Requirements for a Modeling Language to Specify and Match Business Process Improvement Patterns

Alireza Pourshahid, Gunter Mussbacher, Daniel Amyot
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
apour024@uottawa.ca, {gunterm | damyot}@eecs.uottawa.ca

Michael Weiss
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
weiss@sce.carleton.ca

***Abstract*—Businesses are always looking for opportunities to improve their processes in order to become more efficient and effective. Patterns for business process improvement have been defined and used as best practices to help analysts discover such opportunities. A modeling language allowing analysts to define or use a predefined library of *improvement patterns* to detect improvement opportunities in business processes can be of a significant value. Based on a comprehensive set of improvement patterns from the literature, this paper defines the requirements for a modeling language to support a framework capable of defining and detecting such patterns. We use an example from the retail industry to motivate the collected requirements. The paper's contributions allow us to capture more sophisticated business process improvement patterns, bringing us one step closer to a comprehensive model-driven, aspect-oriented business process modeling language. Furthermore, the collected requirements for the desired modeling language clearly indicate that currently popular business process modeling languages are not yet capable of capturing all the required details for business process improvement patterns on a broad scale.**

***Index Terms*—business process improvement, improvement patterns, aspects, goal modeling, AoURN, URN, UCM, GRL.**

I. Introduction

Business Process Improvement (BPI) and redesign approaches have been the center of attention of many organizations for the past several decades. Businesses are always looking for opportunities to provide competitive customer service with smaller cost and higher profit margins. BPI methods were initially focused on manual improvements and fundamental changes to the business (i.e., Business Process Reengineering) [31]. In recent years though, the focus moved to iterative and incremental improvements [27]. Finding a set of patterns that can be applied to various business contexts and improve business processes is a challenging goal. Although generic BPI patterns [28] exist that can be considered good practices for any business, they often require context-related information and customization, preventing them to be readily applied in other contexts. Therefore, we believe only patterns with specific characteristics can be used to detect improvement opportunities in business processes. In addition, defining the patterns themselves requires specific modeling support from the desired process modeling language. In this paper, we mainly discuss the characteristics required to model the patterns to increase an analyst's ability to find improvement opportunities in a large set of existing processes based on best practices. We, hence, identify the modeling language requirements to support the modeling and detection of BPI patterns at the requirements level. We also touch on possibilities for business process improvement through automatic or semi-automatic application of the defined patterns, but argue that human intervention and decision-making is required for that in many cases.

This paper not only contributes to business process improvement by proposing an approach to capture BPI patterns at the requirements level but also contributes to moving model-driven requirement engineering (RE) languages for business processes forward to be able to capture situations that are currently not supported by most RE languages.

In Section II, we discuss the background on BPI patterns and the Aspect-oriented User Requirements Notation (AoURN) as well as the characteristics of BPI patterns that can be modeled using the proposed approach. Note that we use AoURN in this paper to illustrate and concretize some of the BPI patterns. In Section III, we illustrate relevant pattern examples and discuss their application to a business model of a consignment retail store. In Section IV, we summarize the list of identified requirements to improve business process languages based on our investigation. Section V concludes and discusses future work.

II. Background

A. Business Process Improvement Patterns

Patterns are solutions to repeatedly observed problems in a specific context that can be formalized as a reusable solution [2]. Patterns are often presented in a structured format [8]. The context of a pattern is the precondition that makes the pattern applicable and the solution resolves the described design trade-offs (i.e., the forces involved) [17][18]. There have been several efforts to formalize design pattern [32] and to define and automatically apply patterns [7], but these efforts typically do not take concrete performance measurements of a business process into account.

The User Requirements Notation (URN) [10] has been used previously to model architectural patterns [17], more formally defining them while focusing on highlighting their impact on forces and enabling pattern users to perform trade-off

analysis [18]. URN was also evaluated and improved to better support business process and workflow patterns [16].

The BPI patterns introduced by Reijers [28], expressed in plain English, touch on various improvement possibilities and show examples of use at different levels of abstractions. Therefore, these patterns represent an excellent data set to derive the requirements for a comprehensive BPI language (called language X from now on). Since these patterns cover many different best practices accepted in the process improvement community [29], it is likely that any proposed requirements covering these patterns will also enable us to model other potential patterns. TABLE I illustrates the list of patterns and describes their purpose. In section III, we discuss the clustering of these patterns in the context of this paper in more detail (i.e., the last two columns).

