are represented as OR decompositions in the goal model (shown in Fig. 8-2). Here, we make slight adjustments to the goal modeling notation provided by the NFR framework to show how softgoals can be attained by goals and tasks, including the resources needed. The goal model was developed by this researcher based on information provided by researcher members from the company and was used as a method for determining the relevant softgoals for the enterprise example.

Alternative A shown is for the As-Is situation with the enterprise attaining its functional requirements for CBO by utilizing a traditional analytical model-based solution. The softgoals for Simplified CBO are decomposed down to their operational goals with the roles required for their attainment annotated in blue. For example, we see that a **Data Scientist** (shown through annotations in the goal model) is responsible for several operationalized goals, which include **Select Algorithms**, **Develop Analytical Models**, etc. Similarly, the **Business Analyst** is responsible for studying the enterprise space and helping **Develop Business Solutions** for that particular organization. These collectively help convey the significant involvement of these high-skilled individuals in the overall project activities, which prevent the attainment of the softgoals of **Learnability** and **Reusability**.

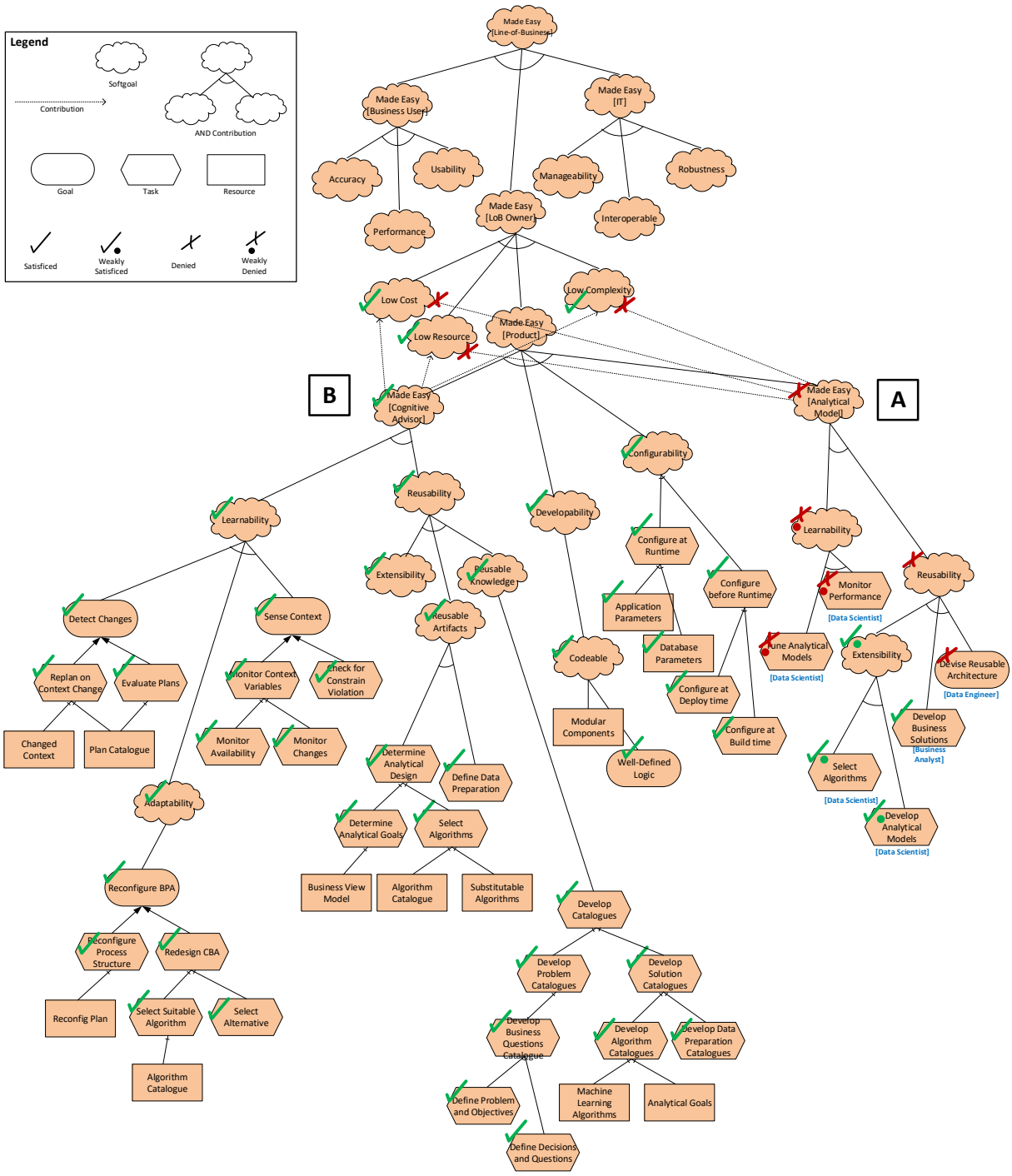

Fig. 8-2: Goal Model showing two alternatives for attaining Cognitive Business Operations

We use a goal satisfaction analysis technique [172] to qualitatively assess whether the softgoals can be satisficed (✓) or denied (✗). Weakly satisficed or weakly denied conditions are shown using

the same symbols, but with a "dot" added. As can be seen in Fig. 8-2, the primary softgoals of Simplified CBO for the Line-of-Business is not satisficed; thus Alternative A is not ideal, despite attaining the functional goals.

The branch marked Alternative B in Fig. 8-2 pertains to the incorporation of a cognitive agent that helps attain the softgoals of **Learnability**, **Reusability**, **Developability** and **Configurability**. The cognitive agent leverages recommenders to aid and assist with decision making (e.g. loan approved, loan rejected etc.) for the knowledge workers. The cognitive agent alternative is qualitatively analyzed against the same set of softgoals; however, the solution provided in this alternative contributes to the softgoals differently. The **Reusability** softgoal comes at the cost of runtime flexibility. The selection of either one is made based on the priority and preference of the enterprise (as ascertained through goal analysis). E.g. one enterprise may feel that there are no unpredictable situations expected to occur in the future, and thus, **Pre-Built Catalogues** (limited in scope as they may be) would suffice. Another organization may be uncertain with regards to changing situations and would wish for **Data Scientists** to be engaged to populate the **Algorithm Catalogue** until a stable state is achieved. In order to achieve **Reusability**, there is a reliance on creating reusable knowledge nuggets and artifacts as part of pre-deployment activities. **Learnability** is achieved through ongoing **Sense Context** and **Detect Change**, based on which suitable actions are performed to process context changes and selection configurations.

For both Alternative A and Alternative B, **Configurability** is managed at runtime and pre-runtime respectively through having configurable settings. By runtime, we mean reconfiguring the cognitive agent (or **Analytical Model**) to behave differently during the execution of business operations, which can be done through changing application or database parameters. Pre-runtime configuration is achieved through configuration settings done at either build or deploy time. Finally, **Developability** of the product is attained through having modular components, no-code / low-code and codeable architecture for extending product features.

By identifying and prioritizing the softgoals for the domain under study, we devise a to-be solution that encompasses the interplay of systems, processes, and user engagements. Such a solution (a) reduces the cost, time and complexity of integration of cognitive agents in business processes, and

(b) minimizes the process reconfiguration and systems reimplementation as the enterprise environment changes.

8.5.3 The To-Be Situation for Simplified CBO

We come up with the to-be hiBPM model shown in Fig. 8-3 by following the steps mentioned in Chapter 4 for determining a hiBPM model from a goal model. In this figure, we show both the domain-specific process stages for the **Loan Approval** process, as well as surrounding and supporting process stages that help attain the functional goals and the softgoals for Simplified CBO that were determined through the goal model; both sets of process stages are reviewed below. This To-Be hiBPM model is provided for the target enterprise based on the initial As-Is scenario and softgoals provided by the different members of the research team.

We **Reusability** and **Adaptability** softgoals to present the additional process stages that emerge out of the analysis for ensuring the satisficing of these softgoals. The **Reusability** softgoal was achieved by having reusable artifacts and reusable knowledge nuggets. Particularly, reusable knowledge is attained through having **Solution Catalogues** that provide the necessary patterns for quickly determining the solution to a commonly occurring business problem. We show the creation of this catalogue by the process stage **Develop Solution Catalogues**. Another means for attaining the **Reusability** softgoal was by having reusable artifacts that are pre-built and quickly incorporated into the analytical solution as and when needed. These reusable artifacts are built by the **Determine Analytical Design** process stage.

Similar to the process used above for the **Resuability** softgoal, the **Adaptabilty** softgoal can be achieved by introducing additional process stages to the domain-specific business process. To have **Adaptabilty** in the system, both the process architecture and the cognitive agent needs to be amenable to redesign. Further, there should be higher level processes that plan for reconfiguring the process architecture and the cognitive agent when the enterprise context changes. For this, we introduce the **Reconfigure Process Structure** and **Redesign Cognitive Agent** process phases in the **Reconfigure Cognitive Operations** process stage. These (the overall hiBPM model for enabling these softgoals) collectively come together as the cognitive agent solution that encompasses both process-level and systems-level reconfigurations.

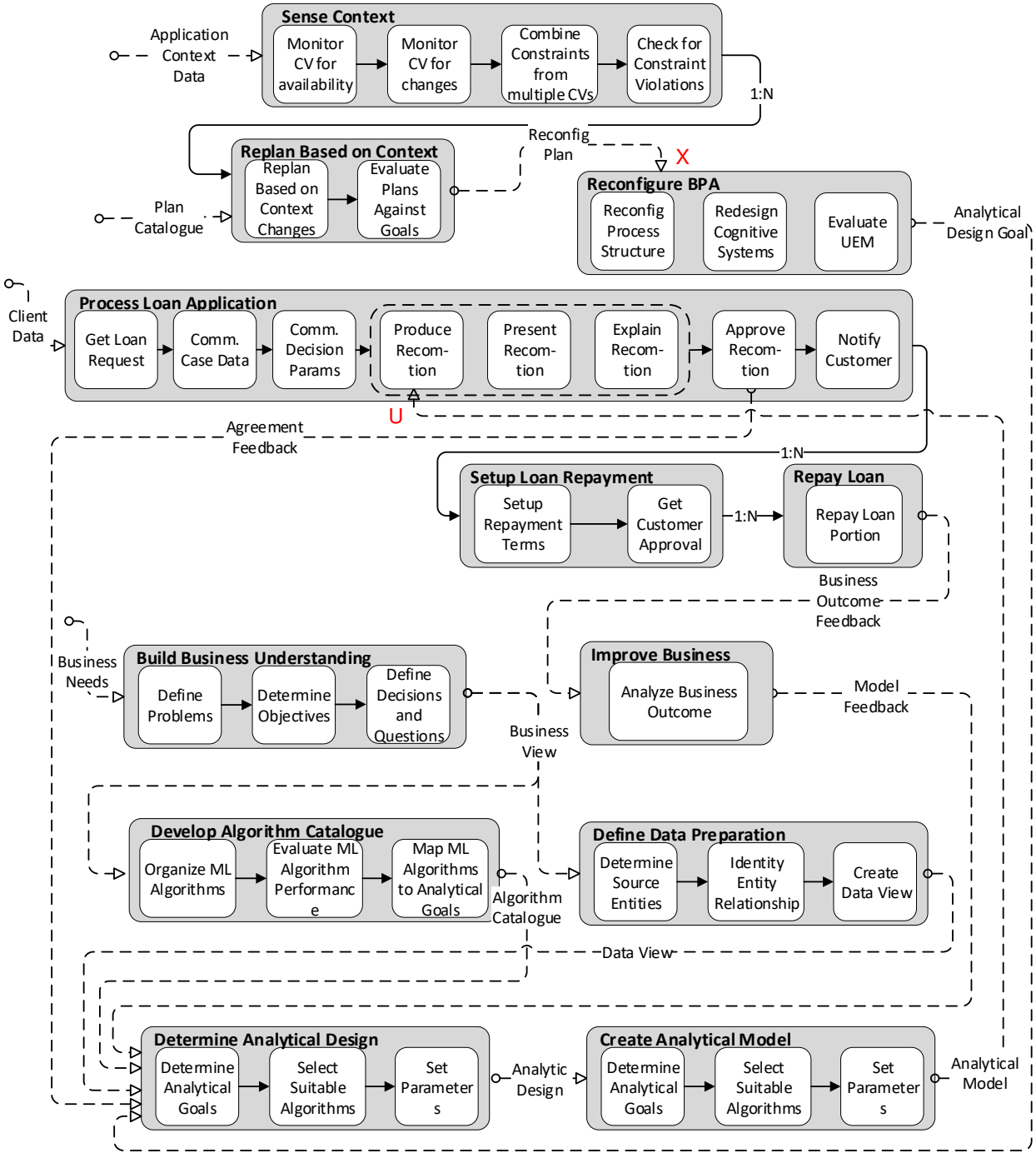

Fig. 8-3. To-Be Process Architecture Model for Simplified CBO

8.6 Context and Adaptation

In the previous section, we mentioned that the cognitive agent needs to be designed to operate under evolving circumstances and varying contextual situations. The need to continually identify, monitor and sense context can result from (a) the changing of enterprise or process goals resulting in context changes (as it pertains to the attainment of those goals), (b) non-availability of the information pertaining to domain entities, and (c) the changing values of domain entities' attributes beyond a certain threshold.

In this section, we consider the need to monitor and manage external context, and how changes in the context in one area of the model can influence a hiBPM redesign in another area. We consider context as entities (including their attributes) of the domain under study (or some particular portion thereof) that are relevant to the attainment of particular enterprise or process functional or non-functional objectives. Considering the complete domain context may be not possible or necessary, therefore a more localized context is selected to consider it's impact on hiBPM model reconfiguration. We consider both the processes that need to handle context in the hiBPM model, and also the processes that need to respond to context stage changes. Thus, the entire hiBPM model needs to be holistically analyzed as we determine how ongoing changes in context sensed in one part of a model can result in discrete redesigns of related cognitive-enhanced business processes in another part. Further, it would be unrealistic to expect that the relevant context, once identified, does not need to be re-evaluated later, therefore these analyses need to be periodically performed, as and when context changes.

Suppose in our example domain, we want to ensure that the hiBPM model needs to be designed in a manner that ensures corporate policies (with regards to customer satisfaction and accuracy) are suitably satisfied in the execution of the operational business process. In this case, the context would be limited to customers (and the processing of customer service requests) while other areas of the domain can be ignored. As before, this example was developed through discussions with the broader research team and was designed to reflect actual environments that the company's product would be deployed into.

8.6.1 Modeling and Analyzing Context-Induced Reconfigurations

We consider an initial stage of the cognitive agent adoption in a business process at an enterprise. When the business processes are first setup and operations start, detailed data about customer behaviour and profile is generally not available and the cognitive agent relies on collaborative filtering approaches for recommendation generation [194]. Collaborative filtering is an approach for implementing recommender systems where the cognitive agent would provide non-personalized recommendations in the absence of customer profile information. As additional customer and transactional data are accumulated, profiles may emerge for the customers, and it may be possible to generate more customer-specific recommendations using personalized approaches in the cognitive agent implementation. This requires the selection of different machine learning techniques that provide more personalized and context-based recommendations [194]. A reconfiguration to the process architecture is also needed whenever this additional contextual state is made available.

