

Run-Time Variability for Context-Aware Smart Workflows

Aitor Murguzur and Salvador Trujillo, IK4-Ikerlan

Hong-Linh Truong and Schahram Dustdar,
Vienna University of Technology

Óscar Ortiz, Technical University of Madrid

Goiuria Sagardui, Mondragon University

// The proposed framework lets developers model and manage process variability by composing base models, fragments, and variability models and by deferring binding to run time. Base models and fragments are reusable, thereby reducing the modeling effort for developing variants. //



CURRENT MARKET DYNAMICS increasingly force companies to customize software products for their customers. Accordingly, product requirements are changing rapidly and continuously. Therefore, software developers need

to flexibly react to these changes and reduce design time and costs while maintaining the high quality and prompt delivery of software.

A well-established approach for complying with dynamic requirements

is software product lines (SPLs). An SPL is a set of software products that are closely related (commonality) and focused on a particular market segment, but that exhibit significantly different requirements (variability).¹ Over the last decade, researchers have proposed several variability-modeling and SPL approaches for maximizing reuse. However, product line approaches remain challenging in dynamic environments with frequent context changes, which often require run-time adaptation. These new requirements have prompted the emergence of dynamic SPLs (DSPLs) that leverage context awareness and dynamic variability (also called *run-time variability*) to defer product configuration to run time.²

In recent years, DSPLs have become frequent in various research areas, such as workflow-based systems.³ In those systems, variability deals with a set of similar processes (process variants) that contain common and variable activities, adjusting them to meet custom user requirements and context changes. A major concern for process variability in a DSPL, though, is the context-aware configuration of process variants. This means that context information, not users, drives process configuration. So, the system can dynamically configure variants on the basis of context information and effectively deal with dynamic situations at run time.⁴

This is exemplified by automated warehouse solutions that use *operational processes* to move goods between warehouse locations. Each operational process consists of common and variable parts that can be customized to satisfy a particular warehouse's requirements. The variability among such processes can be



denoted by a set of decision points and resolved on the basis of context information from underlying sensors such as presence sensors, scanners, and RFID. However, to make this happen, workflow-based systems must be context-sensitive to support intelligent, run-time decision making and enable smart workflows—that is, processes crossing the boundary to the physical world.⁵

In smart workflows, context awareness and run-time variability are crucial. Both ensure that once the system detects context changes, based on external-sensor data or new user requirements, it can appropriately decide which features of the configurable process to activate or deactivate and can trigger the best selection by a run-time binding. In this light, researchers have proposed approaches and tools to address standard workflow-based systems' shortcomings (see the sidebar). Although such specialized workflow systems seem like a natural way to scope down the challenging problems in the dynamic environment, they don't explicitly address these requirements:

- *Design by reuse.* Common and variable parts can be understood as connected, reusable process structures, which might allow easier, faster development of variant-rich workflow-based systems.
- *Run-time variability.* To effectively deal with dynamic situations of the process context, the system should be able to dynamically configure variants at run time.
- *Context awareness and automated decision making.* In context-aware process configuration, the system should

autonomously handle context information to select different options depending on the conditions.

Because of these limitations, a unified dynamic-variability approach for context-aware smart workflows would significantly increase both flexibility and usability, especially for context-aware and variant-rich workflow-based systems. The LateVa (Late Variability for Context-Aware Smart Workflows) framework constitutes the first step toward this goal.

Introducing LateVa

LateVa aims at the convergence of smart workflows and DSPLs to fill the gap between context provisioning and the dynamic-variability

a set of related process variants. This model is in essence the intersection or greatest common denominator (GCD) of all correlated process variants. These variants share a common part of a core process (captured in a base model), whereas concrete parts (expressed as fragments) fluctuate among variants.

A fragment describes a single variant realization option for each *variation point* in a particular base model. Depending on the *variability description* and considering *context data*, LateVa selects applicable fragments and executes them to resolve context-aware process variability.

To represent variability, developers can annotate a base model with variation points. Their placement has two fundamental goals:

To fully exploit content data in smart workflows, context awareness and run-time variability are crucial.

mechanism needed to enable process variability at run time. Figure 1 shows the framework's main building blocks.

