Re-designing Process Architectures

CSRG Technical Report 625
Department of Computer Science, University of Toronto

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 for their day-to-day functioning and for longer term viability and sustainability. Together, these processes constitute the business process architecture (BPA) of the organization. While extensive efforts have been devoted to the analysis and design of business processes and associated information systems, there has been relatively little attention paid to the design of BPAs, i.e., how the processes of an organization should best relate to each other. As organizations are experiencing enormous changes, brought on by disruptive technologies and continual business model innovation, they can no longer optimize individual processes in isolation from each other. Their BPAs can no longer remain static, but need to be rethought from time to time. BPA design involves making trade-offs across multiple processes, particularly regarding how to balance flexibility and agility with other design objectives such as costs and efficiency. In this paper, we propose a framework for supporting the design of BPAs, by identifying several dimensions along which process elements (activities or decisions) could potentially be repositioned across processes. We illustrate the approach using the domain of transportation.**

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

I. INTRODUCTION

Organizations rely on a multiplicity of processes for their day-to-day functioning as well as for longer term viability and sustainability. These processes come together to constitute the *process architecture* of the organization.

Extensive efforts have been devoted to the analysis and design of individual business processes and associated information systems. Leveraging process models, information technology systems have produced tremendous gains in efficiency and productivity by automating or redesigning business processes (BPs). In contrast, relatively little attention has been paid to the design of process architectures, i.e., how the many processes of an organization should *best relate to each other* – the way these processes come together to serve the overall objectives of the organization.

As organizations are experiencing enormous changes, brought on by disruptive technologies and continual business model innovation, it is no longer adequate to optimize individual processes in isolation from each other. BPAs can no longer remain static, but need to be rethought and re-engineered from time to time.

Process architectures need to be rethought because traditional approaches to innovation and new capability development are being challenged. In highly dynamic organizations, new capabilities are constantly being created. Innovation cycles are being shortened, with development processes becoming intertwined with operational processes, as in the movement towards "DevOps" [16] in software. Mobile apps must be created and updated continually to keep pace with changing expectations and competitive pressures. Users and their usage processes are coupled with development processes in a virtuous cycle of value co-creation.

Similarly, the conventional boundary between planning processes and processes that execute those plans are being redrawn. On the one hand, a more dynamic and fluid world with high uncertainty invalidates many traditional planning approaches. On the other hand, the availability of massive amounts of data from numerous sources such as social media, mobile devices, sensor networks and devices together with powerful analytics enable much greater context-awareness and near-real-time response, and even pre-emptive action based on predictive capabilities. Decisions that used to be made during planning are often moved to the execution stage. Conversely, there can be benefits for moving an activity "upstream" to an earlier execution or planning stage.

Today's fast moving organization therefore cannot take an existing process architecture for granted. The process architect should be asking questions such as:

- Should some activities or decisions (process step) be deferred closer to frontline, to take advantage of near-real-time data, to meet customer needs and wants better?
- Should some activities or decisions that are currently performed in a planning stage be moved to an execution stage?
- Should activities that are previously performed for an aggregate be performed per instance?

BPA design involves making trade-offs across processes, particularly regarding how to balance flexibility and agility with other design objectives such as costs, efficiency, and so on.

Existing conceptions or treatments of process architecture (e.g., [4] (Ch. 2), EA frameworks [20], enterprise modeling

languages – UEML [21], EO [2]) typically treat the process architecture as a given, or as something to be discovered. There is no conscious effort to shape or design the architecture.

We believe there is need to analyze architectural alternatives, and provide support for exploring the space of alternatives, and guidance and support to choosing among them, recognizing the complex trade-offs that may exist. Need a conceptual model as a foundation. This paper proposes a number of dimensions including temporal, recurrence, plan/execution, and design use, which could serve as key elements of such a framework.

This paper is organized as follows. In Section II, we introduce the business domain in which we demonstrate our approach. In Section III, we motivate and outline the architectural design space, whereas in Section IV, we discuss the four dimensions that comprise this space. In Section V, we discuss the analysis needed to arrive at the right business process architecture, while Section VI discusses the related work. Section VII discusses challenges and outlines future research directions. In Section VIII, we conclude.

II. The Example Setting

A tsunami of change is sweeping through almost every business and industry sector today. To respond to these large-scale changes, organizations are adjusting or rearranging their process architectures, albeit mostly in an ad hoc fashion, in the absence of systematic frameworks or methods. For example, many organizations are making their processes more agile, by bringing their development processes closer to the user. Social media and other data analytics are used to shorten new product development cycles. In retail, marketing decisions can be made more frequently with better data about customer behaviour. In finance, mobile payment options are shaking up many related industries.

In this paper, we chose the domain of passenger transportation to illustrate our approach. The rationale for selecting this domain is that it is a common-sense domain that most people can relate to. It is rich enough to illustrate the features of our approach and patterns identified here can easily be found in other business domains.

While having a long and rich history, this domain still continues to experience change today, driven by business and technical innovations and the shifting priorities of governments and the public. Among such driving forces in this domain are the sustained focus on reducing energy consumption and emissions, traffic congestion, the increasing costs of owning cars and therefore the growing popularity of car sharing, public transit, bicycles, and bike sharing. Moreover, the proliferation of location sensors and the availability of up-to-the-minute traffic information can greatly improve service quality and flexibility. Smartcard or smartphone-based electronic payment methods improve the convenience for the service providers as well as for the passengers and add flexibility to support innovative variable payment methods that can take into consideration travel distance, travel date/time, frequency, etc. and guarantee savings for passengers while incentivizing them to use the service more. The recent introduction of new disruptive personal transportation services that deliver point-to-point service coupled with mobile application-supported ordering and billing for service consumers while allowing people to use their private cars to deliver these services is an innovation that forces many traditional players in the industry to rethink their business models to remain competitive. Self-driving cars may soon disrupt transportation even more. Public transportation is one subarea within the domain that, despite being quite heavily regulated, is an interesting case study for our approach. We discuss this domain throughout the paper and present a version of the BPA for that domain in Fig. 5.

III. A Design Space for Process Architectures

To work towards a design framework for BPA design, a fundamental question is: what is the space of possible alternative architectural designs? We need a modeling notation to be able to express different design configurations of processes, so that we can reason about the pros and cons of alternative designs. In this paper, we consider BPA design for an existing organization, where there is an existing process architecture. Alterative designs are therefore different ways of modifying the architecture.

We consider potential modifications as movements along four dimensions:

(1) The temporal dimension – moving a process element earlier or later in relation to other process elements. A *process element* (PE) may be an activity that produces some output or outcome. It may also include the act of making decisions. For example, in the transportation domain, should we have the payment placed before or after riding the bus?

(2) The recurrence dimension – positioning a PE in a process (or process segment) that is repeated more frequently or less frequently with respect to other PEs. For example, in the transportation domain, should we plan the bus route and schedule jointly, or we can determine that route and change the schedule more frequently.

(3) The plan/execution dimension – positioning a PE on the planning side of a process versus on the execution side, i.e., whether the activity specified by the PE is done during planning or during the execution of the plan resulting from the planning. For example, in the transportation domain, should we plan the bus route for every ride or can we have it planned and just execute the plan in each ride.