TABLE I BUSINESS PROCESS IMPROVEMENT PATTERNS

BPI Pattern	Description	G	R
Knockout	Order knock-outs in an increasing order of effort and decreasing order of termination probability	I	HS
Control Relocation	Move control toward customer	I	HS
Case-Based Work	Remove batch-processing and periodic activities from business process	I	HS
Split Responsiblities	Do not assign responsibility to resources from different functional units	I	HS
Resequencing	Move the tasks to a better place in the processs to reduce the setup time	I	HS
Parallelism	Consider executing the sequential tasks in parallel	I	HS
Case Manager	Make one person responsible for handling of each type of process and point of contact with customer	I	HS
Task Automation	Use technology to improve processes	D	BD
Outsourcing	Outsource part of the entire business process	D	BD
Trusted Party	Use information from a trusted party instead of determining the information	D	BD
Extra Resource	Increase the resources	D	IL
Task Composition	Combine small tasks and divide large tasks	D	HS
Contact Reduction	Reduce the number of contacts with customers and third parties	D	HS
Numerical Involvement	Reduce the number of involved parties	D	HS
Interfacing	Standardize the interface with other parties to reduce the error	D	HS
Integration	Integrate process with supplier or customer to increase efficiency and reduce overhead	N	BD
Exception	Isolate exception cases from the normal flow	N	IL
Specialist-Generalist	Increase number of specialists/generalists depending on the case	N	HS

G: Group, R: Reason for human intervention
I: Improvement, D: Detection, N: Not-Modeled
IL: Instance level information required, HS: Human Smarts required, BD: Business Decision

The patterns can improve business processes from four different aspects (i.e., forces): cost, time, quality, and flexibility. Usually, not all aspects are impacted positively or equally. For instance when the "Extra Resource" pattern is used, time is impacted positively while cost is impacted negatively. Therefore, it is important to decide which aspect of the business requires improvement and what the priorities are before the right pattern can be selected and applied to a business process.

The patterns also affect various entities involved in the business process context including operations, IT, customers, structure, and population as elaborated in [29]. For example while the "Task Elimination" pattern influences operations, the "Case manager" pattern affects the structure.

Although these patterns are helpful for business process analysts, it is not easy to decide when to apply the patterns, especially when there are many existing process models in an organization. This motivates our approach to help analysts detect where and when the patterns can be applied [23].

In TABLE I, we divide the patterns into three groups, namely Improvement (I), Detection (D), and Not-Modeled (N). The patterns in Group I can be modeled and used to improve business processes, if language X supports the requirements identified in the remainder of this paper. A model of the BPI patterns in this group consists of a detection model (i.e., a pattern expression that matches the As-Is process and goals) and an improvement model (i.e., a description of the To-Be process and goals illustrating the suggested changes to be applied to the As-Is process and goal models). The generic suggested changes often require human intervention unless customized for a specific context.

Group D consists of patterns that only have detection models, but do not have improvement models. Therefore, the patterns in this group only find potential deficiencies in the process and leave their resolution to the user. Finally, patterns in Group N cannot be modeled even with languages that support all requirements identified in this paper; this means that neither a detection model nor an improvement model can be defined for patterns in this group. Eleven of the patterns introduced by Reijers [28] fall in this group. However, we only show three of them as examples in TABLE I due to lack of space.

We performed this grouping after trying to model these patterns and realizing that only patterns with the following characteristics can be modeled with language X:

- The improvement model operates at the process model level, i.e., the process is changed for all process instances;
- The improvement model changes the process flow / steps and structure;
- The improvement model adds and removes model elements to and from the business process being improved; and
- The pattern is specific enough to be detectable by the detection model, i.e., it is not at a higher level of abstraction than operational business processes.

Patterns requiring modeling capabilities outside the aforementioned characteristics were placed in Group N (e.g., the "Integration" and "Exception" patterns).

In addition, we observed that patterns with the following characteristics are harder to model with language X. In some cases, without knowing enough about the context it is not possible to formulate a generic improvement model. Consequently, those patterns fall into Group D:

- Require human smarts (HS): e.g., resource elimination, merging/splitting tasks, change responsibilities, and major change to process flow;
- Changes are at business model level and have significant impact on how the business operates (BD): e.g., outsourcing a part of business, investment in technology automation, and changing the organization structure;
- Process instance level information required (IL): e.g., involves resource skills, data used in the process, and context (e.g., type of order/customer).

Column R in TABLE I, specifies why we were not able to capture the improvement models in a generic enough way for application of the patterns without human intervention.