We adopt the method of determining context based on goal model analysis originally proposed in [79] for our research. The sequence of reasoning and analysis steps to manage a hiBPM is presented below.

Step 1 - Determine Domain Goal: Domain goals need to be identified before any context can be studied. This is important as business goals can evolve and this situation necessitates reconsidering of what context should be captured and studied to ensure the satisfaction of these evolving goals. Fig. 8-4 shows a goal model that emphasizes the need for personalized loan application recommendations for customers. The model decomposes the functional goal **Get Loan Recommendation** into sub-goals using OR refinement. Here, the customer profile is the context that is of interest, and based on its availability or non-availability, either one of the two sub-goals is the more suitable alternative. In Fig. 8-4, the **Customer Data** contextual tag is shown as a blue annotation on the **Personalized Recommendation** goal alternative, thus indicating that this alternative will not be available in the absence of **Customer Data**.

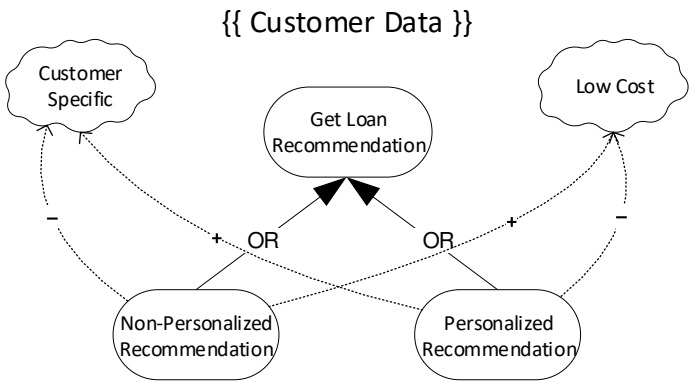

Fig. 8-4: Goal model for the Loan Application process

The activities involved in determining the business goals in Step 1 are represented in a process stage called **Determine Goals** with several process elements as shown in Fig. 8-5 below.

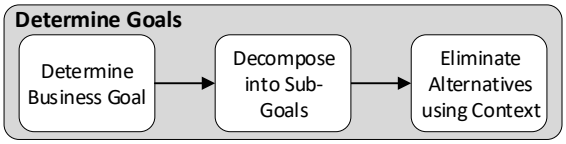

Fig. 8-5: Process stage for determining business goals

Step 2 – Identify Context and Contextual Variables: The required context and associated contextual variables can now be identified using either ER diagrams [195] or UML class diagrams [196]. We previously stated that context is considered as domain entities, and their attributes help in the attainment of applicable enterprise goal. Certain aspects of the domain are not pertinent to the attainment of enterprise goals and do not need to be considered further. In our case, the goal is to provide personalized recommendations to help with a customer's loan application decision and we consider just those domain entities that are part of processing the customer loan.

Fig. 8-6 shows an ERD with four entities **Customer**, **Loan**, **Loan Officer**, and **Bank**, with selected attributes for each. Monitoring the context can be done using context variables, which are the attributes of these entities that can be sensed and can help in the attainment of the selected enterprise goal.

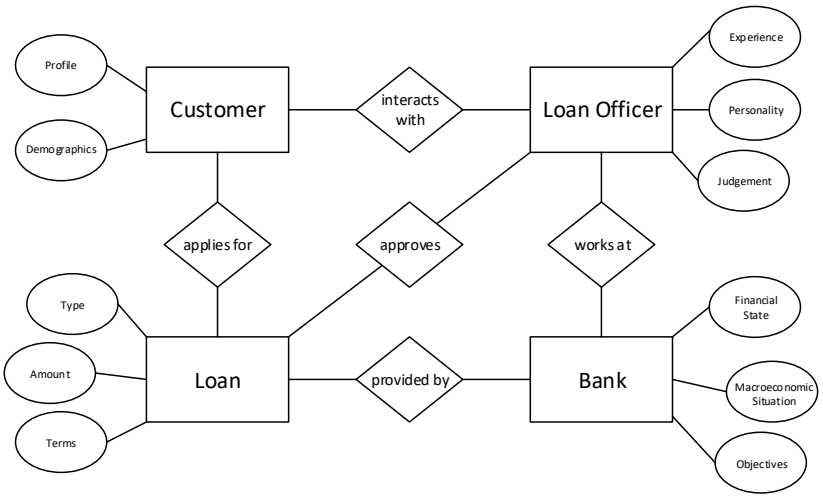

Fig. 8-6: Context and context variables selection using ERD diagram

We represent the activities in Step 2 as a process stage **Identify Context Variables** with the individual activities represented as **Identify Entities to Achieve Objectives**, **Identify Attributes of Entities**, and **Select Variables from Attributes** process elements.

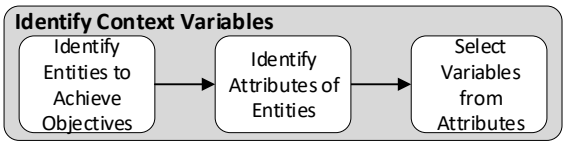

Fig. 8-7: Process stage for determining domain context variables

Step 3 – Analyze Context Variables: Capturing, monitoring and analyzing context variables incur some cost to the enterprise and it would be preferable only to identify those context variables that are relevant for the enterprise goals. Further, context variables may not always be available to be sensed and their unavailability should also be considered during this analysis step. Thus, only those context variables that are available and contribute to the attainment of domain goals and softgoals should be monitored.

In the previous step, we identified various context variables **Customer Demographics**, **Customer Profile**, **Loan Type**, **Loan Amount**, **Loan Terms**, etc. as having relevance to the **Personalized Recommendation** goal. By following the multi-step analysis process provided in [79], we can reduce the context variables of interest to just one (**Customer Profile**) in this case. Based on the outcome of this qualitative analysis, the **Customer Profile** context variable is deemed to be (in

comparison to the other variables) providing customer-specific recommendations for their loan applications. The process stage **Analyze Context Variables** shown in Fig 8-8 below contains several process elements that present the activities that need to be performed in this step.

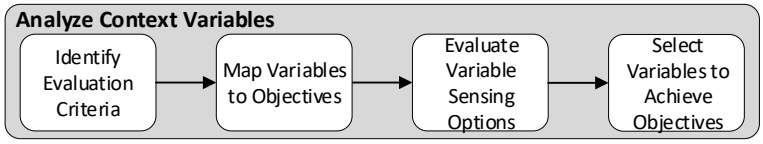

Fig. 8-8: Process stage for identifying context variables and their applicability

Step 4 – Monitor Context Variables for Availability and Change: In this step, we first consider the validity and availability of this context variable. By availability, we mean that not only should the context variable be available in the real world, but it should also be available to be sensed. Further, the validity of the context variable means that its value is an accurate reflection of real-world data. As previously mentioned, there is no guarantee that the **Customer Profile** would always be available, and, indeed, it is only after the accumulation of customer data and transactional history, are machine learning techniques able to intelligently create profile clusters of customers based on common behavioural attributes. Also, customer profiles already calculated may become invalidated because of a variety of reasons (e.g., corporate policy changes, economic environment changes, updated government regulations, etc.).

In the **Sense Context** process stage, contextual variables at suitable sensing points are continuously monitored for breaches to defined parameter thresholds or state changes while evaluating the satisficing of relevant softgoals. For example, there may be a revision in how **Customer Profiles** are calculated, rendering the previous values stale. Such a change needs to be monitored and processed. Fig. 8-9 shows that external **Application Context Data** is communicated to the **Sense Context** process stage. Context is, by definition, external to the system under consideration, therefore, feedforward paths are better suited to show how external context changes can be used to pass a control signal to a system. We show **Application Context Data** as an input as that is a more generic term and covers different specific contexts. In this example, the **Application Context Data** is the **Customer Profile**.

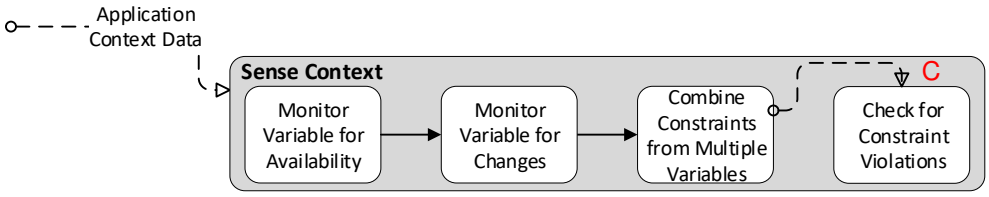

Fig. 8-9: Process stage for monitoring the availability and validity of context variables

Step 5 - Maintaining Pre-Build Plan Catalogues: Pre-built plans exist as a catalogue of design patterns that help guide the selection of various alternatives. These are for situations that manifest themselves as the context evolves. For example, having **Customer Profile** available means that the cognitive agent (and the surrounding processes) should be configured to take advantage of this changed context. In our example here, we can consider the catalogue consisting of two plans, one corresponding to **Non-Personalized Recommendation** with the other for **Personalized Recommendation**. These plans would have been inserted in the catalogue by some other processes at an earlier time, i.e. before an enterprise engages in a Simplified CBO adoption exercise. In Fig. 8-10 below, we show how pre-built **Plan Catalogue** can be used for determining the replanning of the process architecture, the software system, or the user engagements (discussed in Step 7). Design catalogues are discussed in more detail in the next section of this chapter.

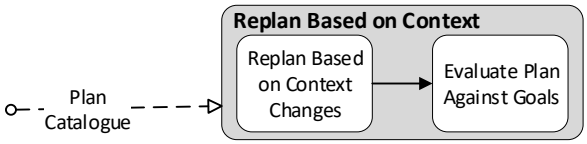

Fig. 8-10: Process stage for replanning the solution using pre-built plan catalogues

Step 6 – Determine and Select Suitable Plan: The availability (or change in state of) **Customer Profile** is communicated to the **Sense Context** process stage, which then informs the **Replan Based on Context** process stage, as shown in Fig. 8-11. This stage utilizes pre-built plans from the catalogue and evaluates them against the business goal while considering the new state of the context variables. The stage selects the plan corresponding to **Personalized Recommendation** and passes it along to the execution stage where the reconfiguration is carried out at a variation point. Hence the plan informs the variation point of this situation and guides the follow-up action using data passed through feedback paths. Triggers are also present at the **Replan Based on Context** and

are “fired” once non-satisficing of softgoals is determined and the appropriate plan selected. The **Customer Profile** is shown as a flow in Fig 8-11 (generalized as **Application Context Data**) into the **Sense Context** process stage, while the pre-built **Plan Catalogue** is a resource enabler and thus is shown as a flow into the **Replan Based on Context**.

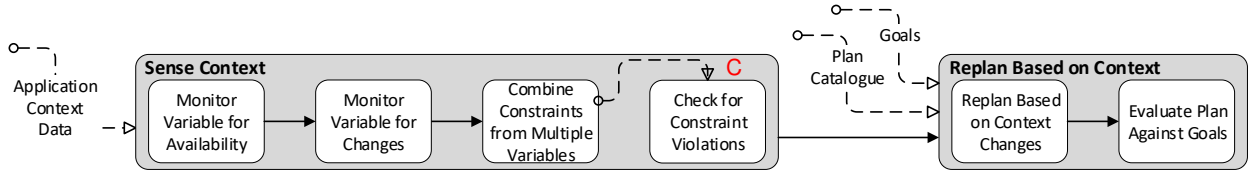

Fig. 8-11: Process stages for determining and selecting suitable plans for adaptability

Step 7 – Reconfigure Process Architecture: The execute stage receives the plan and is now responsible for reconfiguring the hiBPM model to handle the context stage change better, and (by extension) the continuing satisfaction of enterprise goals. Here, reconfiguring the hiBPM model can mean one or more of the following,

- Rearranging the structure of the process stages in the model, including their relationships along various change dimensions.
- Redesigning cognitive agents that are used as part of the business process execution so that they provide different results on subsequent use.
- Changing the nature of engagements between the systems and the business process.

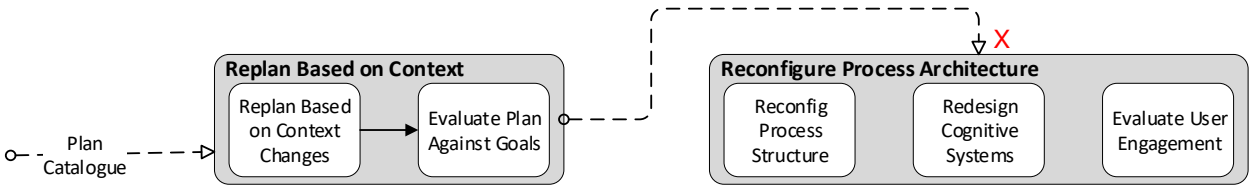

Fig. 8-12: Process stages in a plan-execute relationship for reconfiguring the process architecture

The hiBPM model reconfiguration results from the execution of the plan, with the plan guiding the selection of suitable variants. This is shown in Fig. 8-12 where the as-is process architecture configuration is geared towards providing non-personalized recommendations whereas the to-be configuration provides personalized recommendations for use in the business process. The **Replan**

Based on Context informs the execute process stage **Reconfigure Process Architecture** to now generate personalized recommendations.

8.6.2 The Complete hiBPM Model

We can now show the complete hiBPM model for the enterprise in Fig. 8-13. Here, the enterprise may have a defined corporate policy with regards to processing loans for customers who cannot be matched to an individual pre-defined profile, possibly because of insufficient customer data. When starting, the company may take a more conservative approach towards loan approval and refuse to process such customer cases. However, over time, it may revise its corporate policy and start to process them differently. Thus, there is a change in the enterprise goal, i.e., **Process Non-Profile Customers**, and the additional context **Customer** needs to be incorporated. With sufficient transactional history, customers may now be adequately profiled and thus, an additional context **Customer Profile** is now available for the enterprise to use as part of the loan application process. Attributes of the context (for example, the information on when the corporate policy was changed) can also be monitored.

Managing context requires the addition of meta-level processes for sensing and retrieving contextual data. The monitoring of context at sensing points is done to ensure that associated sensing goals are continuously being attained, including confirming their validity, applicability, and suitability. In the case of non-attainment, a plan is selected and executed from a catalogue of pre-defined plans. Specifically, the process stages about context management are **Identify Context Variables**, **Analyze Context Variables** and **Sense Context**, and they confirm that the availability of and changes to **Customer Profile** data will be detected and specific actions are taken based on the satisficing of associated softgoals.