(4) The design/use dimension – positioning a PE on the design side or the usage side of a process, i.e., whether the PE is invoked as part of a design process, or is invoked during the usage of that artifact, tool, or capability that is the outcome of the design. For example, in the transportation domain, should we design a vehicle for each route, or we can use that design for each of the routes.

In considering the positioning of process elements along these four dimensions, we aim to address one of the crucial concerns of BPA design – the tension between flexibility and efficiency. For an organization operating in a completely static environment, all of its processes can be tightly coupled and globally optimized for maximum efficiency once and for all. For a dynamic environment however, the architecture needs to provide appropriate flexibility at the right places in order for

the organization to respond effectively to circumstances and uncertainties.

A fundamental idea for accommodating uncertainty is to keep options open. The ability to take alternate courses of action contingent on the actual circumstances at hand as they unfold allows for a high degree of flexibility. On the other hand, keeping options open could incur considerable costs as extra resources and capabilities need to be at the ready even if they are not deployed. Determining where and when options should be kept open is therefore a central mechanism for process architecture design. A model of the architectural design space hinges on a representation of where and what kinds of options (or choices) exist, and the conditions under which the options should no longer be held open, i.e., that a choice or decision be made and becomes committed.

We use the term *variation point* (VP) to refer to the point in a process where multiple options exist. Variation points may appear anywhere in a process. Together with activities, they constitute Process Elements. We use the term *variant* to refer to the individual options at a VP. We say that a VP is *bound* when one of its variants is selected. When and where a variation point becomes bound is the basis for much of the reasoning behind the positioning of a PE along the four dimensions. We discuss the relationship of this work to the software variability literature in a later section.

Note that although we refer to modification, the suggested dimensions and framework can be utilized for designing new business process architecture.

These dimensions were determined based on existing studies (see Section VI), our own experience in BPA, and the analysis of existing BPAs and their potential changes. The purpose of the set of dimensions is to expand the space of alternatives for business process architecture. By no means are we claiming that these are the only possible dimensions. Nevertheless, we found these suitable for characterizing the architecture design space.

In the next section, each of these dimensions will be explained in detail and illustrated.

As in the area of software systems architecture, the architectural description we are aiming for here should outline the major elements and relationships while avoiding over-specification. A process architecture, while specifying the crucial relationships among processes, should not impose unnecessary constraints on the detailed specification of the individual processes. The process architecture description should be sufficiently expressive so as to allow relevant architectural properties to be analyzed (e.g., how flexible one options is compared to an alternative). Therefore, the architectural description will likely need to refer only to certain selected elements from process specifications, and not the complete sequence of steps, control flows, data flows, etc.

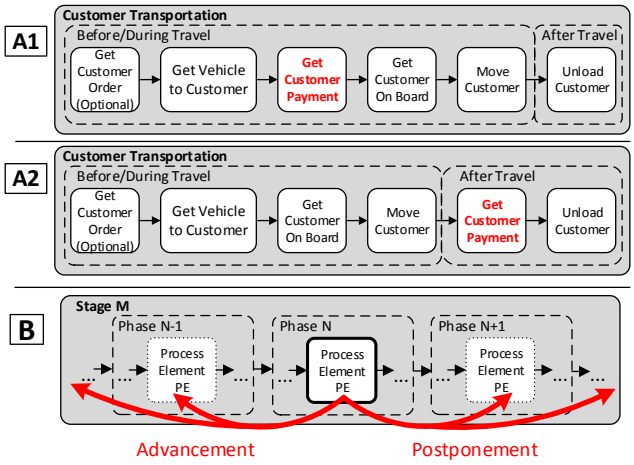

Fig. 1. Trip payment options: (A1) Before trip's end. (A2) At trip's end. Options for moving elements along the temporal dimension (B).

For the purpose of illustration, in this paper, we adopt a notation which is adapted from a combination of BP modeling [8] and function modeling [13].

IV. Dimensions for Process Architecture

In this section, we describe the four dimensions along which process elements can be (re)positioned within a BPA.

A. The Temporal Dimension

In many cases, there are multiple possible placements for PEs within a BP/system specification that comply with the existing functional dependencies (e.g., to calculate order prices, the system needs to know the contents of customers' shopping carts), achieve the same functional objective, but are different in terms of their non-functional characteristics.

This is illustrated by the passenger payment options presented in Fig. 1A. For instance, unlike the standard fare that is charged before a trip, after a trip, it is possible to charge a differentiated fare based on the distance travelled. Both variants achieve the objective of obtaining customer payment, but are, of course, different in how much is paid, how fair or precise the charge is, etc. Therefore, there may be multiple options along the temporal dimension in BPs – somewhere from the process' start to its finish – where decisions/actions can be placed and these choices need to be resolved by looking at how each variant affects the quality criteria that the enterprise is interested in.

Introducing phases. What is better – to charge the customer before he boards a bus, during the trip or after travel? In fact, this evaluation depends on one's point of view. Looking at payment fairness, we want differentiated payments based on the actual distance travelled (this is how taxis and other similar services operate). Here, the system needs a richer context – the ability to sense that distance, which can only be done at the trip's end. This makes the payment on exit a better option. The other variants are the same in terms of payment fairness. We then identify *phases* – portions of a process such that placing a PE under consideration anywhere within a phase produces the same result (e.g., charging passengers before boarding or upon boarding is no different since the trip distance is unknown in both cases). However, moving PEs across *phase boundaries* may affect the quality of decisions and the outcome of actions. While most software variability approaches focus on the technologies that enable variability in systems, we further analyze when (in which phase) it is best to execute actions or make decisions. Regardless of the positioning of PEs along the temporal dimension, we note that unlike other dimensions, no re-

use happens: an output of a phase belonging to one process instance can only be used by the subsequent phases of the same process instance. Overall, the benefits of phases are: 1) the reduction of the number of possible PE placement alternatives and thus the decreased analysis effort and 2) the ability to focus on the important issues while abstracting over some of the lower-level BP modeling details.

Postponement. In this dimension, we are looking at choices to postpone decisions/actions by placing them in later phases or to advance them by placing the PEs into earlier phases. Fig. 1B is a generic illustration of the temporal dimension showing the potential movements for the process element PE currently located in Phase N. *Postponement* is a well-known business strategy (e.g., in supply chains [15]) that aims at minimizing risks and maximizing benefits by delaying certain activities/decisions that require precise, up-to-date information until the last possible moment. Therefore, the key idea about postponing process elements is the expectation that there will be better, more precise information available at some later point, which would allow for better, more context-sensitive outcomes. On the other hand, 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 (e.g., through predictive analytics). Additional concerns include the availability of the sensed data required for postponed process elements and the cost and other resources required to collect and analyze that data. For example, when processing differentiated fare payments at the end of trips, the system should have the infrastructure to actually measure the distance travelled by each passenger and thus requires investment of resources for its development and operation.

Similarly, business domain volatility has a significant impact on system/enterprise flexibility, with highly dynamic domains requiring more flexibility (and thus variability) to keep achieving system objectives in the presence of continuous change. For instance, moving a PE from one phase to the next may have virtually no improvement in flexibility and other NFRs (non-functional requirements) in stable domains (in fact, there, we may move a PE all the way from the earliest to the latest phase without seeing much of an impact), but may result in very significant benefits in highly dynamic, volatile domains. Note that the benefit of postponing a fare payment to a later phase depends on the sensing capability of the system. Also, the assumption here is that trip distances are widely different, thus requiring differentiated fares to be fair. If most trips are about the same (i.e., the domain is stable from this point of view), the benefits of the added flexibility will not outweigh the extra complexity and cost. Overall, postponement requires system flexibility and finer-grained sensing/analytics.