Base Models, Fragments, and Variability Models

At design time, the *LateVa modeler* provides an extended BPMN 2 (Business Process Model and Notation ver. 2.0) editor to let developers define *base models* and *process fragments* (or simply *fragments*). (The editor is an extension of the open source Activiti workflow designer; www.activiti.org.) A base model describes the commonality and variability of a process family—that is,

- specifying a local or remote data endpoint from which context data is gathered and
- indicating a placeholder activity in which variability occurs.

The *LateVa engine* (which we describe in the next section) uses these variation points to retrieve context data and guide fragment selection by exploiting context information and variability dependencies.

The LateVa modeler uses the Clafer (*class, feature, reference*; www.clafer.org) modeling language to describe a *variability model* by including all fragment choices (variants) and corresponding context

RELATED WORK IN PROCESS VARIABILITY AND ADAPTATION FOR SMART WORKFLOWS

To the best of our knowledge, only LateVa (Late Variability for Context-Aware Smart Workflows; see the main article) enables run-time variability for context-aware smart workflows. It achieves this by separating the descriptions of commonality (what is the same across processes) and variability (how and when processes should differ) and by deferring fragment binding to execution time.

Process-variability-modeling approaches such as Provop (*Process Variants by Options*)¹ have extended process-modeling notations with variability primitives. These approaches promote reuse in terms of design-time process variability (static variability). However, variant configuration is often user-supported rather than automated, and few of these approaches are executable in practice.²

Extending traditional service composition techniques to support flexibility is an emerging research challenge.³ Adaptive, flexible service composition, which involves the binding and rebinding of composite services, often driven by quality of service, is important for determining the optimized service sequence.⁴ In a similar vein, variability modeling in dynamic service adaptation is essential for dynamic-software-product-line approaches that employ context data for run-time reconfiguration. Here, the assumption is that a context change will affect new and running instances, migrating only the compliant running instances to the new configuration.⁵

In contrast, LateVa focuses not on reconfiguration but on run-time variability because reconfiguration can be intrusive. For example, it might interrupt the process execution at the end of each activity and might incur performance penalties, as in the DyBPEL (Dynamic Business Process Execution Language) framework.⁶

Several proposals for smart workflows include context-aware reactive-adaptation support.⁷ These approaches comprise different tasks, depending on the context data gathered from pervasive and interactive devices, to decompose the physical-virtual link's complexity (for example, in mobile computing).⁸ But they provide no run-time-variability construct for workflows, which is the core of LateVa.

References

1. A. Hallerbach, T. Bauer, and M. Reichert, "Capturing Variability in Business Process Models: The Provop Approach," *J. Software Maintenance and Evolution: Research and Practice*, vol. 6, no. 7, 2010, pp. 519–546.
2. G. Valença et al., "A Systematic Mapping Study on Business Process Variability," *Int'l J. Computer Science & Information Technology*, vol. 5, no. 1, 2013; <http://airccse.org/journal/jcsit/5113jcsit01.pdf>.
3. A. Murguzur et al., "Process Flexibility in Service Orchestration: A Systematic Literature Review," *Int'l J. Cooperative Information Systems*, vol. 23, no. 3, 2014.
4. D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Trans. Software Eng.*, vol. 33, no. 6, 2007, pp. 369–384.
5. G.H. Alférez et al., "Dynamic Adaptation of Service Compositions with Variability Models," *J. Systems and Software*, vol. 9, no. 1, 2014, pp. 24–47.
6. L. Baresi, S. Guinea, and L. Pasquale, "Service-Oriented Dynamic Software Product Lines," *Computer*, vol. 45, no. 10, 2012, pp. 42–48.
7. S. Smachat, S. Ling, and M. Indrawan, "A Survey on Context-Aware Workflow Adaptations," *Proc. Int'l Conf. Advances in Mobile Computing and Multimedia*, 2008, pp. 414–417.
8. P. Giner et al., "Developing Mobile Business Processes for the Internet of Things," *IEEE Pervasive Computing*, vol. 9, no. 2, 2010, pp. 18–26.

data mappings (context features) in the form of *feature models*. A feature model provides an abstraction for a base model and its variation points for variability description. It includes both the applicable fragments for each variation point and constraints (variability dependencies) for selecting fragments.

