Re-Designing Process Architectures

Towards a Framework of Design Dimensions

Alexei Lapouchnian, Eric Yu
University of Toronto
Toronto, Canada
alexei@cs.toronto.edu, eric.yu@utoronto.ca

Arnon Sturm
Ben-Gurion University of the Negev
Beer Sheva, Israel
sturm@bgu.ac.il

***Abstract*—Organizations rely on a multiplicity of processes covering everything from their day-to-day functioning to longer term viability. Together, these processes and their interrelationships constitute the business process architecture (BPA) of the organization. While efforts have been dedicated to the analysis and design of business processes, the question of how processes in an organization should best relate to each other (i.e., the design of the BPA) has received relatively little consideration. Supported by technological and business innovations, the torrent of changes faced by today's organizations, forces them to stop looking at their processes individually and focus on designing BPAs, especially concentrating on balancing flexibility/agility and other objectives, such as cost and efficiency. In this paper, we propose a framework for BPA design with several dimensions along which activities or decisions could potentially be repositioned across processes and a goal-driven approach for analyzing possible BPA configurations.**

Keywords—variability; flexibility; business process architecture; process modeling; requirements

I. Introduction

Organizations rely on many business processes (BPs) for their operational activities and for longer term viability and sustainability. These BPs and their relationships form the *process architecture* (BPA) of the organization.

Extensive efforts have been devoted to analyzing and designing individual BPs. In contrast, relatively little attention has been paid to designing BPAs, i.e., how the many BPs of an organization should *best relate to each other* to achieve the company's objectives. As organizations experience massive changes due to technological and business model innovation, one cannot optimize BPs in isolation from each other. BPAs can no longer remain static, but frequently need to be rethought and re-engineered. New capabilities are continuously created, challenging old approaches to innovation. Innovation cycles become shorter, with development being integrated with operations. Users and their usage processes are being coupled with development processes in a virtuous cycle of value co-creation.

Similarly, the boundary between planning and execution processes is being redrawn as analytics and massive amounts of data (e.g., from sensors or social networks) support greater context-awareness, so decisions that used to be pre-planned are increasingly made at runtime to improve responsiveness.

Thus, modern agile organizations cannot take existing BPAs for granted. The process architect should be asking questions such as: Should some activities or decisions be deferred closer to the frontline, taking advantage of near-real-time data, to better meet customer needs and wants? Should more or fewer activities/decisions be pre-planned? Should activities previously performed for an aggregate be performed per instance instead?

BPA design involves making trade-offs across BPs, particularly regarding how to balance flexibility and agility with other design objectives such as cost and efficiency. Existing models of BPA (e.g., [3] (Ch. 2), enterprise architecture (EA) frameworks [13], etc.) typically treat a BPA as a given, or as something to be discovered with no conscious effort to design it. We believe there is need to explore the space of BPA alternatives and to guide the selection among them, recognizing the complex trade-offs that may exist. To address the need for a conceptual model, we propose four BPA analysis dimensions that could serve as the key elements of such a framework.

To illustrate our approach, we chose the easy to understand and sufficiently rich domain of passenger transportation. It continues to experience change today, driven by many innovations and shifting attitudes. Among the driving forces in this domain are the focus on reducing energy consumption and emissions, traffic congestion, availability of location sensors and real-time traffic information, increased customer expectations of flexibility and convenience, etc. Patterns and issues identified here can easily be found in other domains.

This paper is organized as follows. In Section II, we outline the architectural design space. In Section III, we discuss the dimensions comprising this space. Section IV talks about analyzing BPA alternatives, while Section V discusses related work. Section VI presents the discussion and concludes.

II. A Design Space for Process Architectures

When working towards a framework for BPA design, we focus on the space of possible architectural designs and thus need a modeling notation to capture possible BPA configurations and a method to reason about them.

We consider BPA modifications as movements along four dimensions: (1) *temporal*: moving a *process element* (PE) – an activity or a decision – earlier or later in relation to other process elements; (2) *recurrence*: positioning a PE in a BP that is repeated more (or less) frequently with respect to other PEs; (3) *plan/execution*: positioning a PE on the planning or execution side of a process, i.e., whether the PE is done during planning or during the execution of the resulting plan; (4) *design/use*: positioning a PE on the design or usage side of a process, i.e., whether the PE is part of a design process or is invoked during the usage of the outcome of the design – an artifact or capability.