B. The Recurrence Dimension

While in the previous section we described the placement of PEs along the timeline, the assumption was that the decisions/actions are executed for every process instance. Here, we propose another variability dimension that focuses on reusing the outcomes of decisions and activities in multiple process instances. To put it another way, how often should certain decisions or actions be (re)executed and under what conditions?

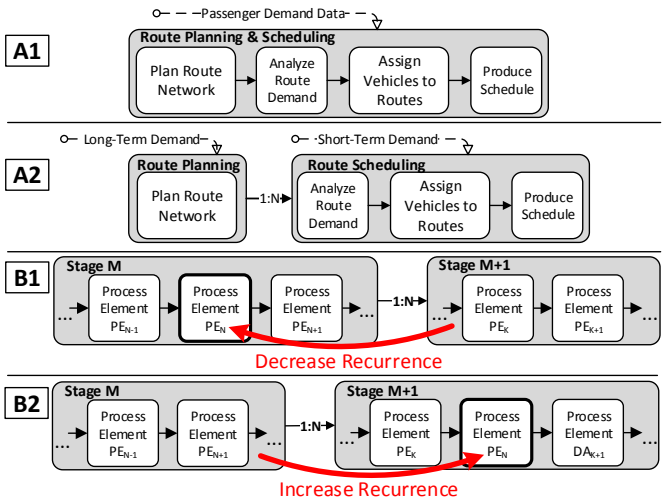

Fig. 2. Splitting a single stage (A1) into multiple ones (A2) to increase flexibility. Moving process elements across a stage boundary (B).

Definitions. For instance, when considering payment for trips, should a passenger pay every time he travels or can he buy a monthly pass and use it multiple times? We group PEs that have the same execution cycle (i.e., that are executed together) 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 using a control flow link (a solid line) labeled with "1:N" to indicate the cardinality of their relationship (see Fig. 2B). We say that there exist a *stage boundary* between a pair of stages connected in this way. This boundary 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 frequency with which a PE is executed, see Fig. 2B, which shows both directions of movement (note that a PE can be moved across multiple stages). In general, a stage represents a (sub-)process. Thus, while in the temporal dimension we focused on the intra-process analysis, here the focus is on inter-process relationships – relative execution/change cycle among processes.

Using the dimension. When coming up with a stage-based configuration, one can identify decisions and actions that can be reused for multiple process instances (i.e., are independent of the variations in those instances), at least for a period of time. Once these PEs are identified, they can be put into a stage to be reused multiple times by the subsequent stages, thus saving time, money and possibly other resources (e.g., buying a transit pass is a convenient option that removes the necessity of paying for every transit system use). Another heuristic for creating stages is to identify process PEs that need to be executed with the same frequency (e.g., yearly product redesign cycles accompanied by the revision of product manuals and marketing materials) or are triggered by the same data-driven trigger that fires, e.g., when a product redesign cycle is deemed necessary by market and customer sentiment analysis, when domain

changes are detected, etc. Certain stages are executed on-demand, being triggered by the appropriate events (e.g., for a taxi-like service, the Customer Transportation stage is triggered by customers requiring transportation services).

Splitting and merging stages. Changes in the stage execution frequency or in their triggers may create or eliminate stages. Thus, a special case of moving a decision/activity across a stage boundary is the split of a stage into two or a merge of two stages into one. For instance, in the former case, decisions/activities previously executed with the same frequency are split into two groups, each with its own recurrence pattern. This is shown in Fig 2A, further discussed below.

Domain example. To illustrate the choices that exist along the recurrence variability dimension, we look at how public transit route network planning and vehicle scheduling is done in one of our case studies. One way of doing this is to combine route planning and scheduling into a single stage (Fig. 2A1). There, whenever a route network needs to be redrawn (e.g., due to significant changes in demand), a Route Planning & Scheduling stage is triggered. The data input for the stage (the message flow arrow arriving at the top of that stage) contains all the passenger demand data the company has available. In this configuration, both route planning and scheduling are bundled together – they have the same change cycle, which means that changing schedules without a route network redesign is impossible. Clearly, in the case of the easily predictable constant demand, this configuration will work well. However, this rigidity will hurt the ability of a transit company to change its schedules more frequently in case of evolving passenger demand, which is the case in most cases. Frequently, one would want to be able to support seasonal schedule changes, to respond to sudden demand spikes due to public attractions, events, etc. while the routes remain the same. To address this issue, we can unbundle the route planning and scheduling as shown in Fig. 2A2 to create two stages, 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. This change to the transportation company's BPA allows the route network (the outcome of the Route Planning stage) be reused for multiple Route Scheduling instances (again note the "1:N" annotation), thus supporting frequent schedule updates reflecting changes in passenger demand. The new BPA configuration is more flexible, but likely incurs higher cost (e.g., the need to inform customers/employees of the changes), complexity and unpredictability.

Moving PEs among stages. Splitting/merging stages is one way to reconfiguring BPAs along the recurrence dimension. The other is to move PEs among stages, as illustrated in Fig. 2B. Given the recurrence dimension on BPA with a number of possible configurations, an organization needs to analyze them to determine which one best fits its needs based on the level of volatility in its business environment – e.g., balancing the cost, complexity, and the increased unpredictability of the unbundled configuration, and the inherent rigidity of the bundled one from the example of Fig. 2A. We discuss the modeling and analysis of BPA configuration choices in Section V.

The recurrence dimension and VP binding time. It is important to note that the recurrence dimension at the level of BPAs described above generalizes the binding time perspective on variability at the BP level, which focuses on whether VPs are bound at design time (static variability) or at runtime (dynamic variability). The traditional view is that static variability limits the systems' (or organizations') ability to change later. Runtime (dynamic) variability improves flexibility and adaptability, but also increases costs and complexity and decreases performance due to the need to implement multiple behaviours, the adaptation infrastructure, etc.

The problem with this view is that it is very inflexible, with just two extreme binding time options. To illustrate this, let us look at an example. In the transportation domain, from the operational standpoint, the decision to operate in a particular city/area (see Service Area Selection in Fig. 5) may seem like a design time, static decision. However, for a truly agile enterprise, it should be possible under certain circumstances to review and possibly change this decision to expand or contract its operations. In our approach, we view the above decision not as being bound at design time, but as its corresponding PE being positioned into a rarely executed stage. By supporting multiple stages with different execution frequencies, the recurrence dimension allows to periodically revisit previously made decisions, thus enabling finer-grained binding options for BP-level choices.

C. The Plan/Execute Dimension

In typical business process modeling (in the context of enterprise agility), the process model describes or prescribes the process that is to be executed, but not how this process gets determined. An important consideration for enterprise agility is whether a decision is made during planning, or is instead left to the execution stage. There are similar considerations for activities – whether they are part of a produced plan or not. In order to support reasoning about the possible placement of a PE on either side of a plan/execute boundary, our modeling framework allows for the explicit representation of planning/specification activities as well as the execution of the planned activities.