LateVa compiles the base models, fragments, and variability models before deployment. In this step, developers can use the Clafer compiler to check that all features and constraints in the variability models are well defined—for example, to validate that each feature has a unique name. Features related to variation

points and fragments use direct naming compounds for the corresponding process model IDs. So, the compiler doesn't check whether the inserted base model and fragment references are already in the *model repository* (a relational database management system) in the *LateVa persistence* module.

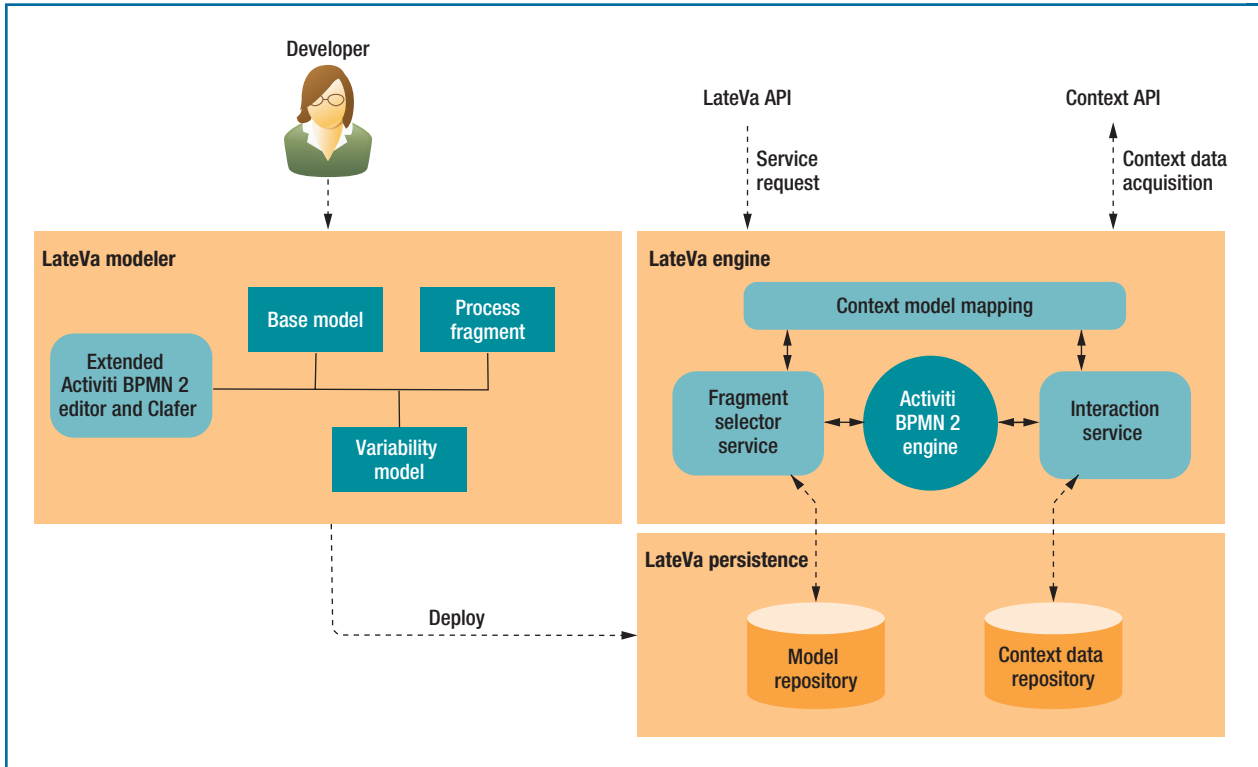


FIGURE 1. The LateVa (Late Variability for Context-Aware Smart Workflows) framework. Design and run-time software modules allow dynamic fragment selection and context-aware smart workflows.