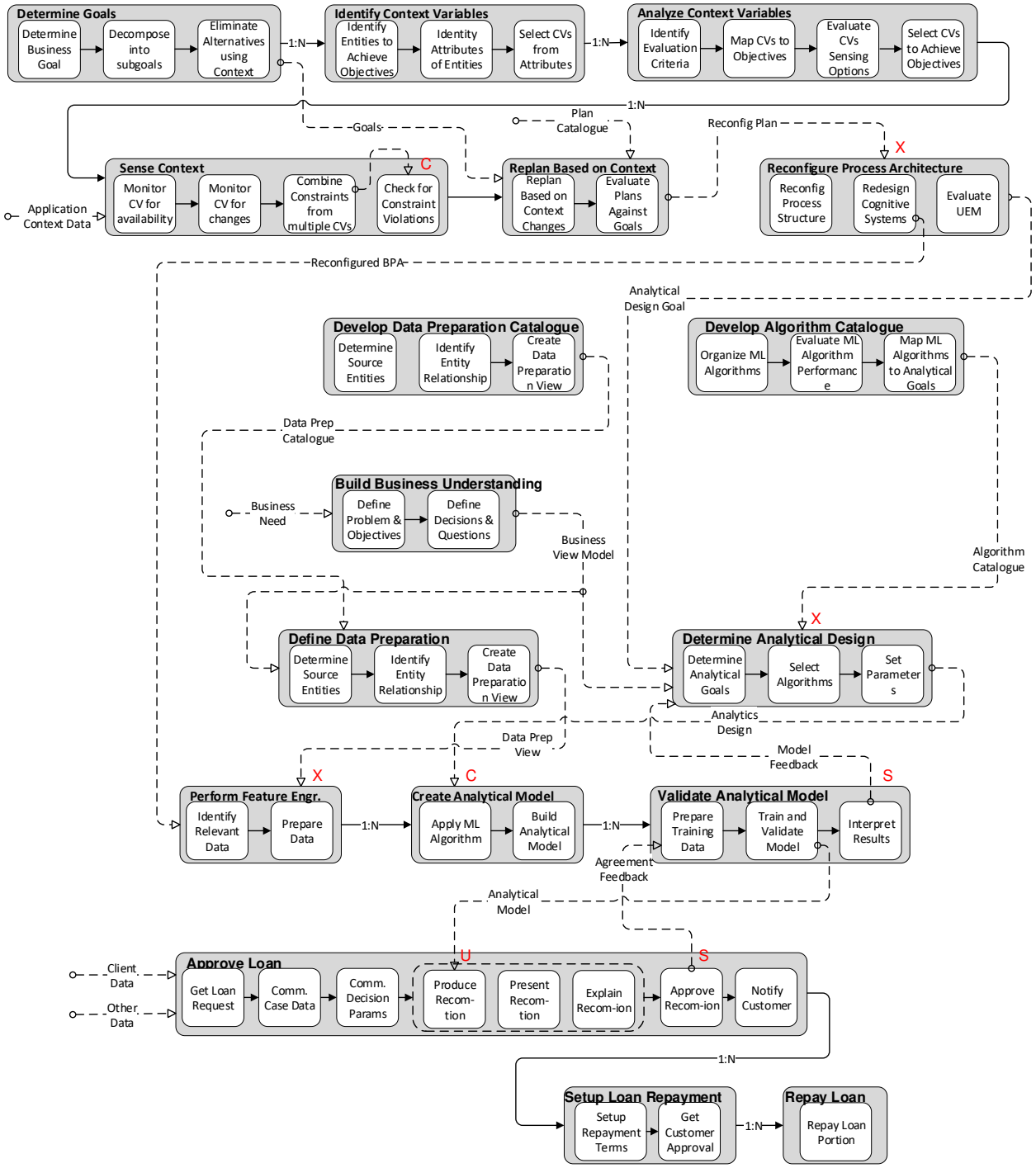

Fig. 8-13: hiBPM model for context based adaptation for the loan approval domain example

Perform Feature Engineering process stage now actively collects customer data so that **Customer Profile** can be calculated through some separate process stage using machine learning techniques

and continuously updated. The **Determine Analytical Design** process stage selects a different recommender approach, specifically for personalized recommendations, and machine learning algorithms. This is the **Analytical Model Design** that is subsequently used by the cognitive agent to generate recommendations. How the cognitive agent is engaged in the **Loan Approval** business process execution could also have changed; instead of playing an advisory role, the cognitive agent can automatically approve or deny the loan application without any human oversight. The shift in engagement between the cognitive agent and the human knowledge worker is represented through user engagement relationships.

8.7 Design Catalogues and Patterns

Some enterprises may be cautious in relation to cognitive technology and automation. To simplify the adoption of cognitive agents for such enterprises, essential types of business problems can be analyzed ahead of time and a relevant set of identified design patterns be reused by multiple enterprises as applicable. Such a knowledge-based approach helps in reducing the high cost and high skills requirements for developing and deploying cognitive solutions, thus overcoming a significant adoption barrier. As a result, solution catalogues consisting of sets of reusable design patterns are pre-built before the solution is deployed at a target enterprise. The solution is then adopted with instantiated patterns that are identified as part of the solution analysis activity. These patterns would be particular to each enterprise environment and would be selected accordingly. Here, a design pattern is in the form of a conceptual model that provides guidelines on solving particular enterprise problems as they exist in some business situations. As part of this research, we explicitly consider hiBPM model patterns. Other research threads within the CBO research project considered design patterns from a cognitive solution design perspective. Our approach of developing and using patterns to solve design challenges is certainly not unique, and examples of various approaches can be found in different research disciplines, such as business processes [197][198], enterprise architecture [199], requirements engineering [200][201] and software engineering [202][203].

In this section, we present several different design patterns against common business problems that we encountered during our research team deliberations. These again pertained to the design

and adoption of the cognitive agent at the example enterprise. The objective was the instantiation of these hiBPM models (as provided by the solution catalogue) to reduce the dependency on individuals with specialized skills (such as data scientists, data engineers, and business analysts) during the deployment of the solution. The patterns discovered as part of this case study were found to be either generalizable enough to cover a range of enterprise adoption scenarios or specific to certain situations that would limit their reusability. We share examples of both in the following sections.

We start by providing a simple baseline, i.e., the loan application process presented in the previous section. There may be many different types of service request types that need to be processed. The human knowledge workers processing these requests may have different capabilities and skills, with some requests being processed more efficiently by some workers than other requests. The manager of the knowledge workers thus needs to make a conscious and deliberate choice on which service request should be assigned to which worker, based on the need for higher overall team efficiency. This is done by the process stage **Assign Request to Worker**. We show a hiBPM model for this situation in Fig. 8-14.

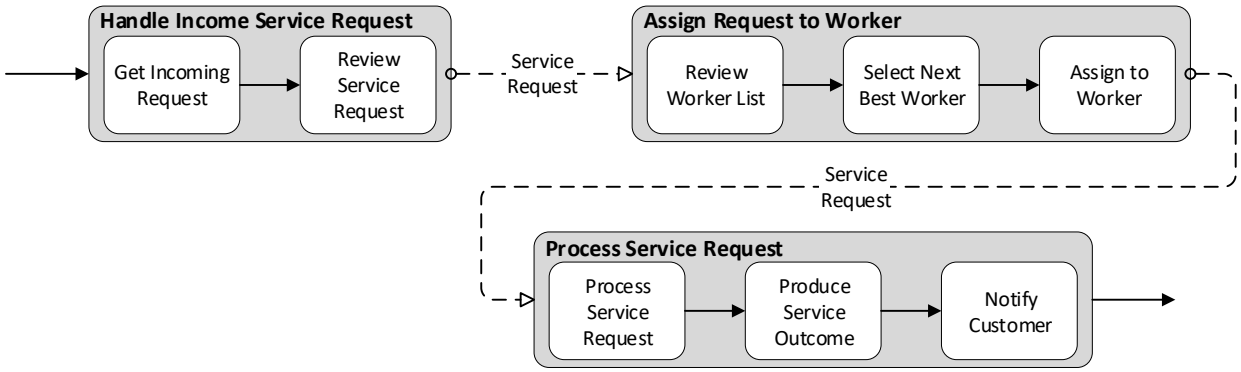

Fig. 8-14: hiBPM model for the baseline business process

8.7.1 Learnability with Control

We consider the case of where the manual activity of selecting the most suitable knowledge worker **Select Next Best Worker** is now performed by a cognitive agent. Here, the business problem is the considering the suitability and applicability of a knowledge worker to a service request type; a suitable worker is determined through reviewing the past performance of all knowledge workers

against each type of service request. This involves mining process logs for historical execution instances for determining insights where each knowledge worker is ranked based on their ability to process certain service request types. This processing is represented by the process stage **Mine Process Execution**.

These performance insights (by knowledge worker) are then made available to the process stage **Develop Cognitive Agent** that develops and trains the cognitive agent to offer a suggestion on the most suited knowledge worker for the incoming service request. Fig. 8-15 below shows a hiBPM model with the design pattern for the mining of data needed for support learnability for the cognitive agent.

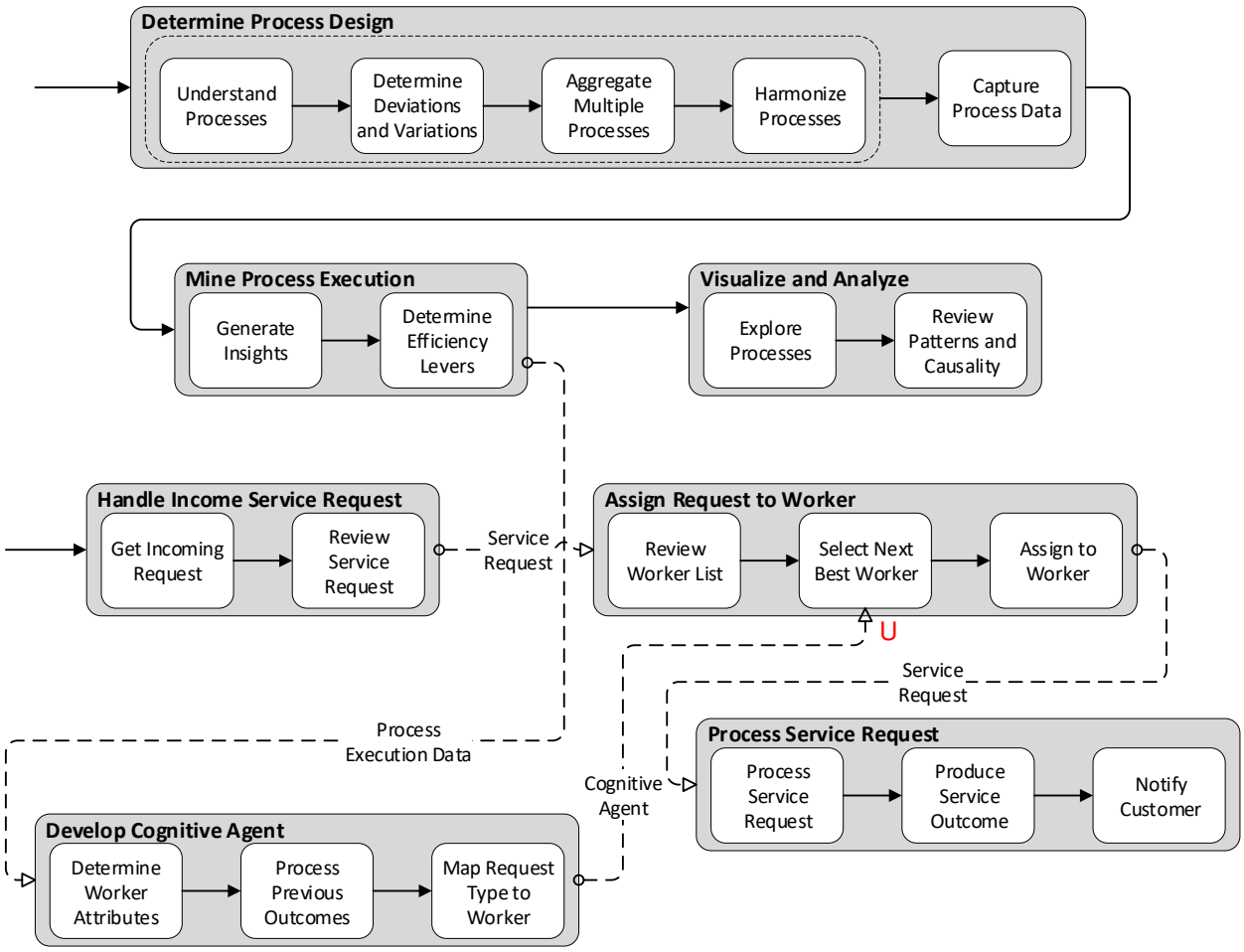

Fig. 8-15: hiBPM model for the learnability design pattern

8.7.2 Learnability through Mimicking Humans and Control

In the previous example, we explained how the cognitive agent can decide which knowledge worker would be best suited for a service request. There was an implicit assumption there that the historical process execution logs contain sufficient information to obtain meaningful information and be able to accurately rank knowledge workers against service request types. However, this may not be practically possible as often there is insufficient data available. For example, a knowledge worker may have recently joined the company and they may not have serviced many requests to correctly profile their execution ability by service request types (also commonly known as a cold start problem [204]). Alternatively, there may be tacit knowledge that is leveraged by a human manager to make service request assignments that is not typically captured in process execution logs. In such cases, the cognitive agent would not benefit from the historical process logging data and the service request assignments to knowledge workers may be misaligned with natural abilities of individual knowledge workers.

In such situations, we consider another hiBPM pattern where the cognitive agent makes an assignment decision by supplementing it with mimicking human decision-making. In cases where the cognitive agent cannot confidently make a service request assignment to a knowledge worker (as measured by recommender-based metrics), the agent relies on recent human manager assignment behaviour and mimics that to the degree possible. We show this in Fig. 8-16, where each instance of the process execution (including contextual data and process outcomes) is captured, reviewed and processed for learnability purposes by the **Capture Environment**, **Monitor Process Execution**, and **Assess Process Execution Outcome** process stages. The cognitive agent now has an ensemble of methods to make an assignment of service request utilizing historical process execution behavior or recent human manager decision making.