In many cases, a stage does not simply produce a result for the subsequent stage to use, but generates a plan or a specification to be executed by it. A plan produced by a stage either fully specifies or constrains the behaviour of the subsequent stages. While a plan 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 (e.g., see Maintenance Plan in Fig. 5), in most cases plans are reused. Therefore, plan generation implies the presence of stages and a stage boundary. We call the stage where the plan is produced the *planning stage*, while the subsequent stage is called the *execution stage*. 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. Planning and execution stages are relative to each other. An execution stage B with respect to some planning stage A can be a planning stage with respect to some execution stage C, in which case A is used to generate the planning procedure for B.

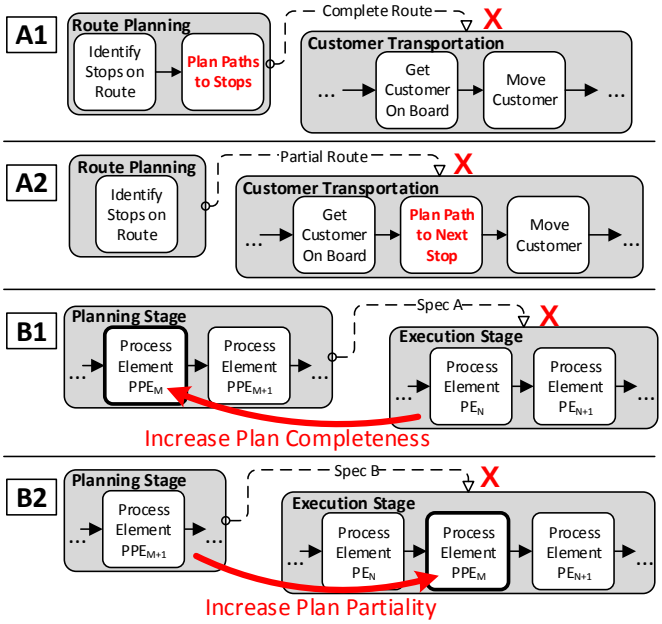

Fig. 3. Complete (A1) or partial (A2) planning choices for route planning. Moving process elements across a plan/execute boundary (B).

Full and partial plans. Within planning stages, we allow for a range of planning possibilities: from complete (full) to partial plans. Full plans completely specify execution in advance. They are quite restrictive and inflexible, but require no further deliberation within the execution stage, which lowers demands on that stage, ensures uniformity and predictability, and allows for high degree of optimization at the aggregate level. E.g., in Fig. 3A, the (bus) route planning stage determines both the stops along the transit route as well as the precise paths between them. When a driver executes the plan, he just has to follow the route and there is no additional planning to be done. This certainly simplifies the job of the driver, but also makes it impossible to make local adjustments to avoid problems on the road. Graphically we link planning stages to execution stages with data flow links annotated with X (for "eXecution") and showing what specifications are being sent to be executed by the execution stages (e.g., Complete Route in Fig. 3A1). The flow enters the execution stage from the top as per convention that we borrowed from the IDEF0 notation [13].

Unlike complete plans, partial plans allow for separation of things that are more stable from those that are more dynamic. Here, the output of a planning stage is a partial plan/specification that allows for a range of behaviours within the execution stage by leaving some previously preplanned choices open by moving them from a planning stage to an execution one. E.g., in Fig. 3A2, a portion of route planning, namely the planning of paths among stops is moved across the plan/execute boundary to Customer Transportation stage. This allows the driver to identify a path to the next stop based on the current context (traffic and weather conditions), thus increasing the system's ability to deal with largely unpredictable changes. At the same time, the driver's job is harder and there is the need monitor and analyze road conditions to calculate paths to next stops.

The concept of a partial plan covers such notions as business rules and constraint sets. The relationship between a controller and its target process where the former manipulates the parameters of the latter can also be seen as the plan/execute relationship, with the output of the planning stage indicating the bindings of control parameters in the execution stage.

Variability in Plan/Execute dimension. The general pattern for crossing the plan/execute boundary in both directions is shown in Fig. 3B (there, the specification Spec A is more restrictive than Spec B). As already mentioned, decreasing plan completeness increases flexibility and ability to handle change when executing the plan. It allows to separate stable and volatile portions of specifications. At the same time, 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. To summarize, in terms of variability that exists in this dimension, the main focus is on analyzing how much is to pre-plan in the planning stage and how much is to leave to the execution stage to achieve the desired level of flexibility in an organization.

It should be clear from the discussion above that in the Plan/Execute dimension 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 sustainability of systems in highly volatile domains.

D. The Design/Use Dimension

The power of technology relies crucially on the creation of enduring capabilities that can be exploited by a user who does not know how the capability is constructed. In typical process and enterprise models, tools, designs and other capabilities can be represented as modeling artifacts and utilized in various (e.g., BP) models. These capabilities are static in a sense that they are assumed to have been externally developed and therefore cannot be modified. On the other hand, to support enduring enterprises and IT systems, one needs to be able to represent those artifacts as evolvable objects that can be periodically redesigned to accommodate changes in the external environments and in business or system requirements. Representations of design, development or other tool/skill/etc. acquisition processes need to be integrated into enterprise architectures to allow for modeling of evolving capabilities available to the enterprise and thus to support continuous design [7][11]. 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 evaluations of cost-effective ways to remove those. Moreover, not unlike the previously discussed plan/execute dimension, the design/use variability dimension supports the identification and analysis of tool/capability design variations from the point of view of their flexibility.

Definitions. In the design/use dimension, designs, tools or other capabilities are produced by the *design stage* and are used within the *use stage* (see Fig. 4A). A design/use boundary is a stage boundary since a tool, once designed, can be used

multiple times, by multiple instances of subsequent stages. To support non-technical capabilities, we allow skills to be acquired as capabilities through the appropriate learning rather than development processes. Graphically, design stages are linked to use stages through data flows annotated with U (for "use"), which enter the latter at the bottom following the Mechanism arrows from IDEF0. The exact circumstances of the tool usage are not specified, thus leaving the use stage the freedom to use the tool as appropriate. Having a tool simplifies the achievement of business/system objectives in use stage. Moreover, the actor (human or artificial) using the tool does not have to know its internals. An example of such a tool is an automatic transmission, which can be used by drivers in place of a manual one while requiring no knowledge of how it actually shifts gears. Even though it is less fuel-efficient and less adaptable to the various road conditions and driving styles than a manual transmission, an automatic gearbox is simpler to operate and thus has fewer demands on the use stage.

Flexibility in Design/Use dimension. When looking at possible process architecture configurations along this variability dimension, our focus is on the flexibility of the tool/capability being produced in the design stage – i.e., how generic or single-purpose it is. The more single-purpose (less flexible) the tool is, the simpler it is to use and the more optimized it can be – for the general case, not for its actual use setting as the tool cannot be customized to take advantage of this contextual information.

Example. Fig. 4A shows how the design/use boundary can be crossed in both directions. Fig. 4A1 illustrates what it means to move a process element from a use phase to its design phase. In case of an activity this means that the tool takes on more functionality, thus increasing the level of automation in the use stage. In case of a decision (BP-level VP), it means that it is bound in the design stage and becomes fixed in the use stage, thus reducing the customizability of the produced tool (Design A). On the contrary, moving an activity in the other direction (Fig. 4A2) 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. In the public transportation domain, an example illustrating the above moves is the design/use of subway trains. If a decision on the length of trains (in terms of the number of cars) is made in the design stage, then the trains cannot be customized to reduce their size when the demand is low (e.g., on weekends). When the decision on train length is left outside of the design stage, the trains are more customizable and can be adjusted for each use instance. Thus, tool customization is done in the use stage by making decisions that are left unbound in the design stage.