978-1-4673-6630-4/15/$31.00 ©2015 IEEE.

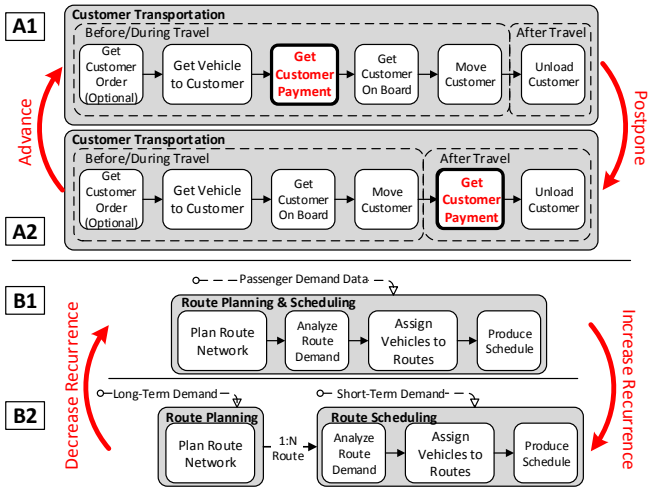

Fig. 1. Trip payment options: (A1) Before trip's end. (A2) At trip's end. Splitting a single stage (B1) into multiple ones (B2) to increase flexibility or merging them to improve stability.

In considering the positioning of PEs along these dimensions, we aim to address a major concern of BPA design – the tension between flexibility and efficiency: an organization in a static domain can have all of its BPs tightly coupled and globally optimized once and for all. In a dynamic domain, however, the BPA needs flexibility for the organization to respond to change.

A fundamental idea for accommodating uncertainty is to keep options open. This supports flexibility and allows for alternate courses of action depending on the context, while also incurring extra costs. So, determining where/when options should be kept open is a core mechanism for BPA design. It needs to be supported by a modeling notation and a reasoning framework.

We use the well-known term *variation point* (VP) (e.g., [4]) to refer to a place in a process where multiple options exist. A *variant* refers to each individual option in a VP. A VP is *bound* when one of its variants is selected. When and where a VP is bound underlies much of the reasoning behind the positioning of a PE along the four dimensions.

These dimensions were determined based on existing studies, our own experiences, and the analysis of existing BPAs. Their purpose is to define the architecture design space. We do not claim that these are the only possible dimensions. However, we found them suitable for characterizing BPA alternatives.

As in software architecture, the architectural description we strive for should outline the major elements and relationships while avoiding over-specification. Thus, it will likely refer only to certain selected elements of BP specifications.

III. Dimensions for Process Architecting

A. The Temporal Dimension

In general, there are multiple possible placements for PEs within a BP/system specification that comply with the existing functional dependencies, achieve the same functional objective, but differ in terms of their non-functional characteristics.

The passenger trip payment options in Fig. 1A illustrate this. E.g., unlike the standard fare charged before a trip, a distance-based fare can only be reliably charged after the trip. Both obtain payment, but differ in the amount charged, how fair and precise the charge is, etc. Thus, there may be many options along the temporal dimension in BPs where PEs can be placed. These choices need to be resolved by looking at how each variant affects the quality criteria that the organization is interested in.

Introducing phases. What is better – to charge the customer before a trip, during the trip or after travel? In fact, this depends on one's point of view. Aiming for payment fairness, we want distance-based fares (this is how taxis operate). Here, the system needs a richer context (the distance travelled) only available at trips' end. So, for fairness, paying on exit is better than other options. We identify *phases* – portions of a BP such that placing a PE anywhere within them makes no difference w.r.t. the evaluation criteria, e.g., charging passengers at any time before a trip's end is the same for fairness as the distance is unknown (Fig. 1A). However, moving PEs across *phase boundaries* may affect the quality of decisions and the outcome of actions. Unlike most software variability approaches, we analyze when (in which phase) it is best to execute PEs. Note that no reuse happens here: an output of a phase of some BP instance can only be used by the subsequent phases of the same instance. Overall, *phases* help reduce the analysis complexity and focus on the relevant issues by abstracting from lower-level details.