After compilation, developers can deploy the released models to the model repository. This repository embodies base models, fragments, and variability models, as well as their corresponding instances. The LateVa persistence module also includes a *context data repository* as NoSQL-backed storage to store context data coming from the *context API*.

Run-Time Variability

The *LateVa engine* provides an API to receive events and trigger base model execution (see Figure 1). In each request, the *fragment selector service* searches for available base models in the model repository that can satisfy user requirements. For instance, considering the context data from Figure 2, the LateVa engine

will search for base models containing the `storageProcess` key.

A *base model element* stands for any kind of model asset in a base model. As Figure 2 shows, the relationship between a variation point and base model element has a zero-to-one cardinality. This is because not all base model elements (for example, common activities among related process models) are affected by variability. That is, some activities aren't placed as variation points.

A *variability element* is any feature or constraint in the variability model related to process variability elements (variation points and fragments), context data, and the relationships among them. The variability description represents a variation

point's variability (see Figure 2), in terms of

- *noncontext features*, which are linked to process variability elements;
- *context features*, which are directly mapped to context data; and
- *cross-tree constraints*, which represent conditions for valid process configuration.

In LateVa, users can create noncontext features by using naming compounds for the corresponding process model IDs (see the patterns for creating variation point and fragment features in Figure 2).

To realize context-aware smart workflows, we must exploit context

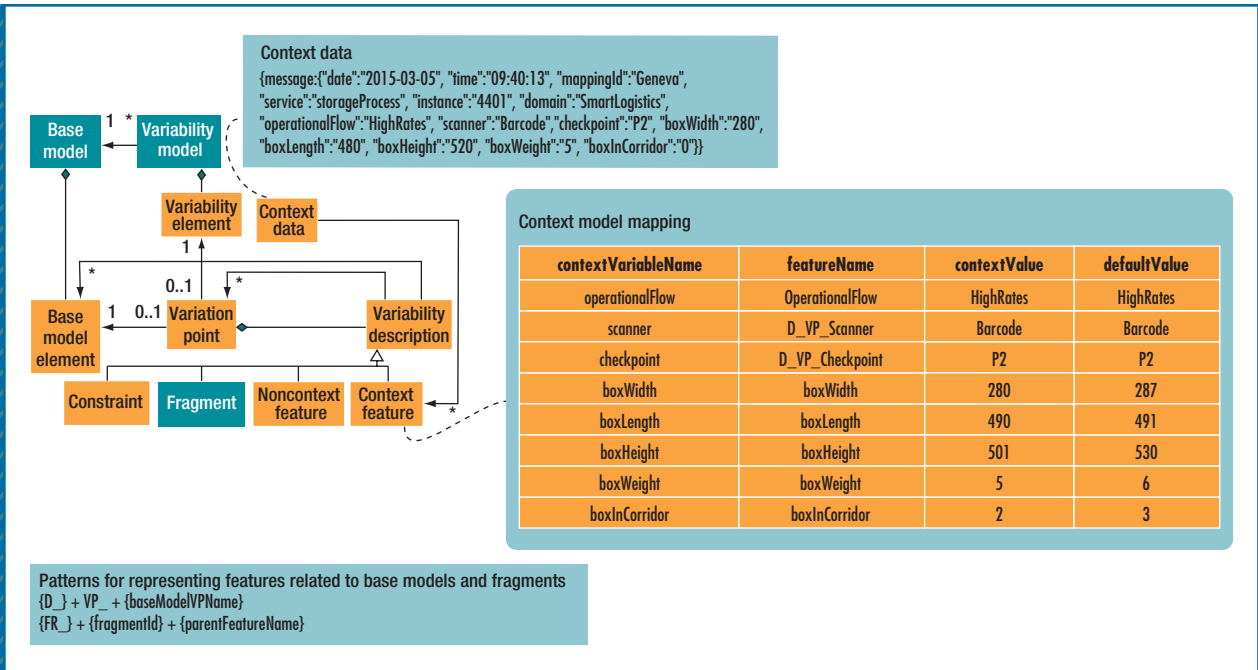


FIGURE 2. Variations of a variability model and context model mapping. The variability model represents both context and noncontext features.

data. We do this by correlating context features in a variability model to those context variable–value pairs retrieved from context information. In LateVa, the *context model mapping* determines the relationship between context features and context data. This model aims to limit which context data operates at each context feature (see the lower right of Figure 2). For that purpose, each context data row defines its

- **contextVariableName**, a concrete variable name in a domain context model;
- **featureName**, a context feature in a variability model;
- **contextValue**, context information for a context variable; and
- **defaultValue**, a valid value that can be assigned to a context variable.