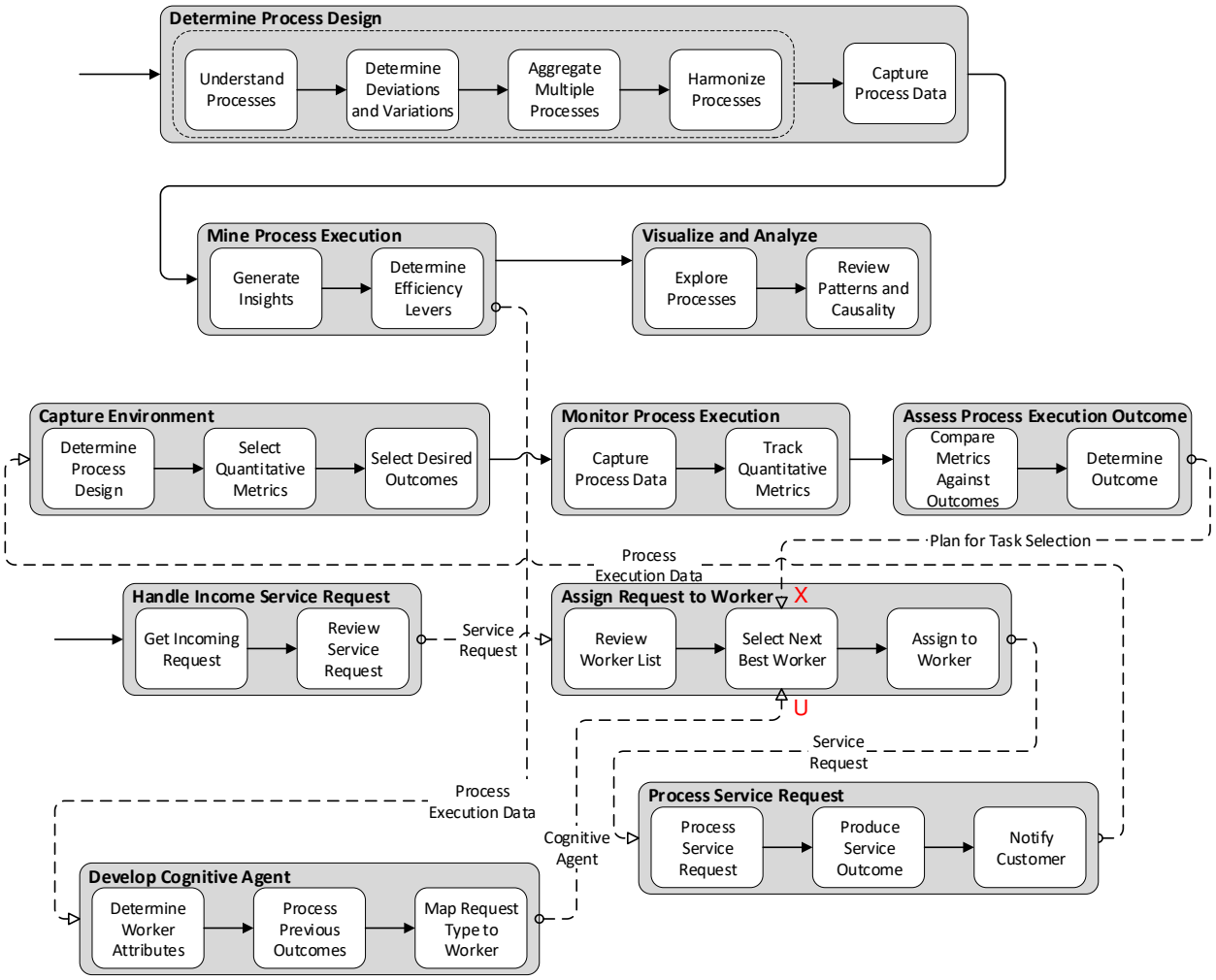

Fig. 8-16: hiBPM model for the learnability with human mimicking design pattern

8.7.3 Advisory User Engagement

The engagement between the human user and the cognitive agent is not static; shifts in work allocation between human users and cognitive agents can be envisioned. They are triggered by well-defined conditions that focus on system performance, user trust in systems' recommendations, and other vital characteristics of user engagement. It varies over time based on not just the evolving capabilities of enterprise systems but also on user requirements and enterprise context and objectives. Previous examples assumed that the assignment was entirely performed by a fully autonomous cognitive agent. However, there are a spectrum of possibilities here, ranging from advisory recommendations by the cognitive agent to fully autonomous where the agent

performs all the necessary process execution. In this hiBPM pattern, we consider the case where the cognitive agent provides an advice to the human knowledge worker who can then decide to approve or reject the provided recommendation.

Looking at the hiBPM model in Fig. 8-17, we see that **Select Next Best Worker** executes the transitions based on a plan prepared by **Select User Engagement**. The **Select User Engagement** process stage decides a chosen user engagement pattern by reviewing the possible set of user engagement patterns that can exist for that decision-making process element, instructing which form of user engagement to execute. A context change, in this case, may trigger a transition to a different user engagement pattern. E.g., as the collected data becomes more significant, the provided recommendation is deemed to be more accurate and the **Select Next Best Worker** can be shift from advisory to fully autonomous. However, if there are external changes which affect the quality of autonomous decision making by the cognitive agent, the **Select Next Best Worker** user engagement may shift from fully autonomous to a more conservative (less-automated) user engagement pattern until the model has been retrained; this is the advisory user engagement.

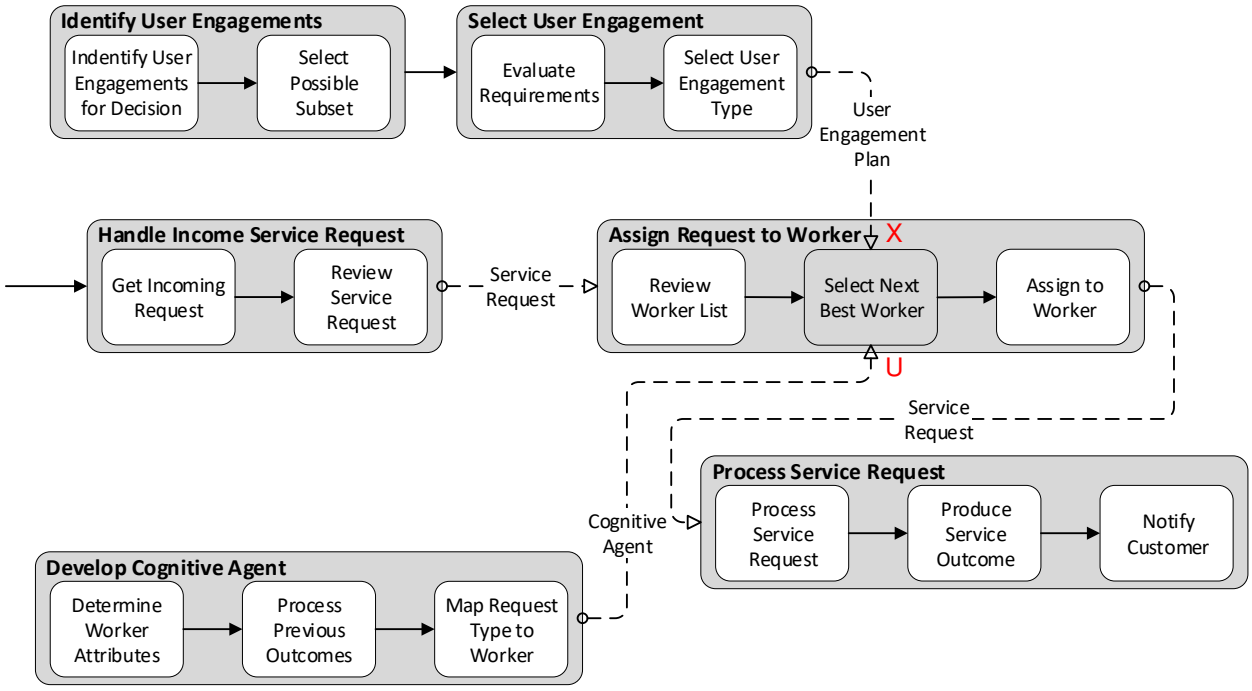

Fig. 8-17: hiBPM model for the advisory user engagement design pattern

8.7.4 Using Autonomous User Engagement with Human Governance

Users' attitude towards automation in general, and the system they are interacting with in particular, can change based on the accumulated history of the performance of the cognitive agent, including the quality of the output the agent produces. If we view this as a progression, then the cognitive agent outcome would be first treated as an advisory recommendation at one end, to a fully autonomous decision at the other end. Quality requirements and organizational domain constraints will affect the transitions as well, with organizations and individual decision-makers striving for hiBPM models that reflect their changing enterprise requirements, business domain constraints, and level of trust in cognitive agents they are employing.

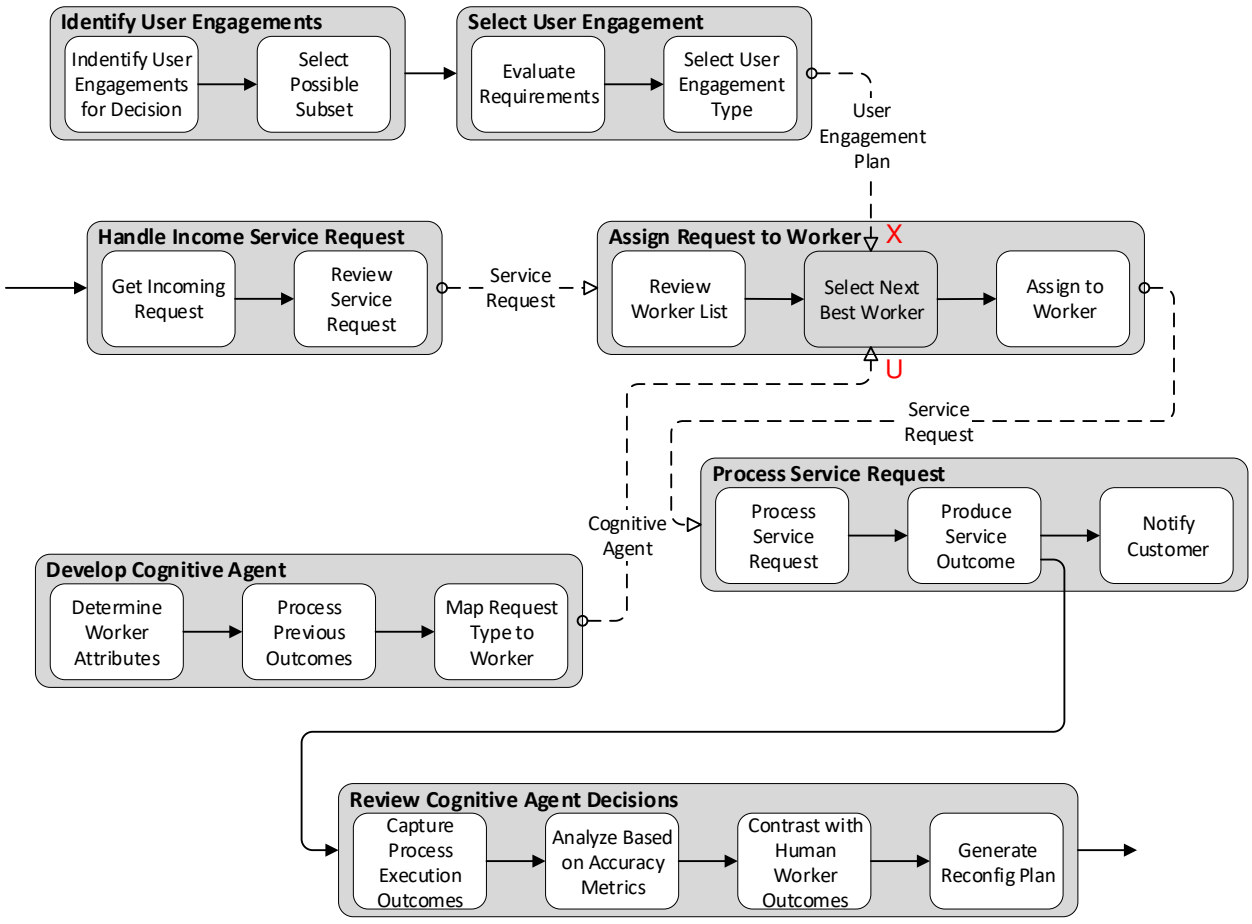

Fig. 8-18: hiBPM model for the human governance design pattern

In Fig. 8-18, we show another hiBPM model with the governance function added to the previous example. In cases where there is not complete trust in the quality of decision making, a governance function may be introduced where, at some defined interval and aggregate level, the outcomes of cognitive agent decision-making would be collectively reviewed and audited. Here, the nature and impact of this user engagement is not limited to the process containing the decisions that cognitive agents are assisting human decision-makers with, but also introduces new supporting processes that monitor and evaluate the cognitive-enhanced business processes to enable humans and cognitive agents to adapt to their changing capabilities.

8.8 Evaluation

The evaluation of the case study research was performed both during and after the conclusion of the case study. Periodic project group meetings were held with the stakeholders where the hiBPM models were presented that pertained to the domain segment that was the focus of that particular period. Feedback was solicited on the ability of the hiBPM models to visualize and analyze the domain segment under study, with the result of the evaluations guiding the next round of study and modeling. These meetings were scheduled and moderated by main contact point (the Secnior Software Engineer) from the company.

At the conclusion of the case study a qualitative evaluation of the performance of the overall hiBPM framework, the modeling notation, and the methods provided was done. This qualitative evaluation was done in the form of a concluding questionnaire (presented in the Appendix) that was filled out by the Senior Software Engineer from the company where the effectiveness of the hiBPM framework was evaluated against the measures of hiBPM's quality and ability to capture the domain and provide a means to understand and reason about possible solutions to the stated business problem. This individual was ideally suited for evaluating the effectiveness of the hiBPM models against the previously mentioned objectives based on their involvement in the reviewing of hiBPM models during the periodic meetings. The questionnaire attempted to find out the effectiveness of the hiBPM framework by asking specific questions on the ability of hiBPM models to characterize the domain, the usefulness of the hiBPM modeling notations in analysis,

and if hiBPM aided in determining at an optimum design faster compared to a situation if hiBPM had not been used.

8.8.1 Evaluation against Research Objectives

The results from the questionnaire against the research objectives (mentioned in Section 8.3) are given below,

Use Goal Model to Design hiBPM Model: The term "simplified" in Simplified CBO could mean differently from enterprise to enterprise. The NFR framework was used to determine key softgoals to consider during any CBO adoption exercise. These softgoals were then used to design the hiBPM model for the target organization while considering the satisficing of those softgoals. Goals and tasks were mapped to hiBPM constructs, such as process stages, process phases, and process elements. The questionnaire respondent confirmed that the structured methods provided as part of the hiBPM framework were able to produce a hiBPM model for the target enterprise considering their Simplified CBO objectives, including the supporting processes that needed to be introduced for those objectives.

Incorporate Context for Continuous Reconfiguration: The questionnaire participant confirmed that the hiBPM framework provided the mechanisms (in the form of the modeling notations and accompanying methods) to identify contextual variables at context sensing points in the hiBPM model that were necessary for introducing continuous redesigns in the hiBPM model. Further, the hiBPM models could emphasize the capturing and managing of context and relaying this context to variations points for redesigning sections of the hiBPM model for ensuring continuing satisfaction of enterprise business objectives for simplified CBO adoption.

Develop Pre-Built hiBPM Design Patterns: Multiple solution patterns could offer a systematic way of attaining different design objectives for the simplified CBO adoption. The questionnaire participant confirmed that the hiBPM framework was able to offer visualization of how design knowledge could be expressed in a reusable form that would address a wide range of business problems, particularly around reducing the complexity of CBO adoption. Further, the participant confirmed that the examples of hiBPM design patterns that were created for several commonly

occurring scenarios identified during the project discussion phase could be considered to be part of a design catalogue that the company could build over time to reduce the time and cost of CBO adoption.

8.8.2 Shortcoming of the hiBPM Framework

The responses in the questionnaire also identified several limitations to the hiBPM framework. It was not possible to model or analyze the full range of design possibilities for integrating supporting processes, including where the cognitive agent was to be used, due to a lack of support in the hiBPM framework for specific design characteristics of cognitive agents.