Postponement. Here, we look at choices to postpone or advance PEs by placing them into later or earlier phases respectively (see Fig. 1A). *Postponement* is a well-known business strategy (e.g., in supply chains [9]) that aims to minimize risks and maximize benefits by delaying some activities/decisions that require up-to-date information until the last possible moment. The key is the expectation of more precise information to be available at a later point, which would allow for better, more context-sensitive outcomes. Conversely, *advancement* improves stability and uniformity and is enabled by either coarser-grained PEs that rely on less information and hence can tolerate uncertainty or by predicting the missing information (e.g., through predictive analytics). The availability of data required for postponing PEs and data collection/analysis costs are also important. E.g., to collect distance-based fares at the end of trips, the organization needs to develop and deploy the infrastructure to measure the distance travelled by each passenger.

B. The Recurrence Dimension

In the previous section, we assumed that PEs were executed for every BP instance. Here, we propose another dimension focusing on reusing the outcomes of decisions/activities in multiple BP instances. To put it another way, how often should certain decisions or actions be (re)executed and under what conditions?

Definitions. We group PEs that have the same execution cycle into process chunks called *stages*. A stage contains one or more phases (e.g., in Fig. 1A, Customer Transportation is a stage consisting of two phases). Once a stage executes, its output remains available to the subsequent stages, if any, until it is re-executed. In our notation, a stage connects to its subsequent stage(s) using a control flow link (a solid line) labeled with "1:N" to indicate the cardinality of their relationship (see Fig. 1B). In this case, a *stage boundary* exists between the stages. It points to the two options for placing PEs – each with a different recurrence pattern. Moving a PE across such a boundary can lead to a significant change in the PE's execution frequency. Fig.

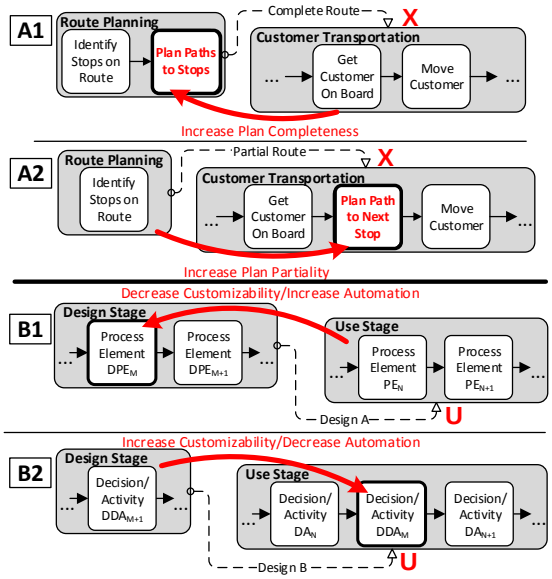

Fig. 2. Complete (A1) or partial (A2) planning choices for route planning. Moving process elements across a design/use boundary (B).

1B shows such movement in both directions for Route Scheduling. A PE can also be moved across more than one stage at a time. A stage represents a (sub-)process and thus, while the temporal dimension focuses on intra-process analysis, here the focus is on inter-process relationships – relative rates of execution/change cycles among processes.

Using the dimension. To arrive at a stage-based configuration we identify PEs that can be reused for multiple BP instances (i.e., that are independent of the variations in those instances), at least for a period of time. Such PEs form a stage to be reused by the subsequent stages, thus saving time, money and/or other resources and perhaps affecting other non-functional objectives (NFRs). E.g., buying a transit pass improves convenience over paying for every trip separately. Another heuristic for stages is to identify PEs with the same execution frequency (e.g., yearly product redesign cycles accompanied by marketing material revisions) or triggered by the same condition (e.g., a change in passenger demand triggers a bus schedule revision, which is then immediately published). A special case of moving a PE across a stage boundary is splitting a stage into two or merging two stages into one (see Fig. 1B).