At run time, the *interaction service* deserializes those context

variable–value pairs and parses them to determine all context mappings.

When LateVa receives a context message, it creates an internal context object on the context model mapping. Context data includes three principal values:

- **mappingId** refers to a valid context-model-mapping identifier,
- **service** describes the base model to execute, and
- **instance** identifies an already running base model instance (see Figure 2).

It could also contain extra data that the LateVa engine will ignore but that others (for example, data-monitoring tools) might recognize. If the LateVa engine encounters the context model mapping and base model service descriptions, it follows the preestablished binding type.

The LateVa engine enables two binding times—*startup* and *pure run time*—to delegate the resolution of variation points. The engine relies on context data to automatically determine suitable fragments. Startup binding can be useful when context data doesn’t change over time, so the engine might assign fragments to variation points before smart-workflow instantiation. In contrast, pure-runtime binding can be necessary when variation points’ execution depends on fluctuating context data and thus faces critical decision making.

In startup binding, the fragment selector service first retrieves base model and variability model definitions from the model repository to satisfy the given request. The fragment selector service transforms a variability model into a constraint satisfaction problem (CSP).⁶ In LateVa, every optional context

feature from the variability model becomes a Boolean variable of the CSP with the domain $\{\text{false}, \text{true}\}$ or $\{0, 1\}$, and every mandatory feature becomes a $\{\text{true}\}$. (For example, in Figure 3, **Barcode** will be set as **true** for the **D_VP_Scanner** option.) On the other hand, dynamic context features in the variability model are determined by setting context values (for example, **box-Width** will set **280** as an integer value).

After processing all context values and setting up constraints, the fragment selector service relies on the Clafer solver to run propagation and search and to derive a valid fragment for a given variation point execution. At this point, the engine might handle three situations:

- *No solution.* No suitable fragment choice exists for the current context, so a default fragment is preestablished.
- *One solution.* Just one fragment option exists.
- *n solutions.* The solver returns more than one valid alternative for the current context.

For *n* solutions, we can select from two strategies:

- *Get first* gets the first feasible fragment.
- *Manual selection* enables user decision making.

In pure-run-time binding, each base model instance interacts with different services exposed in the context. Such services provide access to existing context data via the context API. During execution, for each variation in a wait state, the LateVa engine parses context data to set context features in a variability model that's restored from the model repository by the specified base model

instance ID (for example, 4401 in Figure 2). The Clafer solver uses this altered model as an input and concludes whether any sound configuration exists. If just one is valid, all unresolved variation points are determined by a suitable fragment to ensure a proper configuration. How-

the warehouse to running operational processes.

An operational process encompasses a group of structured, identifiable activities that contribute to moving a specified and measurable number of objects inside or outside the warehouse. Each automated

Each automated warehouse might comprise many operational processes and include exchanging sensor data.

ever, the existence of multiple solutions after variation point resolution implies the activation of the selected *n*-solutions strategy (for example, get first) to assign a fragment for pending variation points. Hence, variation points are subsequently resolved on the basis of context data to start an applicable fragment instance.

Example: Smart Workflows for Automated Warehouses

In an automated warehouse, automated systems carry out some or all of the tasks related to storing, retrieving, and moving inventory. Warehouse goods are tagged so that software systems such as warehouse management systems (WMSs) can locate them. The inventory can be continuously updated as goods move in, out, and around the warehouse. This is achieved by automated identification and data capture systems employing technologies such as RFID and presence sensors. Once the overall system has collected data, it can perform batch synchronization, real-time transmission to a datastore, or both. It can even report the status of goods in

warehouse is complex and might comprise many operational processes, such as

- *storage*—storing goods at different warehouse locations,
- *putaway*—removing goods from the warehouse, following different extraction strategies (first in, first out; last in, first out; least quantity; or expiration time), and
- *picking*—taking and collecting a specified quantity of goods.

