Exploiting Emergent Technologies to Create Systems that Meet Shifting Expectations

Alexei Lapouchnian and Eric Yu

University of Toronto, Canada

Abstract

Properly combined, today's emerging technologies can potentially lead to systems that adapt quickly and smoothly to ongoing shifts in user requirements and expectations through improving sensing and analytics and utilizing advanced software innovations and service orientation to support dynamic reconfigurations. To produce a system flexible enough to continually meet evolving expectations, various emergent technologies need to be assembled in a coherent fashion based on the capabilities and flexibilities they afford. We outline a framework in which the many activities and choices involved from design to execution and usage of a system can be re-positioned in relation to each other in order to achieve different kinds of flexibility and adaptiveness, taking advantage of data available from sensing mechanisms. An example from the transportation domain is used as an illustration.

1 Introduction

In past decades, information and other automation technologies have enabled enterprises to gain efficiency and productivity by orders of magnitude compared to manual operations. However these benefits were often achieved at the expense of flexibility and agility. Today's emerging technologies (such as social, mobile, and sensor networks, combined with data analytics) are giving organizations enormous power to sense and interpret the environment, and to decide what changes to make to their strategies, operations, and systems to meet rapidly shifting needs and expectations. Fortunately, many recent advances and emerging technologies (such as cloud, service orientation, and self-adaptive software) are also available to overcome the rigidities inherent in earlier architectures and systems. In this position paper, we argue that to better exploit today's emerging technologies (ETs), we need to be able to think about them using higher-level abstractions that are not tied to specific technologies, yet will convey their strategic significance. We outline some key elements of a framework for modeling and reasoning about flexibility and adaptiveness, which can be greatly enhanced by deploying today's ETs strategically in an enterprise architecture.

To illustrate the framework's concepts, we use the domain of public transportation since one can easily imagine various flexibility scenarios in everyday life contexts, without extensive knowledge of software capabilities and limitations (one can imagine analogous examples in providing more IT-intensive domains, such as e-commerce or financial services). The following high-level activities/decisions exist in the public transportation domain (ordered from strategic to operational): determining service location and type, obtaining various vehicles, hiring drivers, identifying routes, allocating vehicles to routes and determining their schedule (i.e., assigning capacity to routes), and transporting customers. In a typical public transit company, route planning/scheduling is done periodically (e.g., yearly/seasonally) to account for well-known demand changes (e.g., summer slumps, weekday/weekend differences) and gradual population shifts. Is this configuration the best the company can do? How can it promptly adapt to customer demand fluctuations or to changes in customer preferences and/or the competitive environment? How can ETs be utilized to make the company more nimble?

2 Exploiting ETs for Enterprise Adaptiveness

To abstract away from specific technologies and to focus on flexibility for enterprise adaptiveness, we build on the concept of variability. The concept is widely studied in the context of software product lines [1], BP modeling [3], and software engineering in general [2][5], and can be applied to enterprise architecture. A fundamental concept here is the *variation point* (VP), which is a place in models/specifications where variations occur. The options available at each VP are referred to as *variants*. A VP is *bound* when its variant is selected. The positioning of a VP within an overall development process and system architecture determines the nature of the flexibility possessed by the system. For example, making choices earlier (e.g., static, design time variability), restricts the variability and limits the systems' ability to change later. Runtime (dynamic) variability improves system flexibility/adaptability, but also increases costs and complexity and decreases performance due to the need to implement multiple behaviours, the adaptation infrastructure, etc. Thus, careful evaluation of costs and benefits of variability implementation options is important. Today, no modeling approach exists to help determine just how much variability is needed to achieve the organization's business objectives while balancing other important concerns.

Inspired by goal-oriented requirements engineering [4], we represent VPs as goals, and variants as alternative ways of achieving goals, together with criteria to guide their selection. We go beyond current variability modeling approaches by considering several perspectives on variability: (1) temporal; (2) recurrence; (3) plan/execute; (4) design/use. (The latter two are briefly outlined due to space limitations.) The four perspectives enable a finer-grained view of system variability/flexibility options and help better address changing business needs while taking into consideration the degree of volatility present in each business area. The perspectives provide an abstraction framework to analyze the impact and the strategic significance of ETs (and their combinations) on systems' flexibility and point to how ETs can be used coherently to realize more responsive and adaptive systems.