The hiBPM framework could neither entirely visualize nor provide methods to analyze where the context needed to be captured and analyzed and where reconfiguration of the hiBPM model should happen. Context could be of many types, and different contexts could change at different frequencies. Being able to differentiate between these scenarios was not possible in hiBPM models. When to trigger changes in the hiBPM model was also not supported. Finally, the mechanism of how the appropriate plan is selected, and how the redesign was performed, could not be explained by the present notations and constructs in hiBPM.

Finally, the hiBPM framework was not able to combine the different perspectives that would invariably be part of a single solution pattern. For example, a pattern that involved a design for the cognitive agent, a process architecture configuration, and a specific way of having this cognitive agent being engaged in the business process, could not be expressed. Showing relationships where process stages are responsible for creating “parts” (i.e., a design pattern) for a “whole” (i.e., a solution catalogue) is also not present.

8.8.3 Learnings from the Case Study

The hiBPM framework (along with the application of the NFR framework) proved valuable to understand what the requirements for Simplified CBO adoption would be for different enterprises. The hiBPM framework was applied to a range of situations and suitable models were produced that showed possible hiBPM model configurations. The answers to the questionnaire indicated that overall, the hiBPM framework was able to help (a) determine the (re)configuration of the process

architecture that was needed for simplifying adoption of cognitive agents in an existing business process, (b) uncover any additional processes that needed to be introduced to attain non-functional requirements for the enterprise, and (c) minimize the cost and duration of any implementation exercise by utilizing knowledge-based hiBPM model patterns present in pre-populated solution catalogues. Through the above, the cycle of discovery would be significantly reduced, as would be the requirement for highly skilled technical staff required for any project implementation.

As with the previous case study covered in Chapter 7, not all hiBPM modeling notations were required or used for actual model construction and analysis. The availability of different hiBPM constructs was appreciated by the participating company as different client environments would require using different modeling constructs. All of the modeling notations were found to be useful in the example that we used in the case study. However, the different ways of configuring the hiBPM model were not fully explored or utilized. For example, there are many ways of showing data flows or sequence flows between various structural elements. Not all of those were needed for our example in this case study. The same applies to other hiBPM modeling notations as well. Goal modeling was particularly useful in this case study as it allowed the construction and analysis of both brownfield and greenfield customer environments, i.e., environments with existing business processes and environments where business processes did not presently exist.

Finally, a limitation to the evaluation was the inability to validate the hiBPM framework in actual client settings, i.e., where the users of the IBM Business Automation Workflow software could create hiBPM models and analyze the different configurations based on their non-functional objectives.

8.9 Conclusion

Enterprises are increasingly taking advantage of cognitive computing to improve their business operations, resulting in greater operational efficiencies through better decision-making and ongoing cycles of learning and improvement. Organizational adoption of such advanced capabilities is difficult as user acceptance of advice and recommendations from an automated system requires the development of trust over time. Business processes and the processes responsible for user engagement with enterprise cognitive systems need not only to be designed,

but also have to change together with the supporting processes that can emerge and evolve over a period of time to monitor, evaluate, adjust, or modify the cognitive-enhanced business processes to enable employees to adapt to the enhanced capabilities of cognitive systems. As part of this case study, we discovered these hiBPM user engagement relationships that would signify the changing of relationships between the use of cognitive systems in these cognitive-enhanced business processes.

Any solution needs to be multi-dimensional and consider the design of the cognitive systems, their integration in business processes, and their usage by human users; while considering the changing nature of the business, human engagements, and systems complexity. At design-time, these are often unknown, so it is challenging to design the business process integration and simultaneously being able to achieve non-functional requirements of simplifying the cost and complexity of the overall solution. Thus, it is imperative not just to consider the design requirements of the cognitive systems for integrating into business processes, but also the context (and the possibility of changes to context) in which they are to operate.

In this chapter, we present a case study where the hiBPM framework was applied to the design of cognitive-enhanced business processes. Specifically, we considered designing and configuring hiBPM models when non-functional requirements for simplifying the adoption of cognitive agents in enterprises needed to be considered. The NFR framework was used to determine and prioritize the softgoals to consider during such an adoption. In order to achieve certain softgoals, such as adaptability and learnability, we identified the external context that was to be monitored and to be then used to determine alternative hiBPM model configurations at key variation points. We used solution catalogues to identify design patterns that could be used to propose hiBPM model configurations for reducing the adoption barrier. Through the application of the hiBPM framework in this case study, we found some interesting patterns on how hiBPM models could be configured.

9 Conclusions

9.1 Summary

The objective of this PhD research project was to characterize the fundamental types of software-enabled enterprise transformations and to provide conceptual modeling techniques for analyzing and designing process architectural reconfigurations to analyze these transformations. An essential assumption in this thesis was that having reconfigurable and flexible business processes enables these software-enabled enterprises to respond to changing situations by selecting suitable process architectural alternatives that help best meet enterprise business goals. We now review the stated research objectives in Chapter 1 against the research work performed as part of this thesis project.

Our first research objective was "*to identify a set of characteristics of enterprises undergoing change due to software technology innovations and determine a set of requirements for a conceptual modeling framework that can model and analyze the reconfigurations of enterprise process architectures*". In Chapter 3, we performed a systematic literature review to identify common traits present in enterprises that are incorporating emerging digital technologies and software to foster innovation and change. By performing a systematic literature review, we discovered commonalities across multiple organizations and industry segments and extracted them as a set of characteristics of such enterprises. These characteristics were then abstracted to uncover a set of requirements for an conceptual modeling framework that would allow modeling and analyzing these characteristics. The requirements were deduced by studying each characteristic with regard to its implications to the enterprise from a process perspective.

Our second research objective was "*to use these requirements to design a conceptual modeling framework that identifies the upstream factors (i.e., the "whys") that should be considered in the design of business and software processes that can be traced to enterprise business objectives*". In Chapter 5, we introduced the hiBPM framework that can visualize multiple business and software processes and their relationships as part of an overall process architecture, including various possible configurations that can occur. In Chapter 6, we provided accompanying methods and techniques to guide the selection of alternative process architecture configurations that help meet

certain enterprise objectives through modeling and analyzing changes in the hiBPM model. This thesis extends the preliminary work done in [26] and [27] by developing the hiBPM framework and adding new constructs, details and preciseness to the previously introduced notations.

Our third research objective was "*to consider the downstream effect (i.e., the "hows") of software systems design and software usage during the design of business processes, including acknowledging the interplays between software design and the design of business processes that use these software*". In Chapter 5 and Chapter 6, we emphasized the existence of different types of hiBPM process constructs for modeling the interactions between processes responsible for developing software tools and artifacts, and the processes where these software are used. Process stages offered insight into where software artifacts and tools are built versus where they are used, and the existence of planning process stages that influences the execution of the process stages responsible for either building or using the software artifacts. These were captured as design-use and plan-execute relationships in a hiBPM model. Further, through sense-and-control pathways, areas of the enterprise responsible for collecting data and using it for adaptation and transformation were identified.

9.2 Contributions

In Chapter 3, we presented several requirements for a conceptual modeling framework that would help understand and analyze the type of transformations that could be undertaken in software-enabled enterprises. We revisit those requirements individually below;

R1 - Relationship Amongst Processes: hiBPM models were able to show the relationships between multiple business and software processes that result in a process architecture by representing aggregations of activities as process stages and using relational linkages to show the associations between them. The resulting process architecture differed from other, prior, notions of business process architecture, for example, as presented in [50] and [51]. In the process architecture presented in [50], only three types of relationships were highlighted, sequences, decomposition and specialization, whereas additional ways of considering the relationship between processes by considering notions such as composition, specialization, trigger, and information flow were defined in [51]. We went beyond these prior contributions by introduction

relationships such as plan-execute, design-use, and sense-and-control that allow additional dimensions of changes to be analyzed in the process architecture. These relationships helped analyze ongoing changes in the enterprise through hiBPM models and use process architectures to model those changes and analyze possible variants of process architecture configurations that can exist.

R2 - Multiple Types and Levels of Processes: As part of our motivating arguments in Chapter 1, we mentioned that it is no longer sufficient to study and optimize individual and isolated business processes to accommodate change in enterprises. Instead, when considering the process architecture, we need to ignore the traditional boundaries that demarcate the different types of processes, such as business or software, and find additional possibilities for redesigning that goes beyond what the analysis of a single process could offer. There would be different levels of processes, with some processes being responsible for strategic planning that influence the behaviour of operational processes, whereas other processes would be building software artifacts that are used elsewhere in the process architecture. hiBPM was able to capture these different process types and levels using the relational elements of plan-execute, design-use, and sense-and-control. Process boundaries are used to highlight the segregation between different areas in the hiBPM model. The approach is different from that of other enterprise modeling approaches. For example in ArchiMate [44], the enterprise layers map to different functional areas, such as business, application, and infrastructure, whereas the hiBPM framework highlights the process types and levels based on their contribution to other parts of the hiBPM model.

R3 - Enterprise and Process Goals: As part of the hiBPM framework, we used goal models to help determine the structure of the process architecture by constructing and navigating the goal graphs and seeing how a goal structure can be applied to an appropriate configuration of the process architecture model. Using goal models for analyzing and comparing alternative options is well established in scholarly literature. There is already significant work done on associating goal models and process models [93][188][189][205]. Our work differs from these as we consider associating process stages with goals or softgoals, as these process stages indicate some accomplishment of enterprise functional or non-functional objective. The goal models are further

able to provide a relational structure to the hiBPM model, while also helping populate the internals of these process stages.

R4 - Trade-Off Analysis: Enterprise are expected to function in uncertain environments. Thus, these enterprises need to have ingrained in their process architecture a certain degree of flexibility that permits the enterprise to reconfigure its process architecture as needed. However, maintaining this flexibility comes at a cost; this cost can be analyzed through trade-off analysis using goal models. Further there are many ways of designing the hiBPM model, with each design configuration still attaining the enterprise or process functional goal but other non-functional goals may not be satisfied. The goal models were additionally used for deciding between hiBPM model alternatives by analyzing possible configurations of the process architecture that satisfy both functional and non-functional goals. This was done by identifying variation points in the hiBPM model where there could exist alternative process architecture configurations. Reconfigurations to attain functional goals can be shown as multiple choices using OR decompositions in the goal model. We used existing techniques for creating goal models and performing goal satisfaction analysis [172]. However, we associated the goal models (for performing trade-off analysis) at variation points in the hiBPM model; there were locations where the hiBPM model was open to reconfiguration.

R5 – Abstract Software Artifact Design: Through the use of design-use and plan-execute relationships, hiBPM models could indicate where software tools and artifacts were being built and where they were being used, including the planning that needed to be done for the building of and using of these software tools and artifacts; these were presented as designs in the hiBPM model, with the design completeness providing guidance to supporting evolvability requirements for those software artifacts. Flexibility in process execution was provided through the creation of plans that would guide the execution. These abstract designs permitted conceptual and visual analysis of their contribution to, and participation in, the overall enterprise business and software processes, particularly those which need to be altered to introduce change. Having such abstract software artifacts as part of the design of the hiBPM model allows the capturing of processes that are responsible for their building and the processes where these artifacts are used, without necessarily having to detail the complete requirements and structure of the artifacts being

developed. This differs from other artifact-centric process modeling approaches such as those proposed in [80].

R6 – Pushing Design Decisions Downstream: Through design-use relationships, the hiBPM framework was able to show the existence of two levels of processes, one where the process is responsible for the creation of the software artifact while the other is where the process is responsible for (repeatedly) using the designed artifact. Further, there were many possible ways of designing this software artifact, ranging from full designing where all design decisions have been made for that artifact to minimum designing where a lot of choices regarding artifact use are left at use-time. Such a dimension of change allowed considering situations where moving an activity in the other direction reduces the level of automation available to the use process stage, while moving a decision increases the level of customizability of the tool since the decision is no longer built into the tool and can be changed during its use. We consider designs to be black-box artifacts in hiBPM, tools that are produced in one part of the hiBPM model, and to be used at another part. This notion of designs differs from approaches proposed by other researchers where designs are considered to be absolute or completely defined [181] or where designs are considered along with the environment in which they are to be used in [182].

R7 – Upfront Planning vs. Deferred Planning: Through hiBPM models, we were able to show planning relationships between different process stages, with various plans being prepared based on their execution by downstream processes. Some plan-related activities were left for later because of the unavailability of data required for processing the plan. Through plan-execute relationships, the hiBPM model was able to illustrate a degree of planning, ranging from full planning where each downstream activity is entirely planned, to minimum planning where a lot of decisions regarding process execution are left at execution time. The motive for having plan-execute relationships in hiBPM is to maintain a degree of enterprise flexibility by analyzing how much to pre-plan in the planning stage and how much to leave to the execution stage. Plan-execute relationships also help identify different parts of the hiBPM model, such as those responsible for planning, and those responsible for executing, irrespective of the areas of the organization those processes are placed in. The plan-execute relationship in hiBPM provides reasoning about how the

plan came about and how much to pre-plan. This differs from the ArchiMate servicing relationship as that is a simple service relationship with one layer providing a service to another layer.

R8 – Feedback and Feedforward Paths: In hiBPM, paths of change were identified and analyzed as sense-and-response feedback paths through which the enterprise adapts and improves. Process stages would be designated as either sensing or responding to the sensed data, based on sense-and-control relationships. Sensing and responding take place in business and technology processes that exist at different levels of dynamics and timescales. By differentiating control and design inputs from regular data inputs, and sensing from regular outputs, the hiBPM framework was able to locate adaptive loops as they exist within a model. hiBPM was also able to highlight the various relative timescales (based on the different process levels, and recurrence relationships between process stages) that such a loop would traverse. The inclusion of feedback paths in hiBPM models differs from other, somewhat similar, approaches of adaptive process design, such as that presented in [90] where the feedback paths are used to assign resources and take corrective actions in the execution of processes, rather than redesigning them at an architectural level. In hiBPM, we are mostly interested in emphasizing the information flow back into a higher-order process stage rather than the iterative executions of the same activities. Such information flows can then assist with determining appropriate selection of hiBPM process configurations.

R9 – Represent and Reason about Speed, Timescales and Process Cycles: Once a hiBPM model had been created, it was possible to reason about the design of the process architecture, particularly around the speed of execution of various process stages that produce an enterprise deliverable. Through adjustments and modifications to this hiBPM model, it was possible to reconfigure the model for improved recurrence of the process stages or faster delivery of the enterprise product of service. These design modifications can be done by making changes to the hiBPM structural elements, by making changes to the hiBPM relational element, or a combination of both. As detailed in Chapter 5 and Chapter 6, hiBPM offers many different ways of defining the process architecture along various dimensions of change, which help attain diverse enterprise requirements around the speed and frequency of product or service delivery. By determining and maintaining variation points in the process architecture, it is possible to incorporate the design

modifications as needed once the necessary information is available to aid in decision making and trade-off analysis.