Each operational process might differ from the preceding one, influenced in various ways by warehouse and location types, storage areas, and sensors. But some processes might have similarities allowing their reuse, to some degree, in different warehouse installations. Such variability should be managed because designing ad hoc operational processes could be time-, resource- and cost-consuming, as well as error prone.

However, in context-aware systems, static variability is often insufficient because you can't predefine which control flow to activate owing to the unpredictability of upcoming

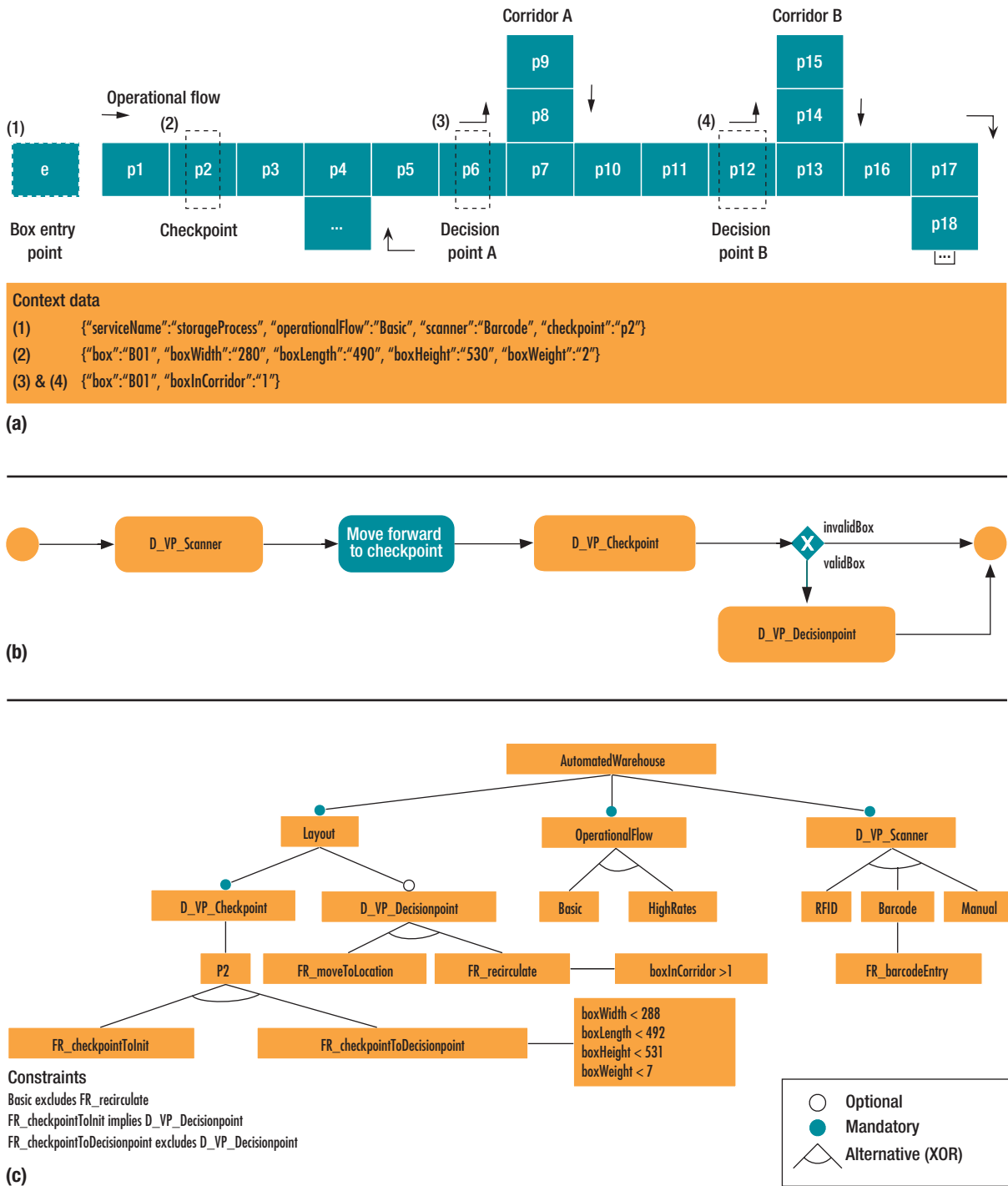


FIGURE 3. The execution of smart workflows in automated-warehouse logistics. (a) The automated-warehouse layout. (b) The storage base model. (c) The variability model. A service request triggers the execution of a base model instance. At each variation point, LateVa uses context data to select a suitable fragment.

context data (for example, sensor data). This means it's crucial to pick up events just-in-time to decide which predefined flows (for example, expressing fragments as variation points' alternatives) to activate. Otherwise, inadequate event processing might negatively affect the day-to-day warehouse operation, often requiring manual intervention (for example, intrusive process reconfiguration).

To illustrate a first step toward dynamic, variant-rich, workflow-based systems, the following automated-warehouse example shows how LateVa can maximize reuse and increase flexibility. The storage workflow reflects a typical scenario in which packaged goods are stored according to different location search strategies (for example, in terms of manual location, fixed location, next empty location, and storage unit type).

Figure 3 presents the storage process's main activities. LateVa orchestrates the entire process to track and control all warehouse flows. It also enables interaction between physical devices (for example, conveyor systems, transelevators, pick-to-light systems, RFID devices, and presence sensors) and warehouse operators (for example, the maintenance manager, workstation agents, and pickers). This provides smart monitoring of the operational process and warehouse status, as well as complex event triggering from context layers.

In this example, the warehouse (see Figure 3a) consists of

- one material entry point at which goods are packed in cardboard boxes and
- two corridors, each containing 2,000 locations.