Domain example. We now look at how public transit route planning and vehicle scheduling can be done. One option is to combine both into a single stage (Fig. 1B1): whenever routes need to be redrawn (e.g., due to a significant demand change), Route Planning & Scheduling stage is triggered. The data input for the stage (the message flow at the top) has all the available passenger demand data. Here, route planning and scheduling are bundled together, which means that changing schedules without a route network redesign is not permitted. Clearly, this option works for an easily predictable constant demand. However, its rigidity will hurt the company's ability to change its schedules more frequently (while keeping the same routes) in case of evolving passenger demand. To address this, the process architect can unbundle the two stages as shown in Fig. 1B2, creating Route Planning and Route Scheduling, each triggered independently: the former when changes in long-term demand are detected and the latter when shorter-term demand changes are perceived. This BPA change allows the route network (the outcome of the Route Planning stage) to be reused for multiple Route Scheduling instances (again, note the "1:N" annotation), thus supporting frequent schedule updates in the same route network. The new BPA configuration is more flexible, but likely incurs higher cost, complexity and unpredictability.

Overall, given a number of stages as possible PE placement choices, an organization wants the one that best fits its needs based on the volatility in its business environment – e.g., balancing the cost, complexity, and the increased unpredictability of the unbundled route planning configuration, and the inherent rigidity of the bundled one from Fig. 1B1.

C. The Plan/Execute Dimension

In typical BP modeling, a model describes a process to be executed, but not how it gets determined. For enterprise agility, it matters if a decision/activity is part of a plan, or is instead left to runtime. Thus, our framework supports explicit modeling of planning and plan execution and reasoning about the positioning of PEs on either side of the plan/execute boundary.

Stages can generate plans/specifications for the subsequent stage(s) to execute. Such plans either fully specify or only constrain the subsequent stages. In most cases, plans are reused. A stage producing a plan is called a *planning stage* (PS). A stage using the plan is called an *execution stage* (ES). PS and ES are relative to each other: a stage can be both a PS and an ES with respect to several different stages.

Full and partial plans. We allow both complete and partial plans. Complete plans fully specify execution and thus are restrictive and inflexible, but require no further planning in the ES, which lowers the demand on that stage, helps uniformity and predictability, and supports optimization at the aggregate level. In Fig. 2A, the (bus) route PS plans both the stops and the precise paths between them. No extra planning is needed when driving on a route. This simplifies a driver's job, but changing a route to avoid traffic or other problems is impossible. Data flow links annotated with X (for "eXecution") and the plan to be executed link PSs and ESs (e.g., Complete Route in Fig. 2A1).

A partial plan helps separate stable and more dynamic things and allows for a range of behaviours in the ES since some previously planned choices are left open by moving them from a PS to an ES. In Fig. 2A2, the planning of paths between stops is moved across the plan/execute boundary to Customer Transportation stage, aiding minor route adjustments to handle unpredictable changes. The driver's job, however, is harder as he needs to monitor/analyze road conditions and calculate updated routes. Overall, decreasing plan completeness helps flexibility but will likely incur higher costs and effort in the ES since it has to complete the partial plan based on the current context.

D. The Design/Use Dimension

The power of technology relies crucially on the creation of long-term capabilities to be used by others. In today's process/enterprise models, tools, designs, etc. can be represented and utilized as modeling artifacts, though these capabilities are viewed as externally developed and thus not changeable. To support enduring enterprises and IT systems, however, one needs to represent those artifacts as evolvable objects that can

be redesigned to handle changes in business domains/requirements. Models of design, development or other tool/skill/etc. acquisition processes need to be integrated into EAs to support the capture of evolving capabilities and for continuous design [6]. This helps the analysis of tool (re)design cycles and the rigidities or flexibilities due to the tools used.

Definitions. Designs, tools or skills are produced by a *design stage* (DS) and are used in a *use stage* (US), see Fig. 2B. A design/use boundary is a stage boundary since a tool can be reused. DSs are linked to USs through data flows annotated with U (for "use"), entering the latter at the bottom. Tools help USs achieve their business/system goals and the tools' use is not restricted and their internals are not known. For example, a user of an automatic transmission does not need to know how it operates. It is less flexible and less fuel-efficient than a manual one, but it is simpler to operate. The BPA variants here are about the flexibility of a tool produced in a DS – i.e., how generic or single-purpose it is. The more single-purpose (less flexible) the tool is, the simpler it is to use. However, such a tool cannot be customized to take advantage of the available usage context information.