The first and second issues (HS and BD) are both related to the decision-making part of the improvement process, which due to its impact on the business has to be made by a human. Although there has been much progress in the field of Artificial Intelligence, crucial decisions still requires human involvement. For instance, the "Case Manager" pattern requires changes in responsibility and level of authority as well as the flow of the process. Although the pattern can be modeled to detect and suggest an improvement, the final decision and change in the process needs to be made by a person (who also takes responsibility for the actions taken).

The third issue (IL), though, is mainly caused by lack of data and by limitations in capturing information at the level of process instances. For example, the "Exception" pattern is related to the discovery of process instances that do not fit into the majority of the cases, so they can be isolated. We do not address this issue any further in this paper and leave the description of IL requirements for future work. Note, however, that process modeling languages meant for process execution automation as well as logs and data captured in relation to process instances in other enterprise systems (e.g., Data Warehouses) could be utilized for this kind of patterns.

Before taking a closer look at the detailed language requirements imposed by the BPI patterns, there are five basic requirements any modeling language must fulfill to define and detect the required business context, scenarios, and stakeholder goals for BPI patterns at the requirements level:

- support for modeling the behavioral facets of business processes (i.e., the language must support workflow/scenario modeling in some way);
- support for modeling the intentional facets of business processes (i.e., the language must support the modeling of stakeholder goals as well as the impact of business processes on these goals to support reasoning about trade-offs);
- support for modeling of key performance indicators (KPIs) for improved intentional modeling with business process monitoring capabilities;
- support for evaluation and analysis of intentional (goal) models and KPI models, using data provided to the model to allow for business monitoring to quantitatively assess how well a process meets the overall stakeholder goals and KPI targets; and
- support for coordinated, heterogeneous pattern[1] matching against behavioral and intentional models to ensure that structural as well as context-related properties can be detected in the business process model and the specified improvements can be applied to the business process model.

We previously introduced an Aspect-Oriented Framework for BPI [23] that fulfills these five basic requirements as opposed to other languages used for business process and goal modeling such as the languages compared in TABLE II. Therefore, we continue using the framework's language, the Aspect-oriented User Requirements Notation – AoURN [20], to motivate the requirements for language X.

TABLE II COMPARING MODELING LANGUAGES

Languages	Workflow	Goals	KPI	Evaluation	CPM
AoURN [23]	Y	Y	Y	Y	Y
URN [10]	Y	Y	Y	Y	N
BIM [9]	N	Y	Y	Y	N
EEML [12]	Y	Y	N	N	N
i* [33]	N	Y	N	Y	N
NFR [21]	N	Y	N	Y	N
KAOS [6]	N	Y	N	Y	N
Tropos [5]	N	Y	N	Y	N
BPMN [15]	Y	N	N	N	N
EPC [13][30]	Y	N	N	N	N
UML-AD [13]	Y	N	N	N	N
YAWL [1]	Y	N	N	N	N
IDEF3 [13][14]	Y	N	N	N	N
Petri Nets [22]	Y	N	N	N	N

Y: Supports, N: Does not support, CPM: Coordinated pattern matching

For a more detailed comparison of these languages, see [24].

B. *Aspect-oriented User Requirements Notation*

The Aspect-oriented User Requirements Notation (AoURN) [20] is an extension of URN, an international Telecommunication Union (ITU-T) standard for modeling requirements [3][10]. AoURN adds the ability of heterogeneous pattern matching to URN's sub-languages for goal and scenario modeling, i.e., the Goal-oriented Requirement Language (GRL) and Use Case Maps (UCMs), respectively. The details of AoURN's pattern matching approach are explained as needed for the example patterns in Section III. AoURN modeling is supported by an Eclipse-based, open source editing and analysis tool called jUCMNav [11].

Figure 1 depicts the UCM model of the high-level process for a consignment retail store. In this process, customers drop-off and get credited for their consignment items. Start points (●) and end points (|) show the beginning and end of the process, respectively. Figure 1 uses UCM stubs (◇) to abstract from the details of the business sub-process. For instance, the Registration stub contains the UCM model shown in Figure 2, providing more details about the sub-process. Furthermore, URN traceability links (▶) are used to associate the stubs with

[1] Note that in this paper the term *pattern* refers to BPI patterns in the sense of [2] as explained at the beginning of Section II.A except when used in *pattern matching* or *pattern expression*. Then, the term refers to a model that is compared against another model.