To initiate the storage base model (see Figure 3b), the box is registered

in the WMS and placed in p1 (each p indicates a point in the conveyor system). This entrance takes place through complex material entry strategies (RFID, barcode scanner, or manual entry). After p1 is scanned, the box moves to p2, where a loading-gauge checkpoint operates.

At this point, the LateVa engine receives checkpoint data to decide

recirculates the box through conveyors (**FR_recirculate** in Figure 3c) until it can go to its location.

In context-aware systems, modeling variant-rich smart workflows enhances reuse of variable parts (fragments), thereby increasing developer productivity. To achieve


LateVa uses context information for process configuration of process variants at run time.

whether the box can be routed to the selected warehouse location. If not, it dispatches a move-backward activity and sends the box to the material entry point for dimension checking, shape checking, or both (**FR_checkpointToInit** in Figure 3c). If the LateVa engine concludes that the box is qualified to be placed in the warehouse, the storage process executes up to a decision point by executing the specific fragment (**FR_checkpointToDecisionpoint** in Figure 3c).

An automated warehouse must guarantee high rates of operational flow so that smart workflows for storage, putaway, and picking are aware of existing boxes in shared conveyors. This is guaranteed by presence sensors, which might indicate the total number of boxes in the shared conveyor systems. The sensors might also ensure that a given threshold isn't exceeded—for example, accepting all moves that fulfill a criterion (**boxInCorridor** > 1 in Figure 3c). If no boxes are waiting to leave the warehouse, the system routs the box to a fixed location (**FR_moveToLocation** in Figure 3c). Otherwise, LateVa

this, these systems preserve the separation of concerns by modeling process variability into disjoint models such as the base models, fragments, and the variability models. Developers can manage process variability independently, reducing its impact on process variants' commonalities. Thus, developers can associate as many variability models as needed with a unique base model, even representing multiple perspectives for different customers. Moreover, continuous context-data processing permits the execution of context-aware smart workflows through consistent mapping between context data and context features in a variability model, thus supporting efficient run-time variability.