Fig. 2B shows the crossing of the design/use boundary. Fig. 2B1 moves a PE from a US to a DS. For an activity, this increases the level of automation in the US. For a decision, it reduces the tool's customizability (Design A). In contrast, moving a PE to the US (Fig. 2B2) reduces the automation while improving the tool's customizability. An example of such a choice is having a fixed vs. variable number of cars in subway trains.

In addition to flexibility/customizability, other important factors (NFRs) help in analyzing choices in this dimension. Cost is incurred through tool design/development or acquisition. Skills development also incurs costs. Tool renting/leasing and the use of (IT) services is a way to avoid high upfront tool costs.

E. A BPA Model with Relationships from Four Dimensions

Fig. 3 has a fragment of a BPA for a transit company showing various relationships among stages (we abstract from phases here) and implementing the more flexible options from the examples discussed in this section. The model also captures data flows among stages. For a stage, its data input comes from another stage (e.g., Required Capacity) or from the outside through monitoring or by other means (e.g., Social Network Data). A BPA model gives an overview of the organization and its BPs, shows where reuse happens, where plans are created/executed and how tools are utilized. The model shows *one possible* BPA configuration. It is the basis for what-if analysis, with other configurations obtained by moving PEs among stages/phases (see below for discussion). We assume that the starting point for developing a BPA using our notation is an existing BPA in another notation or a collection of BPs. We are working on a goal-driven approach to help design the initial BPA using ideas from [5].

IV. Analyzing Process Architecture Alternatives

A. Representing and Analyzing Business Process Architecture Alternatives with Goal Models

We use goal models (GMs) to represent and analyze choices in BPAs along the four dimensions. GMs are used in Requirements Engineering to capture stakeholder/system objectives. They can represent variability in achieving goals using OR decompositions (here, exclusive ORs) and use NFRs to analyze alternative goal refinements. We adapted GMs to represent alternative placements of PEs along the four dimensions. A GM focuses on a single PE (e.g., see Fig. 3), so multiple such models will be used to analyze different portions of the architecture.

Identifying BPA choices. We look at the transit trip payment options discussed in Section III.A to show how a GM is developed and utilized. A GM provides an intentional (vs. operational) view, captures the PE's goal as the root node (see the top Get Customer Payment node in Fig. 4A), and shows how it is refined when achieved in various stages/phases of the BPA. E.g., Fig. 4A shows three phases of Customer Transportation stage where Get Customer Payment PE can be placed. We use the @P:PhaseName and @S:StageName annotations to express locations in the BPA. Each goal refinement explores alternative PE placements within a BPA along one dimension, which is indicated using annotations ([T] for temporal, [R] for recurrence, [D] for Design/Use, and [P] for Plan/Execute, as in Fig. 4).

How we identify choices for goal refinements is based on the dimension under consideration. For the recurrence dimension, the alternative refinements are the stages where the PE can be placed (existing functional constraints must be respected). The temporal dimension focuses on placing PEs into phases. In Fig. 4A, customers can pay before (@P:Before Travel), during or after travel. Prior to selecting the phase for a PE we need to determine its stage. Hence, a frequent pattern in BPA-level analysis is [R] followed by [T]. For the Plan/Execute and Design/Use dimensions, the choice is binary – whether the PE is in or out of a particular PS or DS. A plan/design is available to all the subsequent stages, so we need to select the best one among them. Fig. 4B shows a generic pattern of analysis for the Plan/Execute case.

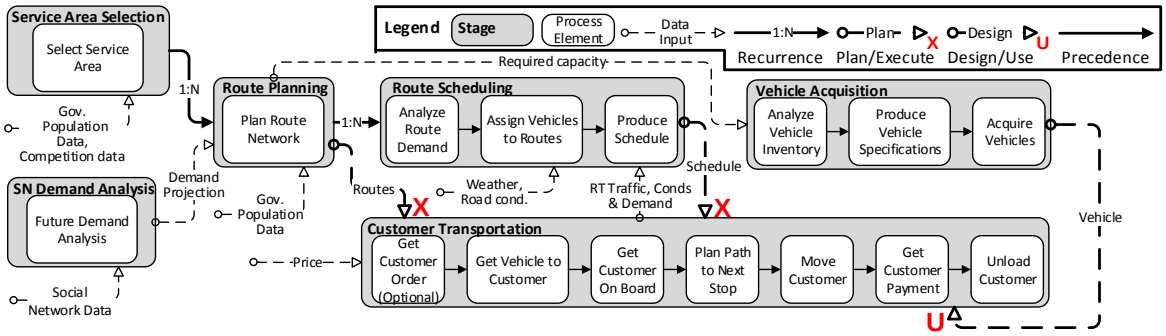

Fig. 3. A fragment of the business process architecture for a public transit company.