The Temporal Perspective. When ordering activities and decisions (let us call both of these *nodes*) in systems, we need to respect the functional dependencies among them. Still, there can be many possible node placements that respect the dependencies, achieve the same functional goals, but differ in terms of their non-functional characteristics. For instance, when paying for public transit, a passenger can pay before boarding a vehicle, when entering it, or upon exit. How do we evaluate these choices? What is better – to *postpone* or to *advance* the payment? 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 operate). Here, the system needs a richer context – the ability to sense that distance, which can only be done at the trip's end, and flexible payment options (both can be enabled by a combination of ETs, see Table 1), making 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 node under consideration anywhere within them produces the same result (e.g., "before/during trip" and "after trip"). Moving nodes among phases 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 (i.e., bind VPs).

The Recurrence Perspective. While the previous section focused on the placement of nodes along the timeline, the assumption was that the decisions/actions are executed for every process instance. We propose another perspective on variability that focuses on reusing the outcomes of decisions and activities for multiple instances. To put it another way, how often should decisions/actions be (re)executed and under what conditions? We group nodes that have the same execution cycle into *stages*. Once a stage executes, its output remains available to the downstream stages until it is re-executed. While the last stage is executed for every process instance, the re-execution of the other (earlier) stages can either be periodic (e.g., the bus schedule is changed yearly) or can be defined through data-driven triggers that fire when the downstream stage(s) cannot be adapted to continue achieving the system's objectives due to their limited flexibility, when KPIs are not met (e.g., negative customer sentiment identified through social network ana-

Emergent Technologies	New capabilities (Sensing, analytics, flexible behaviours)	Enabled Domain Action	The Affected Variability Perspective
Internet-Connected Devices + GPS Sensors + Analytics	RT traffic info + RT vehicle tracking	Predict on-time performance; Change routes and schedules in RT	Plan/Execute (+freq)
	Track vital vehicle performance data	Preventive vehicle fleet maintenance	Recurrence (+freq)
	Track customer location	Pick up customers at on-demand stops	Temporal (+postp)
	Track customer location, distance travelled	Flexible payments on mobile devices	Temporal (+postp)
Cloud and Service Orientation	Add transport. capacity cost-effectively	Assign flexible mix of vehicles to routes	Design/Use (+flex)
	Add transport. capacity cost-effectively	Flexible capacity planning	Plan/Execute (+freq)
	Add data proc. capacity cost-effectively	Fine-grained route planning/scheduling	Plan/Execute (+freq)
	Platform to sell services to companies	Offer excess transport. capacity as service	Design/Use (+flex)
Social Network Analytics	Customer sentiment analysis	Adjust service to handle customer issues	Recurrence (+freq)
	Project demand (concerts, events, etc.)	Change schedule to meet expected demand	Plan/Execute (+freq)

Table 1: Capabilities and variability in the transportation domain enabled by (combinations of) ETs. Notes on the abbreviations used: +freq – increased decision/action/plan frequency; +flex – increased system flexibility; +postp – postponed decision/action; RT – real-time.

lytics, an ET, see Table 1), when domain changes are detected, etc.

The Plan/Execute Perspective. 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 VP gets bound during planning, or is instead left to the execution stage. In order to support reasoning about the possible placement of a VP 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. A planning stage defines a space of behaviours and therefore the flexibility available for the corresponding execution stage. A key factor here is again the availability of sensed data, analytics, and system flexibility. E.g., when detailed customer demand, traffic, and weather information is available with the help of ETs (see Table 1), scheduling (and even route planning) can be done on demand, thus dynamically adjusting transportation services to the changes in the business domain. Also, route planning and vehicle scheduling can either be split into two (nested) planning stages reflecting their different dynamics or performed in a single planning stage if their dynamics are similar. Note that feedback control can be represented using plan/execute stages.

The Design/Use Perspective. 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. Analogous to the plan/execute boundary, a VP can be placed on either side of a design/use boundary. Illustrating rigid and flexible designs respectively, light-rail transit systems can be designed so that trains either have a fixed or a variable number of cars. The latter option allows the trains' capacity to be dynamically adjusted according to demand at usage-time (see Table 1).