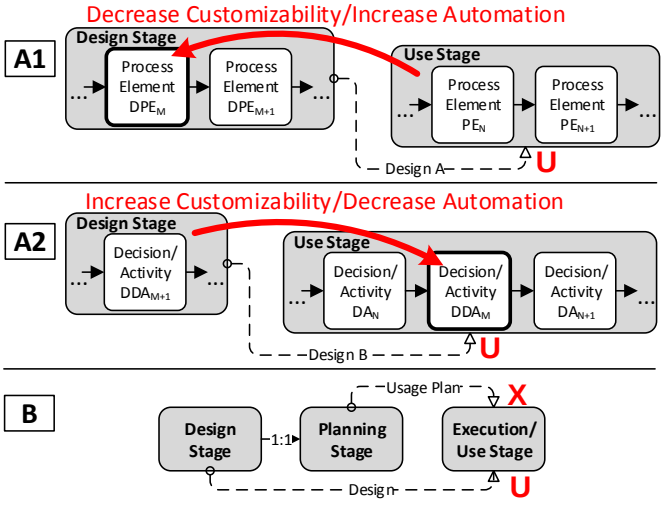

Fig. 4. Moving process elements across a design/use boundary (A). Use stage evolution driven by changes in tool design (B).

In addition to flexibility/customizability, when analyzing choices in the design/use dimension, we pay close attention to a number of other factors, chief among them is the cost of the tool/capability. The cost is incurred either through the design/development process or through tool acquisition (e.g., of a COTS system). Development of skills, obviously, has costs as well. Renting or leasing tools is a way to avoid the high upfront costs. Similarly, cloud-based IT services can be used to make tool/capability acquisition affordable. Moreover, with cloud-based services (or with any service-based capabilities), tool modifications (e.g., in terms of its power, processing ability, scalability, etc.) are easy and do not require redevelopment.

Evolving Capabilities. As discussed earlier in this section, the key reason for introducing the design/use boundary is to gain the ability to model and analyze changes in capabilities in continuously evolving systems. The assumption is that the design stage will not be executed just once to produce a tool or a capability. Rather, 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 stage can be re-executed when appropriate. This will produce/acquire new versions of the capability, thus evolving the enterprise and/or its systems. Note that frequently the use stages will need to change in concert with design stage re-executions to account for tool/capability evolution. This could be accommodated by the configuration shown in Fig. 4B.

To illustrate changes in capabilities, in Fig. 5, which shows an overall to-be process architecture for the public transportation domain, Vehicle Acquisition (a design stage w.r.t. Customer Transportation) gets the information on the required capacity from the Route Planning stage as its input. Based on this, it produces vehicle specifications and proceeds to acquire those vehicles. When demand for transportation services increases to the point where new vehicles become needed, Vehicle Acquisition will be triggered to procure those, therefore changing the transit company's capability to transport passengers to match changes in demand.

E. Process Architecture Model with Relationships from Four Dimensions

Now that we have discussed the four dimensions, we present a complete BPA in terms of interconnected stages, which can be viewed as separate processes within the architecture (Fig. 5). This is the overall BPA for a public transit company, which is innovative in its use of analytics and monitoring (including real-time traffic/weather and demand data), wants to be competitive on price and quality of service with other services and flexibly meet varying passenger demand. The BPA im-

plements the more flexible variants from the examples discussed in Section IV aiming to support agile transportation services. The model emphasizes inter-process relationships and while we capture PEs in stages, we abstract from modeling phases. The model captures how stages relate to each other using the relationship types introduced in this paper as well as the data supplied to and exchanged among the stages. Given a stage, its data input can arrive from another stage (e.g., Service Price) or can be obtained from the outside through monitoring or by other means (e.g., Competitor Pricing). The model gives an overall view on the organization and its BPs, shows where reuse happens, where plans are created and how they are executed and how tools/capabilities are utilized. At the same time, the model provides the basis for what-if analysis, with potential configurations generated by moving PEs among stages and/or phases.

V. Analyzing Process Architecture Alternatives

In the previous section, we presented the four dimensions of variability that help us identify and structure choices in BPAs. We also hinted at how choices along the four dimensions can be evaluated to help with the selection of alternative architectures that can handle change while attaining important quality objectives. In this section, we further elaborate on the modeling and analysis of BPA configuration options and provide example scenarios.

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

In our approach, we use goal models to represent and analyze choices in BPA along the four variability dimensions. Goal models are heavily used in Requirements Engineering to capture stakeholder and system objectives. Their key features are the ability to represent variability in achieving goals using OR decompositions (exclusive ORs in our case) and to use explicitly modeled NFRs to analyze goal refinement choices. Here, we adapted goal models to represent possible placements of PEs within the BPA along any of the four variability dimensions together with the evaluations of those choices against the relevant NFRs. A goal model focuses on analyzing the placement of a PE within a BPA, so multiple such models will be used to come up with a complete architecture (such as the one in Fig. 5).

Identifying BPA choices. To illustrate the development of such a goal model and its use for placing a PE along one of the dimensions, we look at the transit trip payment options discussed in Section IV.A. The goal model provides the intentional (as opposed to the operational) perspective and captures the objective for the PE in question as the root goal (see the top Get Customer Payment node in Fig. 6A) and shows how it is refined when being achieved in different stages/phases of the BPA. For example, Fig. 6A shows three alternative phases of Customer Transportation stage where Get Customer Payment PE can be placed. We use the @P:PhaseName and @S:StageName annotations to indicate where in the BPA PEs are to be placed. Each decomposition explores alternative PE placements within a BPA along one of the four variability dimensions. Thus, we annotate them with [T] for temporal, [R] for recurrence, [D] for Design/Use, and [P] for Plan/Execute, as shown in Fig. 6.

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 while respecting the existing constraints among the PEs – e.g., a PE must be placed before the PEs that rely on