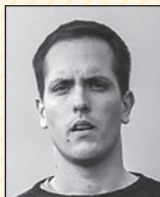
LateVa offers a framework for realizing such systems. So far, besides the basic framework, we've developed a running prototype of the LateVa engine.⁷ We plan to extend our models to incorporate exception handling and to integrate the exception handling with monitoring capabilities to increase availability. We're also extending fragment binding by

reducing the optimization overhead to facilitate the management of large-scale multiple instances of the same smart workflow. As for monitoring, we'll also investigate constraints on the correctness of dynamically configured smart workflows. 

References

1. P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison-Wesley Longman, 2001.
2. S. Hallsteinsen et al., "Dynamic Software Product Lines," *Computer*, vol. 41, no. 4, 2008, pp. 93–95.
3. R.S. Rocha and M. Fantinato, "The Use of Software Product Lines for Business Process Management: A Systematic Literature Review," *Information and Software Technology*, vol. 55, no. 8, 2013, pp. 1355–1373.
4. R. Capilla and J. Bosch, "The Promise and Challenge of Runtime Variability," *Computer*, vol. 44, no. 12, 2011, pp. 93–95.
5. M. Wieland, P. Kaczmarczyk, and D. Nicklas, "Context Integration for Smart Workflows," *Proc. 2008 IEEE Int'l Conf. Pervasive Computing and Communications*, 2008, pp. 239–242.
6. D. Benavides et al., "Automated Analysis in Feature Modelling and Product Configuration," *Safe and Secure Software Reuse*, LNCS 7925, Springer, 2013, pp. 160–175.
7. A. Murguzur et al., "Context-Aware Staged Configuration of Process Variants@ Runtime," *Advanced Information Systems Engineering*, LNCS 8484, Springer, 2014, pp. 241–255.

ABOUT THE AUTHORS



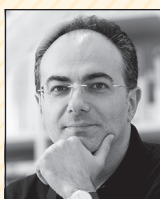
AITOR MURGUZUR is a PhD candidate in computer science at the IK4-Ikerlan Research Center. His research interests include distributed systems, cloud manufacturing, big data, machine learning, the Internet of Things, and workflows. Murguzur received an MSc in computer science from Mondragon University. Contact him at amurguzur@ikerlan.es.



SALVADOR TRUJILLO is the software production area manager at the IK4-Ikerlan Research Center. His research focuses on model-driven engineering, embedded systems, model-based system engineering, and software product lines. Trujillo received a PhD in computer science from the University of the Basque Country. Contact him at strujillo@ikerlan.es.



HONG-LINH TRUONG is an assistant professor in the Vienna University of Technology's Distributed Systems Group. His research focuses on service-engineering analytics, particularly for cloud computing; service-oriented architectures and computing; distributed and parallel computing; the Internet of Things; complex and elastic distributed systems; and context-aware computing. Truong received a Habilitation in computer science from the University of the Basque Country. Contact him at truong@dsg.tuwien.ac.at.




SCHAHRAM DUSTDAR is a full professor of computer science at the Vienna University of Technology and the head of the university's Distributed Systems Group. His research interests are service-oriented architectures and computing; mobile and ubiquitous computing; complex, autonomic, and adaptive systems; and context-aware computing. Dustdar received a PhD in business informatics from the University of Linz. He's an ACM Distinguished Scientist and IBM Faculty Award recipient. Contact him at dustdar@dsg.tuwien.ac.at.



ÓSCAR ORTIZ is an associate professor of computer science and a PhD candidate in the Polytechnical University of Madrid's Telematic and Electronic Engineering Department. His research interests include computer networks, software variability, and self-adaptive systems. Ortiz received an MSc in telecommunications engineering from the Polytechnical University of Madrid. Contact him at oscar.ortiz@upm.es.



GOIURIA SAGARDUI is the head of Mondragon University's embedded-software research group. Her research focuses on software engineering, including software product lines and model-driven engineering. Sagardui received a PhD in computer science from the University of the Basque Country. Contact her at gsagardui@mondragon.edu

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.