3 Analyzing Alternative Placements of VPs

The objective of our framework is to give an organization the ability to design systems that have the "right" amount of flexibility for a given domain while taking into consideration other criteria such as cost, performance, uniformity, etc. Placing VPs in system specifications in relation to each other according to the four perspectives described above defines system flexibility. Moving nodes within each perspective affects a number of concerns. In our approach, we utilize goal models to capture alternative VP placements together with the evaluation quality criteria by constructing the appropriate meta-level VPs. These are not shown in the paper due to the lack of space.

For the temporal perspective, postponing decisions/actions allows the use of the up-to-the-minute information to support more context-aware and flexible behaviours. The downsides include, cost, complexity, unpredictability, etc.

The additional concerns here are the availability of sensed data and the domain volatility (with the highly dynamic domains requiring more flexibility to keep achieving system objectives in the presence of continuous change). Note that the benefit of postponing a fare payment to a later phase depends on the sensing capability of the

system. Similarly, 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, all of which are enabled by *combinations* of ETs. Also, alternative ETs can be analyzed by looking at how much postponement (in terms of phases) they can afford.

In the recurrence perspective, the choice is about which stage to place a given node in to achieve the right balance of reuse and flexibility. Pushing decisions/actions to later stages improves flexibility at the expense of the economy of reuse. The cost of adaptation (including that of sensing, data analysis, and system reconfiguration) is a key concern. Sensing, analytics, and system flexibility are required to cross stage boundaries and thus ET combinations again act as flexibility enablers. In our example, with a combination of ETs (see Table 1), the transit service schedule can go from being set at specified intervals to being adjusted frequently based on projected demand, current traffic/weather conditions, etc., thus crossing the stage boundary into a later stage and improving flexibility. However, executing stages more frequently than the rate of domain change that can be detected by the system's sensing/analytics capabilities (or faster than the changes are actually happening) will not produce tangible benefits.

For the plan/execute perspective, pushing planning to later stages improves flexibility and allows for the creation of better plans by utilizing data (e.g., the daily passenger demand predictions) not available in earlier (e.g., yearly) stages. Implementation/operation costs for planning and sensing/analytics capabilities need to be considered (which can be drastically reduced when using ETs). As illustrated in Section 2, planning can be split into multiple stages based on the different dynamics of the plans' portions.

The choices within the design/use perspective are about how flexible vs. single-purpose the acquired capabilities are (e.g., a bus is more flexible than a streetcar) as well as about the resources needed for their acquisition. ETs play a hugely important role in achieving design/use flexibility. E.g., the cloud and service orientation allow dynamic and cost-effective acquisition of new capabilities (both IT and domain-specific), thereby increasing system flexibility (see Table 1).

4 Conclusion and Ongoing Work

The outlined framework supports fine-grained reasoning about the variability and flexibility to be implemented in a system by looking at four novel variability perspectives. We discussed how pushing decisions, actions, and planning to later stages/phases and acquiring flexible capabilities requires that multiple possible behaviours be implemented within systems, and that fine-grained sensing (including from mobile devices and in social media) and advanced analytics be built-in to support agile context-driven adaptations. These are enabled by today's ETs (as illustrated in Table 1). Identifying where their systems need to be in terms of the four variability perspectives guides organizations in selecting the appropriate combinations of ETs to reach that objective.

In the ongoing work on the approach, we continue to enhance the representations for the four discussed variability perspectives and also focus on the challenges of goal-based meta-VP modeling and analysis, explicit modeling of barriers to change in existing systems, of ET capabilities, and of feedback, including capturing what data needs to be sensed and analyzed. At the same time, we are working on applying the approach in a number of business and IT domains.

References

[1] L. Chen, M.A. Babar, and N.Ali. Variability Management in Software Product Lines: a Systematic Review. In Proc. *SPLC*, pp. 81-90. Pittsburgh, USA, 2009.

[2] 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, Mar. 2014.

[3] 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.

[4] A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In Proc. *RE 2001*, pp. 249–262, 2001.

[5] M. Svahnberg, J. van Gurp, and J. Bosch. A Taxonomy of Variability Realization Techniques. *Softw. Pract. Exp.*, 35(8), pp. 705–754, 2005.