We separate a PE's objective (see the root goal in Fig. 4A) from its implementation (the subgoals in Fig. 4A) since the latter may depend on the stage/phase – e.g., a fare payment may be implemented differently in various places of a BPA. In our GMs, this is captured by varying parameters in goal nodes representing alternative PE placements. These depend on the available data – only when paying after travel, the travelled distance is known and thus a fairer price can be determined (note the contribution link). Moving towards later stages/phases (i.e., increasing recurrence or postponement respectively) increases contextualization, thus increasing the number of goal parameters.

Using NFRs. Finally, we elicit NFRs for evaluating PE placement choices. These are modeled as softgoals and evaluated using *contribution links* (e.g., see [5]). The evaluation can be qualitative, with values such as help(+)/hurt(–), make(++)/break(--) (as in Fig. 4A). A softgoal is *satisficed* if there is enough positive and little negative evidence for this claim. Given more information, more precise evaluations are possible. Softgoal prioritization helps with conflicting NFRs that cannot be achieved simultaneously.

Handling trade-offs. Moving PEs along the four dimensions affects many NFRs. Some are domain/PE-independent. Common NFRs of this type and the effects of moving PEs on them are shown in Table 1. Others are domain-specific, relevant for a particular PE (e.g., security and fairness for fare payment, Fig. 4A). Trade-offs are resolved based on two things: 1) business domain dynamics (what changes, how frequently, etc.) and 2) the enterprise's prioritization of the above-mentioned NFRs.

Once the analysis of BPA alternatives is done, a place (stage/phase) in the BPA is identified for the PE under consideration. This represents the delta between the as-is and to-be BPAs – an instruction for evolving the architecture.

B. Business Process Architecture Adaptation

We strive to help organizations determine the right balance of flexibility and stability in their BPAs based on their business domain volatility and their preferences/priorities.

Change can have two types of effect on a BPA. A particular BPA has some amount of flexibility to support certain types of changes in the domain (e.g., a distance-based fare payment is flexible in supporting payments for varying trip distances). If flexibility currently available in a BPA can accommodate the change (the different trip lengths), no BPA reconfiguration is needed. However, if the assumption that passengers' trips are of very different lengths (which justifies the added complexity/cost

TABLE I. EFFECTS OF MOVING PES ALONG VARIABILITY DIMENSIONS

Dimension	PE Movement	Effect of Movement on NFRs
Temporal	Postpone	+: flexibility, context-awareness; –: cost, complexity, stability
	Advance	+: cost, complexity, stability; –: flexibility, context-awareness
Recurrence	Increase Recurrence	+: flexibility; –: cost, reuse, stability
	Decrease Recurrence	+: cost, reuse, stability; –: flexibility
Plan/ Execute	Move to Plan	+: plan completeness, stability
	Move to Execute	+: plan partiality, flexibility
Design /Use	Move to Design	+: automation; –: customizability
	Move to Use	+: customizability; –: automation

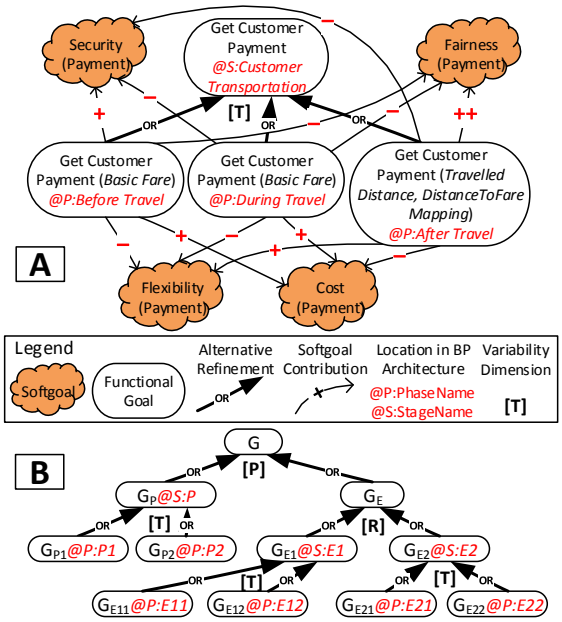

Fig. 4. (A) Analyzing the temporal ([T]) placement of customer payment PE. (B) Generic goal model for positioning a PE with the objective G along the Plan/Execute dimension (goal parameters are omitted).