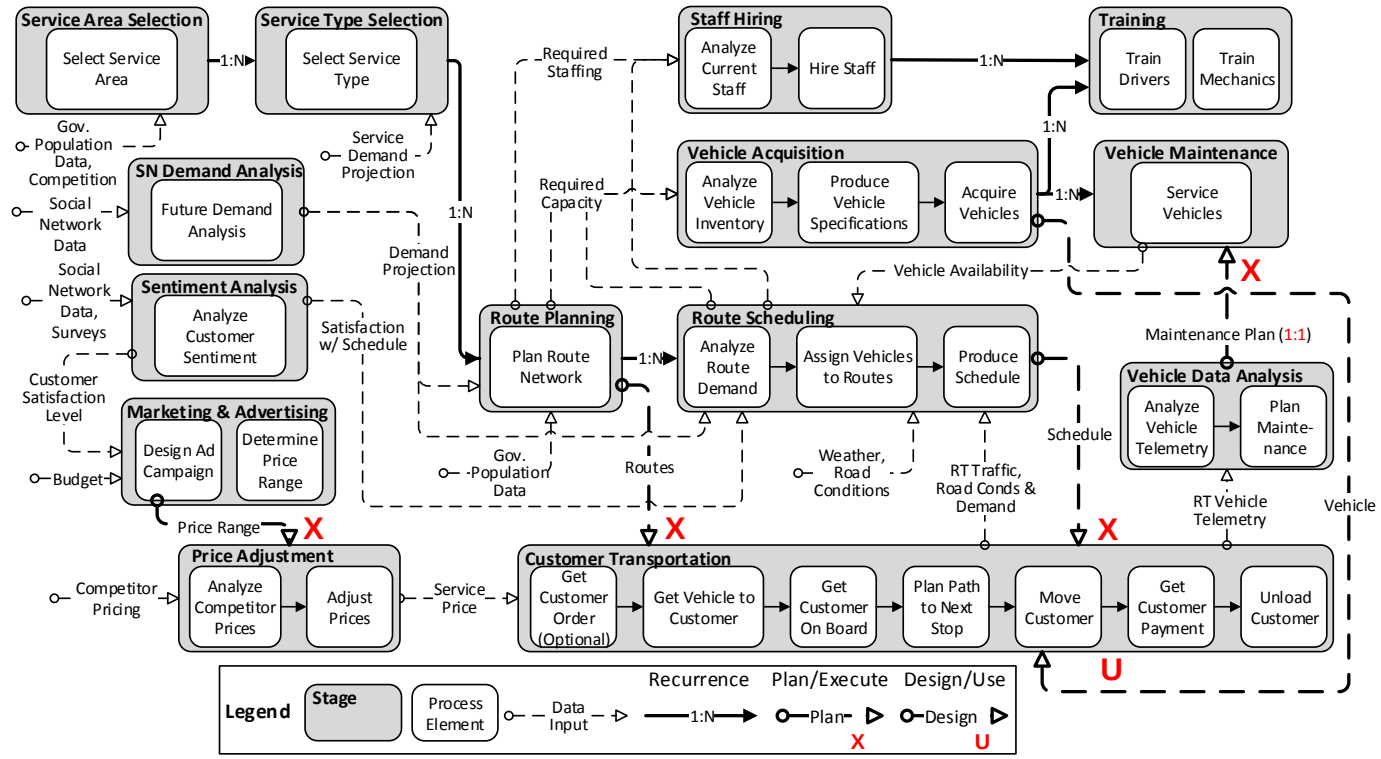

Fig. 5. The business process architecture for a public transit company.

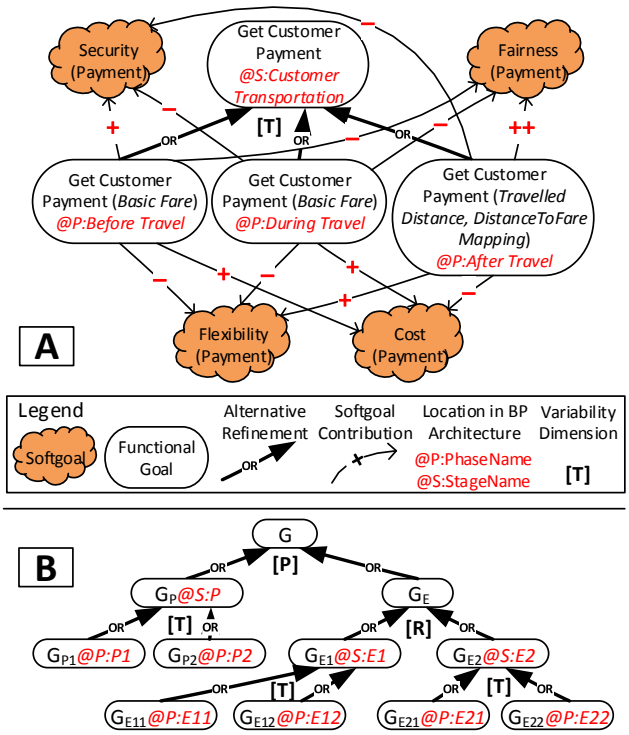

Fig. 6. Analyzing the temporal placement of customer payment PE (A). Generic goal model for positioning a PE with the objective G along the Plan/Execute dimension (goal parameters are omitted) (B).

its output. The temporal dimension focuses on the phases where the PE can be placed. Similarly, constraints must be respected. In Fig. 6A, customers can pay before (@P:Before Travel), during or after travel. Note that before deliberating on which phase to place a PE into, we need to determine its stage. Thus, we have frequent patterns of BPA-level analysis, such as [R] followed by [T]. In Fig. 6A, we do not show the [R]-type analysis, but indicate that the root goal is to be placed in Customer Transportation stage.

For the Plan/Execute and Design/Use dimensions, the choice is binary – whether the PE is inside or outside of some particular planning or design stage. Here, once a plan or a design is produced, it is available to all the subsequent stages. Thus, if a PE is to be placed on the "execute" or "use" side of the plan-execute or design-use boundary respectively, we must further deliberate on which stage is the best for the PE, and furthermore, which phase within that stage to put it in. This is illustrated in Fig. 6B, which shows the model for analyzing three different dimensions for the Plan/Execute case.

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

In our approach, we separate the "essence" of a PE (modeled as the root goal in Fig. 6A) – i.e., what, say, executing an activity helps us achieve – from its implementation (the subgoals in Fig. 6A) since that implementation can change from stage to stage and from phase to phase. E.g., the fare payment procedure, while conceptually achieving the same objective, may be quite different in various places of a BPA. Due to these variances, the parameters in goal nodes modeling alternative PE placements can vary to represent the different data these implementations operate on. E.g., for customer payments done before or during travel (Fig. 6A) the only information they can operate on is the basic fare. However, the variant executed after the trip operates in a context where the distance travelled is already known, so it can calculate a more precise fare based on that, thus achieving payment fairness (note the contribution link). Moving towards later stages/phases (i.e., increasing recurrence and postponement respectively) supports more contextual information, thus adding to the number of goal parameters in our models.

Capturing and analyzing quality objectives. As the last component of the model, we elicit quality objectives used for evaluating PE placement choices. These are modeled as softgoals and the evaluation is done with the help of contribution links with the help of *contribution links*. The evaluation can be qualitative, with a range of possible contribution values, such as help(+)/hurt(–), make(++)/break(--) (as in Fig. 6A). For example, getting payment after travel achieves (makes, ++) fairness, improves (+) flexibility, while negatively impacting (–) cost and security. Then, a softgoal is *satisficed* if there is sufficient positive and little negative evidence for this claim. Quantitative or finer-grained qualitative evaluations are possible depending on the available knowledge. Frequently, desired NFRs are conflicting and cannot be optimized for at the same time. Softgoal prioritization will help resolve these conflicts and arrive at sensible recommendations.