9.3 Limitations

As part of the hiBPM framework, we provided a modeling notation, and accompanying methods, to aid enterprise architects and process architects in designing and analyzing process architectural configurations. However, there exist both limitations to the hiBPM framework and threats to the validity of the research performed. We discuss these below,

Limitations in the Systematic Literature Review: In Chapter 3, we provided the systematic literature review process employed for independent reproduction by other researchers. This review relied on qualitative reasoning and analysis of articles and it is conceivable that other reviewers executing a similar review process may see slightly different results or uncover additional characteristics. Further, a reviewer may come up with a different set of characteristics if other industry trends are selected at the initiation of the literature review. This list of characteristics was not meant to be exhaustive or absolute, as the identification process may be prone to observer bias. Our intention was not to provide a precise definition of software-enabled enterprises but to discover characteristics for developing an understanding of the key challenges in modeling enterprises that are underdoing transformation due to emerging digital technologies.

Limitations in the Design of the hiBPM Framework: Our research has been grounded in the positivist epistemological theory, and that is reflected in the selection of design science research as the methodology used for developing our hiBPM framework. As a result of this approach, we explain the causal relationships and structuring of order in the organizations through a sequence of activities for theorizing, justifying, building and evaluating the hiBPM framework. This is opposed to other research methodologies, like action research, which promotes a more collaborative approach towards the development and refinement of a theory where the researcher and practitioner iteratively and cooperatively work towards solving research problems (that can include the development of artifacts) in an organizational context. Thus, the use of other methods in this research project could have uncovered additional research findings.

Limitations in the Case Study Evaluation: There were some limitations with regards to the research projects with industry partners, particularly around the nature and availability of research data and team members. Both research projects were time bounded and a predetermined problem was presented that required understanding and analysis. Relevant stakeholders were made part of the research team and were available to have team discussions and analyze alternative design choices, including reviewing suitable alternatives that would solve the identified problem. In some case studies, additional data was provided in the form of written documents and architectural diagrams to supplement the team discussions. The defined structure of the research projects both provided and limited the context of the case study and influenced the research activities performed. Not all hiBPM modeling constructs were equally utilized in analyzing the problem domain across both case studies.

Limitations in the Usage of hiBPM: hiBPM does not provide complete coverage for modeling and designing the processes and software for enabling enterprise transformations but rather is an initial step towards this direction. In hiBPM, we focus on the problem space rather than the solution space. We try to understand the situation that an organization finds itself in and propose alternative ways of solving that problem. We abstract away from implementation details by not providing the implementation of the solution. This is left to other modeling approaches (conceptual or otherwise) to continue the design activities that lead towards solution implementation. Moreover, when studying a domain, hiBPM models only capture sufficient detail that is necessary for analyzing alternative ways of configuring the space, as the expanse of detail is difficult to assess and may require multiple rounds of iterations to determine what details to include, and what to omit out. This is a subjective exercise and different modelers may approach this exercise differently and produce different hiBPM models for the same domain being studied.

9.4 Significance

This work contributes to several research areas, including enterprise architecture, business process design, and software process design. We discuss each area below,

Enterprise Architecture: Existing enterprise architecture frameworks and approaches, such as ArchiMate [44] and TOGAF [45] allow for designing enterprises by considering multiple enterprise layers and functions. However, they do not cater to periodic and variable changes including the ability to decide between multiple alternative enterprise configurations and dealing with design uncertainty. Through the hiBPM framework, we consider the bidirectional adaptation influences between business and technology, particularly those enabled by software systems and business processes. hiBPM provides the necessary notations and methods to allow enterprise architects to study and analyze granular and ongoing shifts in process architecture over shorter periods, particularly for enterprises in uncertain environments. hiBPM models allow modeling of fundamental transformation requirements of enterprises from multiple perspectives (process, goal, and artifact) while reasoning about interplays and influences amongst (as well as within) these perspectives. The plan-execute and design-use relationships in hiBPM permit enterprise architects to reason about how the plan or the design came about (thus allowing contemplation of alternative configurations) as opposed to being a simple service relationship between two adjacent layers in enterprise architecture.

Business Process Design: Present approaches to modeling and analyzing business process architectures are used to provide an abstract representation of multiple processes that exist in an enterprise [50][51]. These also provide a means for developing a more holistic view of the organization by associating business process modeling and enterprise architecture while additionally decomposing processes into a higher level of granularity that provides increased visibility on the constituent parts of the integrated processes. By introducing the hiBPM framework, we contribute to these existing approaches by focusing on the need for ongoing change in the enterprise and the ability to accommodate uncertainty in the design of process architectures. Through hiBPM models, we can analyze possible variants of process architecture configurations that exist through emphasizing alternative designs that exist at variation points in the hiBPM model and various means of reconfiguring the hiBPM model for enabling fundamental transformation of the enterprise. Through the design and plan artifacts, we can identify processes that produce software tools and artifacts, and processes that aid other processes in the production of these tools.

This way, we can determine an ecosystem of processes (including their relationships) that exist, and collectively analyze them, without having to focus on individual processes at a time.

Software Processes Design: There have been significant innovations in the design of software process to better support software development and production activities, including process automation, by focusing on activities, software artifacts to be produced or used, and the participants in these processes [80][86][87][88]. However, changing business models or strategic direction requires successfully introducing software process reconfigurations to influence the continuous delivery of value, product and services. Thus, it is no longer sufficient to consider software processes separately from business processes. Through hiBPM models, we can understand these possible forms of software process reconfigurations to identify critical points of process variations and the influencing factors that contribute towards these reconfigurations. We can also link the impact of software process reconfigurations to business goals and requirements in order to exploit synergies and mitigate negative consequences in software production. We can show how the capturing and usage of various software metrics can be used for ongoing software process design improvement, mainly through studying the hierarchical relationship between the processes in all stages of the feedback loop, i.e., sensing, interpreting, deciding and acting.

9.5 Future Directions

As part of this thesis research, several areas were identified that we plan on expanding on and incorporating into the hiBPM framework as part of future work. Additionally, some of the limitations in the hiBPM framework mentioned in the previous section can also be considered. We discuss these in further detail below,

Goal-based and Actor-based Analysis of hiBPM models: Presently we offer limited guidelines on how hiBPM models and goal models can be associated with reconciliation and traceability purposes. However, additional work remains to be done for better associating both these sides, including providing methods for traceability of analysis. We can further leverage other goal modeling techniques and introduce actor modeling to supplement the analysis that can be performed in hiBPM. This may result in the introduction of additional constructs, notations and

methods to hiBPM models. Further, there may be extensions proposed to existing goal modeling and actor modeling techniques.

Use of Software in Processes: Through the use of hiBPM designs and plans, we show the types of software processes that can be designed, including the software artifacts that they need to build. As mentioned previously, our aim was primarily to reason about the design of software processes, and the artifacts they produce, through better evaluation of the problem space. The design of these produced software artifacts is considered at an abstract level (as explained using design-use relationships) without discussing the details on how these software systems would be implemented. We wish to extend hiBPM to be able to better contribute towards the formulation of the requirements of these software systems.

Design Patterns and Solution Catalogues: We introduced the hiBPM solution catalogues in this thesis and provided some preliminary design patterns for common business problems that we encountered during our case study research. We aim to further populate our solution catalogues with new design patterns as we discover them. The expectation is that these catalogues would contain commonly accepted patterns for solving problems from an enterprise configuration perspective. The solution catalogues would not just include hiBPM models but would also show how the software systems design could be determined and how it would satisfy enterprise goals and actor goals.

Data and Context Integration: Monitoring and analyzing the external environment is necessary for enabling agility in enterprises, with flexible processes and adaptable software systems being designed accordingly. This requires identifying and capturing the external data and monitoring for its availability. This data can then be utilized at variation points to determine the appropriate hiBPM variant after suitably assessing the trade-offs involved. We will further expand on how the external context is monitored and the necessary plans determined for reconfiguring of portions of the hiBPM model, including the degree of planning and designing that needs to be done. Another area where we envision additional research being required is the modeling and use of internal context and its effect on the requirements of system design and process reconfiguration.

Moving to Automated Processes: The introduction of intelligent systems in enterprises usually results in increased automation, resulting in changes in responsibility assignments among humans and automated systems. The modes of engagement between users and systems were covered preliminarily in this thesis; however, we plan to study this change dimension in more detail. We aim to identify further the requirements for process automation, including the expressiveness needed to cover an initial set of cognitive services, business patterns, recommendation types, and user engagement modes, while identifying meta-level processes that would help with learning and ongoing improvement for both enterprises and systems.

Formalization: The hiBPM framework relies on model visualization to capture and analyze a domain, with textual explanations provided on how the modeling constructs are to be used. Therefore, users of the hiBPM framework can quickly adopt the modeling approach without needing to spend significant effort in learning the notation and its usage. However, as the hiBPM modeling user base grows and the framework is used more widely, a more precise definition may be needed to ensure that the notation is used correctly. For this, algebraic definitions may be needed for key constructs in the hiBPM framework. Such formalizations would also help with traceability and transformation between different modeling languages (e.g., hiBPM to goal models).

Improvement to hiBPM Notation: There are several enhancements that can be made to the hiBPM modeling notations. Additional sequencing information can be made part of the hiBPM model to improve readability and help differentiate between multiple levels. Notations and rules for collapsing processes can be added to simplify the model and hide portions that are irrelevant for analysis. Improved documentation and a user guide can accompany the hiBPM framework that would allow for practical adoption by a wider user base. Symbols can be introduced that show which elements in the hiBPM model are associated with corresponding elements in a goal model. Additionally, variation points can be explicitly highlighted in the hiBPM model.

Appendix - Questionnaire

[Q1] At the end of modeling sessions, were the modelers (i.e., Zia and Eric) able to arrive at a characterization of your existing analytics solution/product?

- If your answer is NO, please explain what aspects/parts/components of your product/solution were not identified at the end of modeling sessions.
- If your answer is YES, please give 2-3 sentences on which area of the graphical models correspond to which part of your product.

[Q2] Through the course of this collaboration, were there any instances of understandings, findings, conclusions that you and your team were not able to arrive at that prior to the modeling activities?

Please provide 2-3 examples.

[Q3] What did you find useful about the framework? (Write 3-4 sentences or bullet points) (This can include specific modeling language features or methodological steps, as well as the general approach.)

[Q4] What do you think is most lacking in the framework? Are there additions to or variations on the framework that you would like to see?

[Q5] Provide 2-3 examples of features that are not part of current your product/solution, but after the modeling sessions, you think that they can be fruitful additions.

[Q6] What are the aspects or features of the framework that you consider least useful? (This can include modeling language features as well as methodological steps.)

[Q7] In arriving at your current analytics solution/product, you had evolved the product conception and design through one or more iterations in the past. Retrospectively, do you think using the modeling framework would have enabled you to arrive at a viable product more easily or sooner? E.g., in uncovering pain points and analyzing failure stories and scenarios, and in providing guidance and focus in the search for solutions.

References

1. Slaughter, S. A., Levine, L., Ramesh, B., Pries-Heje, J., Baskerville, R.: Aligning software processes with strategy. MIS Quarterly, pp. 891-918 (2006)
2. Wilkinson, M.: Designing an “adaptive” enterprise architecture. BT Technology Journal, 24(4), pp. 81-92 (2006)
3. Westerman, G., Bonnet, D., McAfee, A.: The Nine Elements of Digital transformation. MIT Sloan Management Review (2014)
4. Fuggetta, A.: Software process: A roadmap. In Proceedings of the Conference on the Future of Software Engineering, pp. 25-34, ACM (2000)
5. Dietz, J.: Enterprise Ontology: Theory and Methodology. Springer, Berlin-Heidelberg (2006)
6. Dijkman, R., Vanderfeesten, I., Reijers, H.: Business process architectures: Overview, Comparison and Framework. Enterprise Information Systems, 10(2), pp. 129-158 (2016)
7. 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)
8. Ridley, D.: The Literature Review: A Step-by-Step Guide for Students. Sage Publications Ltd (2008)
9. Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. The Journal of Systems & Software, 80(4), pp. 571-583 (2007)
10. Petticrew, M., Roberts, H.: Systematic reviews in the social sciences: A practical guide. Blackwell Publishing (2006)
11. Okoli, C., Schabram, K.: A Guide to Conducting a Systematic Literature Review of Information Systems Research. Sprouts: Working Papers on Information Systems, 10(26) (2010)
12. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research, MIS Quarterly, 28(1), pp. 75-105 (2004)
13. Papas, N., O'Keefe, R. M., Seltsikas, P.: The action research vs design science debate: Reflections from an intervention in eGovernment. European Journal of Information Systems, 21(2), pp. 147-159 (2012)
14. Valverde, R., Toleman, M., Cater-Steel, A.: Design science: A case study in Information Systems re-engineering. Strategic Information Systems: Concepts, Methodologies, Tools, and Applications. Information Science Reference (IGI Global), pp. 490-503, Hershey PA, USA (2010)
15. Dubé, L., & Paré, G.: Rigor in Information Systems positivist case research: Current practices, trends, and recommendations. MIS Quarterly, 27(4), pp. 597-636 (2003)
16. Iivari, J., & Venable, J.: Action research and design science research–seemingly similar but decisively dissimilar. In European Conference on Information Systems, Vol. 17, pp. 1-13 (2009)
17. Peffers, K., Rothenberger, M., Tuunanen, T., & Vaezi, R.: Design science research evaluation. In International Conference on Design Science Research in Information Systems, pp. 398-410, Springer Berlin Heidelberg (2012)

18. Yin, R. K.: Case study research: design and methods (4th ed.). Sage Publications, Los Angeles, California (2009)

19. Kitchenham, B., Pickard, L., & Pfleeger, S. L.: Case studies for method and tool evaluation. IEEE Software, 12(4), 52-62 (1995)

20. Cavaye, A. L. M.: Case Study Research: A Multi-faceted Research Approach for IS. Information Systems Journal, Vol. 6, pp. 227-242 (1996)

21. Lee, A. S.: A Scientific Methodology for MIS Case Studies. MIS Quarterly, Vol. 13, pp. 33-50 (1989)

22. Shanks, G.: Guidelines for Conducting Positivist Case Study Research in Information Systems. Australasian Journal of Information Systems, 10(1) (2002)

23. Easterbrook, S., Singer, J., Storey, M. A., Damian, D.: Selecting empirical methods for software engineering research. In Guide to Advanced Empirical Software Engineering, pp. 285-311, Springer London (2008)

24. Miles, M. B., Huberman, A. M., Huberman, M. A., Huberman, M.: Qualitative data analysis: An expanded sourcebook (1994)

25. Dubé, L., Paré, G.: Rigor in information systems positivist case research: Current practices, trends, and recommendations. MIS Quarterly, pp. 597-636 (2003)

26. Lapouchnian, A., Yu, E., Sturm, A.: Re-designing process architectures towards a framework of design dimensions. In 2015 IEEE 9th International Conference on Research Challenges in Information Science, pp. 205-210 (2015)

27. Lapouchnian, A., Yu, E., Sturm, A.: Design dimensions for business process architecture. In International Conference on Conceptual Modeling, pp. 276-284, Springer Cham (2015)

28. Olanrewaju, T.: The rise of the digital bank, McKinsey on Business Technology, McKinsey & Company, Washington DC, Number 33, (2014)