of distance-based fares) no longer holds, that payment option becomes *too flexible* for the domain and the BPA may need to be changed. Thus, a BPA reconfiguration takes place when the domain dynamics changes – i.e., not when *some* change happens, but when the *rate (or range) of change* becomes different.

V. RELATED WORK

The notion of a BPA has been discussed for a while (e.g., [3]). Over the years, various sets of relationships among BPs have been identified (e.g., [2][3]), including sequence, hierarchy, composition, trigger, etc. In [2], the authors also classified approaches for designing BPAs: goal-based, action-based, object-based, and function-based. All approaches utilize a subset of these relationships when analyzing BPAs.

In the enterprise architecture area, the notion of BPA is referred to as BP cooperation. E.g., in ArchiMate [8] such cooperation includes *causal* relationships between BPs, *mapping* of BPs onto business functions, *realization* of services by BPs, and the *use of shared data*. These aspects can also imply the type of relationships among BPs.

Other relevant domains are BP variability [1], which focuses on the realization relationships among BPs and on BP binding times, and software product lines. A major reason to support variability is to postpone concrete design decisions to the latest economically feasible point [10]. In general, early binding facilitates static analysis, while late binding enables user configuration and post-deployment updates [1]. Positioning VPs earlier in the process can improve efficiency by decreasing uncertainty and identifying redundancies [11]. Intentional variability approaches [5] look at the many ways of achieving organizational objectives as a means to develop customizable, adaptive, and evolving systems (e.g., [5]). Other approaches weaving requirements and BPs have also emerged – e.g., using NFRs and contexts for runtime BP configuration [12].

In process-aware systems, Weber et al. [14] identify four dimensions of change. They use the notion of patterns for changes in predefined regions to define these dimensions and include (1) late selection of BP fragments, (2) late modeling of BP fragments, (3) late composition of BP fragments, and (4) the multi-instance activity. This points to the new BP relationships: *creation*, where a BP creates another one, and *recurrence*, where a BP is followed by another BP multiple times.

Overall, the existing approaches focus on variability at the BP level, while we concentrate on the BPA-level variability. Each set of choices for placing PEs into stages/phases (see Fig. 4) is a VP at the BPA level. Changing the binding of such a VP produces modified BPAs with new characteristics. These BPAs are then analyzed using a goal-driven approach focusing on trade-offs among NFRs. This analysis is missing from the other approaches. For PEs that are decisions (BP-level VPs), positioning them into phases/stages defines *when* and *how often* respectively to bind them, thus creating domain-specific binging options that are much richer and more flexible than the design time/runtime choices usually discussed in the variability research (e.g., [4]). In addition, the Plan/Execute and Design/Use dimensions that allow for modeling BPA evolution are also missing from the current methods.

VI. Concluding Discussion and Future Work

The four dimensions for BPA design help both adaptation and evolution of BPs and BPAs within enterprises. The temporal and recurrence dimensions select, among the configurations supported by the existing set of BPs and enterprise/IT capabilities, the one that best matches the current/expected domain dynamics. The other dimensions look at the options for evolving BPAs to accommodate more significant/unpredictable changes through changing plans/capabilities.

A good BPA must reflect its domain. Assumptions about the domain dynamics, including the availability and volatility of data, need to be carefully analyzed to justify the BPA and its flexibility. Such assumptions are not yet captured in our approach. We currently work on their formal modeling, which will help specify conditions that trigger adaptations – both supported by the current BPA and those needing a change in the BPA.

It is easy to see that the analysis in Section IV favours optimization at the level of individual PEs, likely at the cost of global optimality. We are working on better integrating multiple goal models of the type shown in Fig. 4A to alleviate this.