Handling trade-offs. As hinted in Section III, moving PEs along each of the four discussed dimensions affects a number of NFRs. The set of relevant NFRs is different for each dimension. Some of the NFRs are domain- and PE-independent. The most common NFRs of this type and the positive/negative effects of PE movement on them are shown in Table 1 and cost and flexibility are also modeled in Fig. 6A. It can be seen that in general, moving PEs to later stages and phases in the BPA as well as to use and execution stages improves flexibility, customizability, and context awareness of enterprises and their systems. These gains are offset by the increase in cost to maintain that flexibility (e.g., the flexible infrastructure), the decrease in stability (and predictability) and fewer opportunities for reuse. On the other hand, moving towards earlier stages/phases as well as increasing plan completeness and design automation improves reuse, stability, cost, etc., while negatively impacting flexibility and the ability to accommodate changes. As previously mentioned, these trade-offs need to be resolved based on two things: 1) the dynamics of the business

domain – what changes in the domain, how frequently, etc.; 2) the enterprise's prioritization of the above-mentioned quality criteria.

Additionally, there are domain-specific NFRs that are of relevance to the analysis of some particular PE. For fare payment, these are security and fairness (Fig. 6A). Note that in Fig. 6A, softgoals are parameterized with the particular area of focus, customer payments. Goal models allow for representing and analyzing how the satisficing value of, e.g., Security(Payment) affects the overall Security NFR. This can be used to estimate the impact of local BPA adaptations on the system/organization-wide quality objectives.

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

B. BP Architecture Adaptation Scenarios

Now that we have shown how one can analyze the possible movements of PEs within a BPA, we turn to the question of whether and when an architecture needs to change. The main objective of our approach for designing BP architectures is to help organizations determine the right amount of flexibility to support based on the volatility in their business domains as well as their own preferences and priorities. They need to find the balance between flexibility and stability/cost, etc. Once designed, a BPA has some amount of flexibility to support certain types of changes in the domain.

A simple example is the payment in passenger transportation. The already discussed after-trip payment option supports widely different trip distances without the need for a BPA reconfiguration. As long as the dynamics of the domain justifies that solution, i.e., the domain assumption that passengers generally take trips of quite different distances holds, that options will be adequate considering the added complexity and cost. However, if at one point the analysis shows that passengers mainly take trips of about the same distance (e.g., from a number of equidistant suburbs to downtown and back), the flexibility of the variable fare option becomes *too flexible* for the domain and its cost and complexity can no longer be justified. This situation will require an invocation of the BPA analysis process and a reconfiguration of the BPA.

Another example involves public transit route planning. When a long-term demand changes (e.g., due to a population shift), the BPA fragment in Fig. 2A2 is able to handle this change since by (re-)executing the Route Planning stage to add or alter routes. The assumption here is that these are rather rare changes informed by long-term demand. On the other hand, if a company wants to be able to frequently introduce special routes aimed at handling passenger demand due to special events (festivals, concerts, etc.), then the above assumption no longer holds. This case warrants a BPA reconfiguration to improve route planning flexibility – moving towards a configuration in Fig. 2A1, but with the added expectation that routes will be changing frequently.

In general, BPA reconfiguration takes place when domain dynamics changes – i.e., not when *some* change happens, but when the *rate of change* (or sometimes the range of change) becomes different.

VI. Related Work

"A business process architecture is a collection of business processes and their interdependencies with each other" [5]. That notion has been discussed for a while [4][5]. During the years various sets of relationship among business processes have been identified. For example, Dumas et al. [4] discuss the notion of a *sequence* in which the output of one process is used by another process, *hierarchy* in which the details of a process are further elaborated into sub-processes, *reference* in which a process is using (as an initial step) pre-existing process specification. The authors also provide guidelines of when to split processes, in particular in cases of difference in execution frequencies and time intervals, difference in location, and other such granularities. Eid-Sabbagh et al. [5] suggest another view over relationships among processes. These include "*composition*, which models one business process being composed of a number of other BPs, also called the sub-processes"; "*specialization*", which represents that one business process specializes another; "*trigger*, which represents that one business process causes another BP to instantiate and start"; and "*information flow*, which represents that information or other objects flow from one business process to another".

Lately, Dijkman et al. [3] summarize the existing relationships among BPs as follows: The *decomposition* relation expresses that a process is decomposed into multiple subprocesses; the *specialization* relation expresses that one process is a specialized version of another; the *trigger* relation expresses that the execution of one process can trigger the execution of another; and the *use* relation expresses that one process provides services that are used by another. In addition, the authors classified approaches for designing BPAs: goal-based, action-based, object-based, and function based. All approaches utilize a subset of these relationships when analyzing the business process architecture.

In the enterprise architecture area, the notion of BPA is referred as business process cooperation. For example, in ArchiMate [14] such cooperation includes the following aspects: *causal* relationships between business processes, *mapping* of business processes onto business functions, *realization* of services by business processes, and the *use of shared data*. These aspects can also imply on the type of relationships among BPs.

Another domain that business process architecture may benefit from is the area of BP variability, which has been studied for over two decades [1]. In this domain, the key concepts are the realization relationships among BPs and the binding time of such processes. In particular, this domain is of interest as it refers to alternatives within BPAs and the ways of deciding upon the most suited architecture. Similar situation occurs in software product lines. For example, Svahnberg et al. [17] state a major reason to support variability is to postpone concrete design decisions to the latest point that is economically feasible. In general, early binding facilitates better static analysis while late binding enables configuration by users and post-deployment updates [1]. This is also confirmed by Subramaniam et al. [18] who argue that positioning decision points as early as possible within the process timeline can improve pro-

cess efficiency by decreasing their uncertainties and identifying redundant activities. Other approaches examine variability from the requirements point of view and refer to intentional variability in BPs, e.g., [9][10]. In particular, these approaches looked at the variability in the way objectives can be achieved as the means to develop customizable, adaptive, and evolving systems. The existing approaches propose various ways to handle variability. However, they all focus on variability at the same level of abstraction.

Recently, various approaches that weave requirements and BPs have emerged. E.g., Santos et al. [19] recruit NFRs and contexts for the sake of configuring business processes. Usually, process configuration is done at design time, yet in that paper the authors suggest to have it tailored during runtime.

Our approach focuses on variability management and analysis at the level of BPAs. Each set of choices for placing PEs into stages/phases (such as the one in Fig. 6) can be seen as a VP at that level, and binding all such VPs along multiple dimensions produces a complete BPA. Here, we gain the ability to represent and analyze possible BPA configurations using variability and goal-based techniques. Also, if we consider PEs that are decisions themselves – i.e., VPs at the process level, then phases/stages where these VPs can be placed provide useful (meta-level) options on the binding of these BP-level VPs in terms of *when* (temporal dimension) and *how often* (recurrence dimension) they are to be bound. Thus, these phases/stages offer domain-specific options for binding VPs, which are much richer than "runtime" or "design time" that are usually discussed in variability research (e.g., [6]).

Within the domain of process-aware information system, Weber et al. [22] distinguish among four important dimensions in which change might occur. They use the notion of patterns for changes in predefined regions to define these dimensions and include (1) the late selection of process fragments, which refers to runtime binding of BPs; (2) the late modeling of BP fragments, which refers to modeling BPs in later stages; (3) the late composition of process fragments, which refers to the creation of ad-hoc BPs; and (4) the multi-instance activity which refers to the number of time an activity is executed at runtime. That work implies two new business process relationships: *creation*, in which a process may create another process; *recurrence*, in which a process may be followed by another process several times. While somewhat similar to our proposal, that approach only considers design time and runtime (compared to the flexibility offered by stages in our approach), mainly considers a specific business process, and neglects trade-offs (with respect to NFRs) among the various options.