29. Denecker, O., Gulati, S., Niederkorn, M.: The Digital Battle That Banks Must Win. McKinsey & Company (2014)

30. Burke, W. Warner: Organization change: Theory and practice. Sage Publications (2013)

31. Aritomo, K., Desmet, D., Holley, A.: More Bank for your IT buck, McKinsey & Company (2014)

32. The Economist: Organisational Agility: How Business can Survive and Thrive in Turbulent Times. A report from The Economist Intelligence Unit (2009)

33. 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)

34. Lwakatare, L. E., Kuvaja, P., Oivo, M.: Dimensions of DevOps. In International Conference on Agile Software Development, pp. 212–217. Springer International Publishing (2015)

35. Smeds, J., Nybom, K., Porres, I.: DevOps: A Definition and Perceived Adoption Impediments. In International Conference on Agile Software Development, pp. 166 –177, Springer International Publishing (2015)

36. Wasserman, A.: Software engineering issues for mobile application development. In Proceedings of the FSE/SDP workshop on Future of software engineering research, pp. 397-400, ACM (2010)

37. Haeckel, S.: Adaptive enterprise: Creating and leading sense-and-respond organizations. Harvard Business Press (2013)

38. Cummins, F. A.: Building the agile enterprise: with SOA, BPM and MBM. Morgan Kaufmann (2010)

39. Hoogervorst, J.: Enterprise Architecture: Enabling Integration, Agility and Change. International Journal of Cooperative Information Systems, Vol 13, pp. 213–233 (2004)

40. Opdahl, A. L., Berio, G., Harzallah, M., Matulevicius, R.: An ontology for enterprise and information systems modeling. Applied Ontology, Vol 7(1), pp. 49–92 (2012)

41. Anaya, V., Berio, G., Harzallah, M., Heymans, P., Matulevičius, R., Opdahl, A. L., Verdecho, M. J.: The unified enterprise modeling language—overview and further work. Computers in Industry, Vol 61(2), pp. 99-111 (2010)

42. Fuggetta, A., Di Nitto, E.: Software process. In Proceedings of the on Future of Software Engineering, pp. 1-12, ACM (2014)

43. Krafzig, D., Banke, K., Slama, D: Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall Professional (2005)

44. ArchiMate® 3.0 Specification. Retrieved from http://pubs.opengroup.org/architecture/archimate3-doc/

45. TOGAF® Version 9.1, Retrieved from http://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html

46. Svahnberg, M., Van Gurp, J., Bosch, J.: A taxonomy of variability realization techniques. Software: Practice and Experience, Vol. 35(8), pp. 705-754 (2005)

47. Van Gurp, J., Bosch, J., Svahnberg, M.: On the notion of variability in software product lines. In Proceedings of IEEE/IFIP Working Conference on Software Architecture, pp. 45-54, IEEE (2001)

48. Souza, V. E. S., Mylopoulos, J.: Requirements-based software system adaptation. PhD Thesis, University of Trento, Italy (2012)

49. Doyle, J. C., Francis, B. A., Tannenbaum, A. R.: Feedback control theory. Courier Corporation (2013)

50. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management, Chapter 2. Springer-Verlag Berlin Heidelberg (2013)

51. Eid-Sabbagh, R., Dijkman, R., Weske, M.: Business process architecture: use and correctness. In Proceedings of 10th International Conference on Business Process Management (BPM'12), pp. 65–81, Springer-Verlag Berlin Heidelberg (2012)

52. Rosemann, M., vom Brocke, J.: The six core elements of Business Process Management. In Handbook on Business Process Management 1, pp. 105-122, Springer Berlin Heidelberg (2015)

53. Business Process Model and Notation v2.0, Retrieved from http://www.omg.org/spec/BPMN/2.0/PDF/

54. De Giacomo, G., Dumas, M., Maggi, F. M., & Montali, M.: Declarative process modeling in BPMN. In International Conference on Advanced Information Systems Engineering, pp. 84-100, Springer Cham (2015)

55. Bhattacharya, K., Gerede, C., Hull, R., Liu, R., & Su, J.: Towards formal analysis of artifactcentric business process models. In International Conference on Business Process Management, pp. 288-304, Springer Berlin Heidelberg (2007)

56. Hull, R.: Artifact-centric Business Process Models: Brief survey of research results and challenges. In OTM Confederated International Conferences, On the Move to Meaningful Internet Systems, pp. 1152-1163, Springer Berlin Heidelberg (2008)

57. Bhattacharya, K., Caswell, N. S., Kumaran, S., Nigam, A., Wu, F. Y.: Artifact-centered operational modeling: Lessons from customer engagements, IBM Systems Journal, Vol 46(4), pp. 703-721 (2007)

58. Gerede, C. E., Bhattacharya, K., & Su, J.: Static analysis of business artifact-centric operational models. In IEEE International Conference on Service-Oriented Computing and Applications (SOCA'07), pp. 133-140, IEEE (2007)

59. Liu, R., Bhattacharya, K., Wu, F. Y.: Modeling business contexture and behaviour using business artifacts. In International Conference on Advanced Information Systems Engineering, pp. 324-339, Springer Berlin Heidelberg (2007)

60. Nigam, Anil, Caswell, N. S.,: Business artifacts: An approach to operational specification. IBM Systems Journal, Vol. 42(3), pp. 428-445 (2003)

61. Kumaran, S., Liu, R., Wu, F. Y.: On the duality of information-centric and activity-centric models of business processes. In International Conference on Advanced Information Systems Engineering, pp. 32-47, Springer Berlin Heidelberg (2008)

62. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In Handbook of Research on Business Process Modeling, pp. 503-531, IGI Global (2009)

63. Kapoor, S., Bhattacharya, K., Buckley, S., Chowdhary, P., Ettl, M., Katircioglu, K., Phillips, L.: A technical framework for sense-and-respond business management. IBM Systems Journal, 44(1), pp. 5-24 (2005)

64. Hull, R., Nezhad, H.: Rethinking BPM in a cognitive world: Transforming how we learn and perform business processes. In International Conference on Business Process Management (2016)

65. Reijers, H. A., Mansar, S. L.: Best practices in Business Process Redesign: An overview and qualitative evaluation of successful redesign heuristics. Omega, Vol 33(4), pp. 283-306 (2005)

66. La Rosa, M., Aalst, W.M.P. van der, Dumas, M., Milani, F.P.: Business process variability modeling: A survey. ACM Computing Surveys (2013)

67. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer-Verlag Berlin (2012)

68. Weber, B., Reichert, M., Rinderle-Ma, S.: Change Patterns and Change Support Features – Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Engineering, Vol (3), pp. 438–466 (2008)

69. Simidchieva, B. I., Clarke, L. A., Osterweil, L. J.: Representing process variation with a process family. In International Conference on Software Process, pp. 109-120, Springer Berlin Heidelberg (2007)

70. Santos, E., Pimentel, J., Castro, J., Finkelstein, A.: On the dynamic configuration of business process models. In Enterprise, Business Process and Information Systems Modeling, pp. 331-346, Springer Berlin Heidelberg (2012)

71. Ayora, C., Torres, V., Weber, B., Reichert, M., Pelechano, V.: VIVACE: A framework for the systematic evaluation of variability support in Process-Aware Information Systems. Information and Software Technology, Vol 57, pp. 248-276 (2015)

72. Henricksen, K., Indulska J.: A Software Engineering Framework for Context-Aware Pervasive Computing (2004)

73. Ceri, S., Daniel, F., Facca F., Matera, M.: Model-Driven Engineering of Active Contextawareness. World Wide Web, Vol 10(4), pp. 387-413 (2007)

74. Mattos, T.C., Santoro, F.M., Revoredo, K., Nunes, V.T.: A formal representation for context-aware business processes. Computers in Industry, Vol 65, pp. 1193-1214 (2014)

75. Marrella, A., Mecella, M., Sardina, S.: Intelligent Process Adaptation in the SmartPM System. ACM Transactions on Intelligent Systems Technology, Vol 8(2) (2016)

76. Marrella, A., Mecella, M.: Cognitive Business Process Management for Adaptive Cyber-Physical Processes. In Business Information Processing Workshop Co-located with Business Process Management, Vol 308, Springer Cham. (2018)

77. Nunes, V.T., Santoro, F. M., Werner, C. M. L., Ralha, C. A.: Real-Time Process Adaptation: A Context-Aware Replanning Approach. IEEE Transactions on Systems Man Cybernetics-Systems, Vol 48, pp. 99-118 (2018)

78. Lapouchnian, A., Mylopoulos, J.: Modeling domain variability in requirements engineering with contexts. In Conceptual Modeling Conference, pp. 115-130, Springer Berlin Heidelberg (2009)

79. Lapouchnian, A., Yu. E.: Exploring Context Sensing in the Goal-Driven Design of Business Processes. In IEEE 18th Conference on Business Informatics, Vol. 1, pp. 45-54, IEEE (2016)

80. Kuhrmann, M., Fernández, D. M., Knapp, A.: Who cares about software process modeling? A first investigation about the perceived value of process engineering and process consumption. In Product-Focused Software Process Improvement, pp. 138-152, Springer Berlin Heidelberg (2013)

81. Henderson-Sellers, B., Ralyté, J.: Situational Method Engineering: State-of-the-Art Review. Journal for Universal Computer Science, Vol 16(3), pp. 424–478 (2010)

82. Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0. Retrieved from http://www.omg.org/spec/SPEM/2.0/

83. Jacobson, I., Booch, G., Rumbaugh, J., Rumbaugh, J., Booch, G.: The unified software development process, Vol. 1, Addison-Wesley Reading (1999)

84. Rumbaugh, J., Jacobson, I., Booch, G.: Unified Modeling Language Reference Manual. The Pearson Higher Education (2004)

85. Fernández, D. M., Penzenstadler, B., Kuhrmann, M., Broy, M.: A meta model for artefact orientation: Fundamentals and lessons learned in requirements engineering. In Model-Driven engineering languages and systems, pp. 183-197, Springer Berlin Heidelberg (2010)

86. Pedreira, O., Piattini, M., Luaces, M. R., & Brisaboa, N. R.: A systematic review of software process tailoring. ACM SIGSOFT Software Engineering Notes, Bol 32(3), pp 1-6 (2007)

87. Washizaki, H.: Building software process line architectures from bottom up. In Product Focused Software Process Improvement, pp. 415-421, Springer Berlin Heidelberg (2006)

88. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Kern, J.: Manifesto for Agile Software Development (2001)

89. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall (2002)

90. Madachy, R. J.: Software Process Dynamics. John Wiley & Sons (2007)

91. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. MIT Press (2011)

92. Yu, E., Mylopoulos, J.: Why goal-oriented requirements engineering. In Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality, Vol 15 (1998)

93. 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)

94. Chung, L., Nixon, B. A., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering, Vol 5, Springer Science & Business Media (2012)

95. Horkoff, J., Yu, E.: Analyzing goal models: different approaches and how to choose among them. In Proceedings of the 2011 ACM Symposium on Applied Computing, pp. 675-682, ACM (2011)

96. Ali, R., Chopra, A., Dalpiaz, F., Giorgini, P., Mylopoulos, J., Silva Souza, V.: The evolution of Tropos: Contexts, commitments and Adaptivity (2010)

97. Borgida, A., Mylopoulos, J.: A sophisticate's guide to information modeling (2009)

98. Angelopoulos, K., Souza, V. E. S., Pimentel, J.: Requirements and architectural approaches to adaptive software systems: A comparative study. In Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 23-32, IEEE Press (2013)

99. Sutcliffe, A. G., Maiden, N. A., Minocha, S., Manuel, D.: Supporting scenario-based requirements engineering. IEEE Transactions on Software Engineering, Vol 24(12), pp. 1072-1088 (1998)

100. America, P., Rommes, E., Obbink, H.: Multi-view variation modeling for scenario analysis. In Software Product-Family Engineering, pp. 44-65, Springer Berlin Heidelberg (2004)

101. Jackson, M.: Problem frames: analysing and structuring software development problems. Addison-Wesley (2001)

102. Salifu, M., Yu, Y., Nuseibeh, B.: Specifying monitoring and switching problems in context. In 15th IEEE International Requirements Engineering Conference, pp. 211-220, IEEE (2007)

103. Nuseibeh, B., Easterbrook, S.: Requirements Engineering: A roadmap. In Proceedings of the Conference on the Future of Software Engineering, pp. 35-46, ACM (2000)

104. Hurtado Alegría, J. A., Bastarrica, M. C., Quispe, A., Ochoa, S. F.: An MDE approach to software process tailoring. In Proceedings of the 2011 International Conference on Software and Systems Process, pp. 43-52, ACM (2011)

105. Weiss, D. M.: Software product-line engineering: A family-based software development process. Addison-Wesley (1999)

106. Coplien, J., Hoffman, D., Weiss, D.: Commonality and variability in software engineering. IEEE Software, Vol 15(6), pp. 37-45 (1998)

107. Gomaa, H.: Designing Software Product Lines with UML, pp. 160-216, IEEE (2005)

108. Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K. DeBaud, J. M.: PuLSE: A methodology to develop Software Product Lines. In Proceedings of the 1999 Symposium on Software Reusability, pp. 122-131, ACM (1999)

109. Frakes, W., Prieto, R., Fox, C.: DARE: Domain Analysis and Reuse Environment. Annals of Software Engineering, Vol 5(1), pp. 125-141 (1998)

110. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On goal-based variability acquisition and analysis. In 14th IEEE International Conference on Requirements Engineering, pp. 79-88, IEEE (2006)

111. Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., Peterson, A. S.: Feature-Oriented Domain Analysis (FODA) feasibility study (No. CMU/SEI-90-TR-21). Software Engineering Institute, Carnegie-Mellon University, Pittsburgh PA (1990)

112. Kang, K. C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: A Feature-Oriented Reuse Method with domain-specific reference architectures. Annals of Software Engineering, Vol 5(1), pp. 143-168 (1998)

113. Jacobson, I., Griss, M., Jonsson, P.: Software Reuse: Architecture, process and organization for business success. ACM Press/Addison-Wesley Publishing Co. (1997)

114. Griss, M. L., Favaro, J., Alessandro, M. D.: Integrating feature modeling with the RSEB. In Proceedings of Fifth International Conference on Software Reuse, pp. 76-85, IEEE (1998)

115. Robak, S., Franczyk, B., & Politowicz, K.: Extending the UML for modeling variability for system families. Applied Mathematics and Computer Science, Vol 12(2), pp. 285-298 (2002)

116. von der Maßen, T., Lichter, H.: Modeling variability by UML use case diagrams. In Proceedings of the International Workshop on Requirements Engineering for Product Lines, pp. 19-25 (2002)

117. Simmonds, J., Bastarrica, M. C., Silvestre, L., Quispe, A.: Variability in software process models: Requirements for adoption in industrial settings. In 4th International Workshop on Product Line Approaches in Software Engineering (PLEASE), pp. 33-36, IEEE (2013)

118. Simmonds, J., Bastarrica, M. C.: Modeling variability in software process lines. Departamento de Ciencias de la Computación. Universidad de Chile (2011)