We allow for some model incompleteness to reduce the modeling/analysis complexity. E.g., our BPA analysis requires GMs to be constructed only for the PEs in the volatile portions of the domain, where flexibility is needed to accommodate change. Also, many finer-grained BP modeling details are below our threshold of interest, which is a phase. To help with the goal-based analysis, we plan to utilize both top-down and bottom-up GM analysis algorithms already successfully used in a variety of applications (e.g., in [5] for BP configuration).

To summarize, we presented an approach for analyzing changes to BPAs to allow organizations to be better aligned with their business domain dynamics and their desired level of flexibility. We introduced four dimensions for BPA design and used GMs to analyze BPA alternatives. Based on our experience in the transportation (discussed here), internet retail, and automotive domains, we found the approach useful for identifying/analyzing feasible BPA alternatives and as the basis for changing BPAs.

We believe that IS research needs to start focusing on adaptability, flexibility, etc. in addition to execution, correctness, etc. Consequently, our motivation is to address change through identification, analysis, and management of alternative BPAs, whereas most existing works (e.g., [2][3]) coordinate across multiple BPs, but do not investigate how PEs are distributed/allocated among them. We previously outlined the vision for the proposed framework [15]. Here, we elaborated on a portion of that vision. An extended version of this paper is available [7]. We have identified a research program to enhance our approach with the aim of addressing the above challenges. To evaluate the proposal, we are applying it to domains facing both a high rate of change and changes in the domain dynamics to further validate the expressiveness and usability of the notation and the analytical capabilities of the proposal.

References

[1] V. Chakravarthy and E. Eide. Binding Time Flexibility for Managing Variability, In Proc. the OOPLSA 2005 Workshop on Managing Variabilities Consistently in Design and Code (MVCDC 2), 2006.

[2] R. Dijkman, I. Vanderfeesten, and H. Reijers. Business process architectures: overview, comparison and framework. Enterprise Information Systems, DOI: 10.1080/17517575.2014.928951.

[3] M. Dumas, M. La Rosa, J. Mendling and H. Reijers. Fundamentals of Business Process Management, Ch.2. Springer-Verlag, Berlin-Heidelberg, 2013.

[4] M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou. Variability in Software Systems – A Systematic Literature Review. IEEE TSE, 40(3), pp. 282–306, 2014.

[5] A. Lapouchnian, Y. Yu and J. Mylopoulos. Requirements-Driven Design and Configuration Management of Business Processes. In Proc. BPM 2007, Brisbane, Australia, Sep 24-28, 2007.

[6] A. Lapouchnian, E. Yu, S. Deng. Responding to Ongoing Change – Challenges for Information Systems Modeling. International Journal of Information System Modeling and Design (IJISMD), 5(4), 2014.

[7] A. Lapouchnian, E. Yu, A. Sturm. Re-Designing Process Architectures. Technical Report CSRG-625, University of Toronto, 2015. Available at: ftp://ftp.cs.toronto.edu/csrg-technical-reports/625

[8] Open Group, The. ArchiMate 2.1 Specification, 2013. Retrieved from http://pubs.opengroup.org/architecture/archimate2-doc/

[9] J. Pagh, and M. Cooper. Supply Chain Postponement and Speculation Strategies: How to Choose the Right Strategy. Journal of Business Logistics, 19(2):13-33. 1998.

[10] M. Svahnberg, J. van Gurp and J. Bosch. A taxonomy of variability realization techniques: Research Articles Software – Practice & Experience, v.35 n.8, p.705-754, July 2005.

[11] S. Subramaniam, et al. Improving process models by discovering decision points. Information Systems, 32(7):1037–1055, 2007.

[12] E. Santos, J. Pimentel, T. Pereira, K. Oliveira, J. Castro. Business Process Configuration with NFRs and Context-Awareness. ER@BR, 2013.

[13] TOGAF Version 9.1. 2011. Retrieved from: http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[14] B. Weber, M. Reichert, and S. Rinderle-Ma. Change Patterns and Change Support Features – Enhancing Flexibility in Process-Aware Information Systems. Data and Knowledge Eng. 66(3):438–466, 2008.

[15] E. Yu and A. Lapouchnian. Architecting the Enterprise to Leverage a Confluence of Emerging Technologies. In Proc. ACET 2013 at CASCON 2013, Toronto, Canada, 2013.