In our own work, we have previously outlined an enterprise-level framework that allows organizations to utilize emergent technologies such as the cloud, big data analytics, etc. as well as feedback loops to engineer BPAs that would support flexibility and agility in the face of domain changes and shifting expectations [12][23]. In this paper, we elaborate on one portion of that proposal.

VII. Discussion and Future Work

In this section, we discuss some of the limitations of the approach, outline issues, and propose enhancements.

The four dimensions for BPA design can be viewed as supporting both adaptation and evolution of BPs and BPAs within enterprises. The temporal and recurrence dimensions are about identifying, analyzing, and implementing changes in BPA configurations supported by existing set of processes and enterprise/IT capabilities. They help determine which configuration from the space of BPA alternatives best matches the current business domain dynamics, thus helping enterprises adapt to changing circumstances. The next two dimensions go further and allow for modeling and analysis of options for evolving enterprises and their systems to accommodate more significant and unpredictable changes. This is done by enabling modifications of behaviours represented through changing plans or specifications (the plan/execute dimension) and through supporting modifications of capabilities (designs, tools, skills, etc.) used to achieve business and system objectives (the design/use dimension).

A good BPA must reflect the properties of the domain, especially those related to the changes within it. Assumptions about the fluctuating passenger demand, frequently changing weather/traffic conditions, etc. represent the expectations about the domain dynamics and need to be carefully analyzed to justify the architecture and analyze the flexibility that it affords for dealing with this volatility. While we briefly discuss the role of domain assumptions in Section V.B, they are not yet explicitly captured in our approach. We are currently working on formal modeling of such assumptions, which paves the way for the precise specification of conditions that trigger adaptations already supported by the current BPA (e.g., a change causing a stage re-execution) and those requiring BPA reconfiguration (e.g., a change in domain dynamics).

There is a lot to be said about integrating data into this approach. Monitoring and analyzing the external environment of an organization is important for creating an agile enterprise with flexible IT systems as we need to recognize context changes, failures, etc. Availability and volatility of data plays a crucial role in positioning PEs within a BPA. As we have already seen, the actual distance traveled by a passenger is only available after the end of the trip. Also, while some information can be reused by thousands operational process instances (e.g., the current sales tax), other data, such as the price of fuel, is more volatile and PEs that depend on it must be placed into more frequently executed stages. We are looking at enhancing the identification of data relevant for each PE and at capturing the effects of data availability and volatility on positioning the relevant PEs within a BP architecture.

While performing the analysis procedure described in Section V, it is easy to see that it favours local optimization (at the level of individual PE), possibly at the cost of globally optimal performance. Obviously, one needs to keep the overall configuration of the BPA in mind and evaluate the effects of a local change on its NFRs. Having acknowledged this, we are working on ways to mitigate this problem by, e.g., integrating multiple goal models of the type presented in Fig. 6.

In our approach, we allow for some model incompleteness to reduce the complexity of modeling and analysis and to focus on important aspects of the domain. For instance, the analysis of BPAs using our approach does not require goal models to be

constructed for *all* the PEs. There are certain portions of the BPA that can be considered stable given the dynamics of the business domain. We do not need to construct goal models and analyze BPA alternatives in the stable areas. Only the portions of a BPA that need to be more flexible to accommodate change in certain areas of the environment will be the subjects of our analysis. Similarly, many finer-grained BP modeling details are below our threshold of interest, which is a phase. Thus, these details do not feature in our models and in the analysis.

We view modeling and analysis complexity as an important issue for the approach. This is why (as stated above) we focus variability modeling and analysis on the areas of BPA where 1) change is expected and 2) where high level of flexibility is needed. This would limit the number of possible BPA configurations. To help with the goal-based analysis of alternatives, we plan to utilize both top-down and bottom-up goal model analysis algorithms that have been successfully used in a variety of domains and applications (e.g., in [9] for BP configuration).

In terms of evaluating the approach, we are applying it in several domains that exhibit not only a high rate of change, but also feature frequent changes in domain dynamics (i.e., the second-order change) with the aim of further validating the expressiveness and usability of the notation and the analysis capabilities of the proposal. Furthermore, we are planning to compare the expressivity and comprehensibility of our modeling notation with those used for representing BP and enterprise architectures, specifically focusing on both model construction and the use of models to identify/analyze architecture reconfiguration options.

VIII. Conclusion

In this paper, we presented an approach for specifying and analyzing business processes architectures to allow organization to be better aligned with the dynamics of their business domains and their desired level of flexibility. The approach introduces four dimensions, namely, temporal, recurrence, plan/execute, and design/use, along which the BPA is considered. We adopted goal-oriented analysis of architecture alternatives to identify the best BPA in the context of the organization. Based on our experience with the approach in the transportation (discussed here), internet retail, and automotive domains, we found the approach useful for generating feasible BPA alternatives, for analyzing the trade-offs among these, and as the basis for dynamically changing BPAs. We have identified a research program to further refine and evaluate the approach.

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] J. Dietz. Enterprise Ontology: Theory and Methodology. Springer, Berlin-Heidelberg, 2006.

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

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

[5] R. Eid-Sabbagh, R. Dijkman and M. Weske. Business process architecture: use and correctness. In Proc. 10th International Conference on Business Process Management (BPM'12), Springer-Verlag, Berlin-Heidelberg, 65-81, 2012.

[6] 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.

[7] R. Garud, S. Jain, and P. Tuertscher. Incomplete by Design and Designing for Incompleteness. In Design Requirements Engineering: A Ten-Year Perspective. Springer, Berlin-Heidelberg, pp. 137–156, 2009.

[8] A. Hallerbach, T. Bauer and M. Reichert. Capturing Variability in Business Process Models: the Provop Approach. J. of Soft. Maint. and Evol.: Research and Practice, 22(6-7), pp. 519–546, 2010.

[9] A. Lapouchnian, Y. Yu and J. Mylopoulos. Requirements-Driven Design and Configuration Management of Business Processes. In Proc. 5th International Conference on Business Process Management (BPM 2007), Brisbane, Australia, Sep 24-28, 2007.

[10] A. Lapouchnian, Y. Yu, S. Liaskos and J. Mylopoulos. Requirements-Driven Design of Autonomic Application Software. In Proc. 16th Annual International Conference on Computer Science and Software Engineering CASCON 2006, Toronto, Canada, Oct 16-19, 2006.

[11] 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.

[12] A. Lapouchnian and E. Yu. Exploiting Emergent Technologies to Create Systems that Meet Shifting Expectations. In Proc. Emergent Technologies Track at CASCON 2014, Toronto, Canada, 2014.

[13] NIST. Integration Definition for Function Modeling (IDEF0), 1993. Retrieved from http://www.idef.com/pdf/idef0.pdf

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

[15] 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.

[16] J. Roche. Adopting DevOps practices in quality assurance. Communications of the ACM, 56(11):38-43, 2013.

[17] 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

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

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

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

[21] F. Vernadat, F. UEML: Towards a Unified Enterprise Modelling Language. International Journal of Production Research, 40(17):4309-4321, 2002.

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

[23] E. Yu and A. Lapouchnian. Architecting the Enterprise to Leverage a Confluence of Emerging Technologies. In Proc. 1st International Workshop on Advancement from Confluence of Emerging Technologies (ACET 2013) at CASCON 2013, Toronto, Canada, 2013.