119. Martınez-Ruiz, T., Garcıa, F., Piattini, M., Munch, J.: Modeling software process variability: an empirical study. Software, IET, Vol 5(2), pp. 172-187 (2011)

120. Schnieders, A., Weske, M.: Activity diagram based process family architectures for enterprise application families. In Enterprise Interoperability, pp. 67-76, Springer London (2007)

121. Babar, Z., Lapouchnian, A., Yu, E.: Modeling DevOps Deployment Choices Using Process Architecture Design Dimensions. In The Practice of Enterprise Modeling, pp. 322-337, Springer International Publishing (2015)

122. Andal-Ancion, A., Cartwright, P., Yip, G.: The digital transformation of traditional business. MIT Sloan Management Review, Vol 44(4), pp. 34-41 (2012)

123. Patel, K., McCarthy, M.: Digital transformation: the essentials of e-business leadership. McGraw-Hill Professional (2000)

124. Reis, J., et al.: Digital transformation: A literature review and guidelines for future research. World Conference on Information Systems and Technologies, pp. 411-421, Springer Cham (2018)

125. Henriette, E., Feki, M., Boughzala, I.: The shape of digital transformation: A systematic literature review. Ninth Mediterranean Conference on Information Systems Proceedings, pp. 431-443 (2015)

126. Morakanyane, R., Grace, A. A., O'Reilly P.: Conceptualizing Digital Transformation in Business Organizations: A Systematic Review of Literature. BLED eConference, pp. 427-443 (2017)

127. Bossert, O.: A Two-Speed Architecture for the Digital Enterprise. In Emerging Trends in the Evolution of Service-Oriented and Enterprise Architectures. Intelligent Systems Reference Library, Vol 111, Springer Cham (2016)

128. Haffke, I., Kalgovas, B., Benlian, A.: The transformative role of bimodal IT in an era of digital business. (2017)

129. Tallon, P. P.: Inside the Adaptive Enterprise: An Information Technology capabilities perspective on business process agility. Information Technology and Management, Vol 9(1), pp. 21-36 (2008)

130. Berman, S. J.: Digital transformation: opportunities to create new business models. Strategy & Leadership, Vol. 40(2), pp. 16-24 (2012)

131. Resca, A., Za, S., Spagnoletti, P.: Digital platforms as sources for organizational and strategic transformation: A case study of the Midblue project. Journal of Theoretical and Applied Electronic Commerce Research, Vol. 8(2), pp. 71-84 (2013)

132. Lan, F., Liu X.: Business model transformation in digital enablement context through frugal innovation: Learning from Chinese experience. International Journal of Technology, Policy and Management, Vol. 17(4), pp. 360-373 (2017)

133. Schallmo, D., Williams, C. A., Boardman, L.: Digital transformation of business models—Best practice, enablers, and roadmap. International Journal of Innovation Management, Vol. 21(08) (2017)

134. Remane, G., et al.: Discovering digital business models in traditional industries. Journal of Business Strategy, Vol 38(2), pp. 41-51 (2017)

135. Kotarba, M.: Digital transformation of business models. Foundations of Management, Vol. 10(1), pp. 123-142 (2018)

136. Matt, C., Hess, T., Benlian, A.: Digital transformation strategies. Business & Information Systems Engineering, Vol. 57(5), pp. 339-343 (2015)

137. Loonam, J., et al.: Towards digital transformation: Lessons learned from traditional organizations," Strategic Change, Vol. 27(2), pp. 101-109 (2018)

138. Delmond, M., et al.: How Information Systems Enable Digital Transformation: A focus on Business Models and Value Co‐production. HEC Paris Research Paper No. MOSI-2016-1161 (2016)

139. Earley, S.: The digital transformation: staying competitive. IT Professional, Vol. 16(2), pp. 58-60 (2014)

140. Berman, S., Marshall, A.: The next digital transformation: from an individual-centered to an everyone-to-everyone economy. Strategy & Leadership, Vol. 42(5), pp. 9-17 (2014)

141. Westerman, G.: Why digital transformation needs a heart. MIT Sloan Management Review, Vol. 58(1), pp. 19 (2016)

142. Hossain, M., Lassen, A. H.: How Do Digital Platforms for Ideas, Technologies, and Knowledge Transfer Act as Enablers for Digital Transformation?. Technology Innovation Management Review, Vol. 7(9), pp. 55-60 (2017)

143. Heavin, C., Power, D. J.: Challenges for digital transformation–towards a conceptual decision support guide for managers. Journal of Decision Systems, Vol. 27(1), pp. 38-45 (2018)

144. Andriole, S. J.: Skills and Competencies for Digital Transformation. IT Professional, Vol 20(6), pp. 78-81 (2018)

145. Burden, A., et al.: Technical debt might be hindering your digital transformation. MIT Sloan Management Review, Vol. 60(1), pp. 1-5 (2018)

146. Shrivastava, S.: Digital Disruption is Redefining the Customer Experience: The Digital Transformation Approach of the Communications Service Providers. Telecom Business Review, Vol. 10(1), pp. 41 (2017)

147. Narayanan, V. K.: Customer-focused IT: a process of continuous value innovation. Strategy & Leadership, Vol 43(4), pp. 11-17 (2015)

148. Kaivo-Oja, J., Roth, S., Westerlund, L.: Futures of robotics. Human work in digital transformation. International Journal of Technology Management, Vol. 73(4), pp. 176-205 (2017)

149. Shaughnessy, H.: Creating digital transformation: strategies and steps. Strategy & Leadership, Vol 46(2), pp. 19-25 (2018)

150. Wahi, A. K., Medury, Y.: Digital businesses: Creation of a research framework for organizational readiness for Enterprise 2.0. Big Data: Concepts, Methodologies, Tools, and Applications. IGI Global, pp. 1832-1858 (2016)

151. Weill, P., Woerner, S. L.: Is Your Company Ready for a Digital Future?. MIT Sloan Management Review, Vol. 59(2), pp. 21-25 (2018)

152. Masuda, Y., et al.: Architecture board practices in adaptive enterprise architecture with digital platform: A case of global healthcare enterprise. International Journal of Enterprise Information Systems, Vol. 14(1), pp. 1-20 (2018)

153. Troilo, G., De Luca, L. M., Guenzi, P.: Linking data‐rich environments with service innovation in incumbent firms: A conceptual framework and research propositions. Journal of Product Innovation Management, Vol 34(5), pp. 617-639 (2017)

154. Gölzer, P., Fritzsche, A.: Data-driven operations management: organisational implications of the digital transformation in industrial practice. Production Planning & Control, Vol 28(16), pp. 1332-1343 (2017)

155. Kolbjørnsrud, V., Amico, R., Thomas, R. J.: Partnering with AI: how organizations can win over skeptical managers. Strategy & Leadership, Vol 45(1), pp. 37-43 (2017)

156. Pikkarainen, M., et al.: Data as a driver for shaping the practices of a preventive healthcare service delivery network. Journal of Innovation Management, Vol. 6(1), pp. 55-79 (2018)

157. Masuda, Y., et al.: An adaptive enterprise architecture framework and implementation: Towards global enterprises in the era of cloud/mobile IT/digital IT. International Journal of Enterprise Information Systems, Vol 13(3), pp. 1-22 (2017)

158. Basole, R. C.: Accelerating digital transformation: Visual insights from the API ecosystem. IT Professional, Vol 18(6), pp. 20-25 (2016)

159. Alos-Simo, L., Verdu-Jover, A. J., Gomez-Gras, J.: How transformational leadership facilitates e-business adoption. Industrial Management & Data Systems, Vol 117(2), pp. 382-397 (2017)

160. Ardolino, M., et al.: The role of digital technologies for the service transformation of industrial companies. International Journal of Production Research, Vol 56(6), pp. 2116-2132 (2018)

161. Schwarzmüller, T., et al.: How does the digital transformation affect organizations? Key themes of change in work design and leadership. Mrev management revue, Vol 29(2), pp. 114-138 (2018)

162. Sainger, G.: Leadership in Digital Age: A Study on the Role of Leader in this Era of Digital Transformation. International Journal on Leadership, Vol 6(1), pp. 1 (2018)

163. Nwaiwu, F.: Review and Comparison of Conceptual Frameworks on Digital Business Transformation," Journal of Competitiveness. Vol. 10(3), pp. 86-100 (2018)

164. Weber, M. S., Monge, P. R.: Industries in turmoil: Driving transformation during periods of disruption. Communication Research, Vol 44(2), pp. 147-176 (2017)

165. Andriole, S. J.: Five myths about digital transformation. MIT Sloan Management Review, Vol. 58(3) (2017)

166. Newman, S.: Building Microservices. O'Reilly Media, Inc. (2015)

167. Bosch, J. (Ed.): Continuous Software Engineering. Springer (2014)

168. Ståhl, D., Bosch, J.: Modeling continuous integration practice differences in industry software development. Journal of Systems and Software, 87, pp. 48–59 (2014)

169. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Using scrum in distributed agile development: A multiple case study. In Fourth IEEE International Conference on Global Software Engineering, pp. 195–204, IEEE (2009)

170. 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)

171. Van Lamsweerde, A.: Goal-oriented requirements engineering: A guided tour. In Fifth IEEE International Symposium on Requirements Engineering, pp. 249-262, IEEE (2001)

172. Horkoff, J., Yu. E.: Comparison and evaluation of goal-oriented satisfaction analysis techniques. Requirements Engineering, Vol 18(3), pp. 199-222 (2013)

173. Emeakaroha, V. C., Brandic, I., Maurer, M., Breskovic, I.: SLA-aware application deployment and resource allocation in clouds. In 35th Annual IEEE Computer Software and Applications Conference Workshops, pp. 298-303, IEEE (2011)

174. Xu, Y., Chen, N., Fernandez, A., Sinno, O., Bhasin, A.: From infrastructure to culture: A/B testing challenges in large scale social networks. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2227-2236, ACM (2015)

175. Pinheiro, C., Vasconcelos, A., & Guerreiro, S.: Microservice Architecture from Enterprise Architecture Management Perspective. In International Symposium on Business Modeling and Software Design, pp. 236-245, Springer Cham (2019)

176. Taibi, D., Lenarduzzi, V., Pahl, C.: Architectural Patterns for Microservices: A Systematic Mapping Study. In CLOSER, pp. 221-232 (2018)

177. Seidewitz, E.: What models mean. IEEE Software, Vol 20, pp. 26-32 (2003)

178. NIST. Integration Definition for Function Modeling (IDEF0), 1993. Retrieved from http://www.idef.com/pdf/idef0.pdf

179. CMMI for Development, Technical Report Version 1.3. Retrieved from https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf

180. Simon, H. A.: The sciences of the artificial. 3rd Edition, MIT Press (1996)

181. Baldwin, C. Y., Clark, K. B.: Design rules: The power of modularity, Vol 1, MIT Press (2000)

182. Garud, R., Jain, S., Tuertscher, P.: Incomplete by design and designing for incompleteness. Organization studies, Vol 29(3), pp. 351-371 (2008)

183. Vernon, D.: Artificial cognitive systems: A primer, MIT Press (2014)

184. Dix, A. Human-computer interaction. In Encyclopedia of database systems, pp. 1327-1331, Springer (2009)

185. Shedroff, N.: Information interaction design: A unified field theory of design. Information design, pp. 267-292 (1999)

186. Bouquet, P., Ghidini, C., Giunchiglia, F., Blanzieri, E.: Theories and uses of context in knowledge representation and reasoning. Journal of Pragmatics, Vol 35(3), pp. 455-484 (2003)

187. Rosemann, M., Recker, J.: Context-aware process design: exploring the extrinsic drivers for process flexibility. In Proceedings of the International Conference on Business Process Modeling, Development and Support, Luxembourg (2006)

188. Koliadis, G., Ghose, A.: Relating Business Process Models to Goal-Oriented Requirements Models in KAOS. In: Advances in Knowledge Acquisition and Management. PKAW 2006. Lecture Notes in Computer Science, Vol 4303. Springer, Berlin, Heidelberg (2006)

189. Nurcan, S., et al.: A strategy driven business process modeling approach. Business Process Management Journal, Vol 11(6), pp. 628–649 (2005)

190. Gandomi, A., Haider, M.: Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management, Vol 35(2), pp. 137-144 (2015)

191. IBM Business Automation Workflow. Retrieved from https://www.ibm.com/ca-en/marketplace/ibm-business-automation-workflow

192. Gartner Peer Insights, Intelligence Business Process Management Suites Market. Retrieved from https://www.gartner.com/reviews/market/intelligent-business-process-management-suites

193. Ogiela, L., Ogiela, M. R.: Advances in cognitive information systems. Springer Science & Business Media, Vol 17 (2012)

194. Ricci, F., Rokach, L., Shapira, B.: Introduction to Recommender Systems Handbook. In Recommender Systems Handbook, pp. 1-35, Springer US (2011)

195. Chen, P.: The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, Vol 1(1), pp. 9-36 (1976)

196. Object Management Group. Unified Modeling Language (UML), Version 2.5, 2015. Retrieved April 29, 2016 from www.omg.org/spec/UML/2.5.

197. Barros, O.: Business process patterns and frameworks: Reusing knowledge in process innovation. Business Process Management Journal, Vol 13(1), pp. 47-69 (2007)

198. Eriksson, H. E., & Penker, M.: Business modeling with UML, pp. 1-12, New York (2000)

199. Perroud, T., & Inversini, R.: Enterprise architecture patterns: Practical solutions for recurring IT-architecture problems. Springer Science & Business Media (2013)

200. Gross, D., Yu, E.: From non-functional requirements to design through patterns. Requirements Engineering, Vol 6(1), pp. 18-36 (2001)

201. Cunha, H., Sampaio do Prado Leite, J. C.: Reusing non-functional patterns in i∗ Modeling. In IEEE 4th International Workshop on Requirements Patterns (RePa), pp. 25-32, IEEE (2014)

202. Pree, W., Gamma, E.: Design patterns for Object-Oriented Software Development, Vol 183, Addison-Wesley, Reading MA (1995)

203. Johnson, R., Vlissides, J.: Design patterns. Elements of Reusable Object-Oriented Software Addison-Wesley, Reading MA (1995)

204. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S.: Facing the cold start problem in recommender systems. Expert Systems with Applications, Vol 41(4), pp. 2065-2073 (2014)

205. Kueng, P., Kawalek, P.: Goal-Based Business Process Models: Creation and Evaluation. Business Process Management Journal, Vol 3(1), pp. 17-38 (1997)

206. Paulk, M. C., Curtis, B., Chrissis, M. B., Weber, C. V. (1993). Capability Maturity Model for Software, Version 1.1, Technical Report. SEI - Carnegie Mellon University.

207. García-Borgoñon, L., Barcelona, M. A., García-García, J. A., Alba, M., & Escalona, M. J. (2014). Software process modeling languages: A systematic literature review. Information and Software Technology, 56(2), 103-116.

208. Cugola, G., & Ghezzi, C. (1998). Software Processes: a Retrospective and a Path to the Future. Software Process: Improvement and Practice, 4(3), 101-123