Key Performance Indicators (KPIs) (⬭) in the goal model shown in Figure 3, hence defining relevant measurements for the process steps described by the stubs (e.g., the Registration Lead Time KPI is captured for Registration).

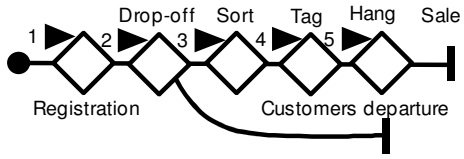

Figure 1. Retail store high level process – UCM example 1

In Figure 2, UCM responsibilities (✕, e.g., Fill out the form) show the tasks performed by different roles (☐, e.g., Customer) in the process. Furthermore, OR-forks (⌥) and condition labels (i.e., [hasAccount]) enable the modeling of alternatives. For example, users without an account are handed a registration form and their accounts are created after they fill out the form. Otherwise, a user with an account proceeds directly to the next step of the process as described by the Drop-off stub.

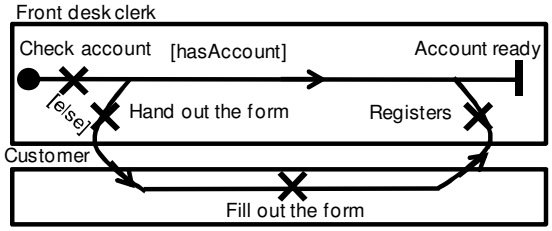

Figure 2. Retail store registration sub-process – UCM example 2

A typical GRL model is illustrated in Figure 3, and consists of high-level objectives of the organization, represented with GRL softgoals (⬭), and KPIs. KPIs are used to measure important factors in an organization. KPIs accomplish this by converting real-world values from the business domain (e.g., 3 days or $15) into GRL satisfaction values that are used in the GRL model (e.g., a value in the range of [-100, 100]). KPIs were added to GRL in [27], and their application to business process validation is discussed in [26]. The latest version of the URN standard now includes full support for KPIs [10]. The Ⓣ and Ⓠ icons identify KPI types, i.e., the time and quality types, respectively.

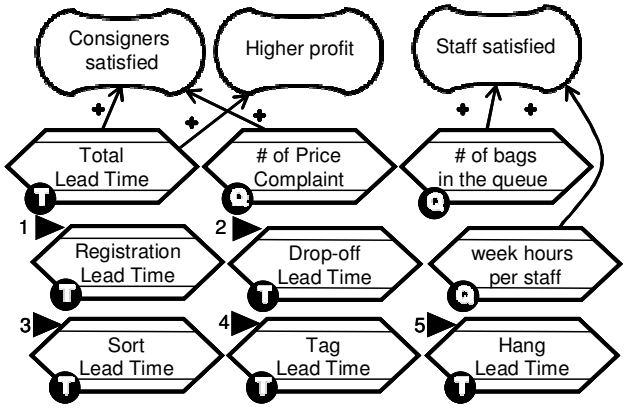

Figure 3. Retail store goals and KPIs – GRL example

Furthermore, contribution links (→) are used to connect the KPIs and goals (e.g., Total Lead Time to Higher profit). Contribution links show the impact of model elements on one another. They are used in GRL evaluation strategies [4] to propagate the GRL satisfaction values throughout the model starting from one or several model elements with initial satisfaction values (typically defined for leaf nodes). Various qualitative and quantitative evaluation algorithms exist [4]. They have been extended in [25] to support the definition of mathematical formulas between KPI model elements, hence enabling the modeler to investigate different relationships between KPIs within the model itself.

III. Example Patterns

In this section, we identify the requirements of language X by modeling three patterns from TABLE I in subsection A, C, and D and highlight the current limitations of business process modeling languages as exemplified by AoURN. Subsection B discusses more advanced support for pattern matching and reporting of matched patterns.

A. *Task Composition Pattern*

The "Task Composition" pattern combines small tasks to reduce setup time and improve turnaround time of a process. Although combining tasks on the surface may seem simple, deciding on which tasks to combine and the roles that will perform the combined task as well as optimizing the combined task requires human smarts. Therefore, we consider this pattern a "Detection" only pattern. Figure 4 and Figure 5 respectively show the behavioral and intentional detection models for this pattern, which helps identify some of the proposed requirements.

The behavioral detection model in Figure 4 matches when there is repetition of either two or more UCM responsibilities (✕) or stubs (◇) (i.e., Option A and Option B, respectively).

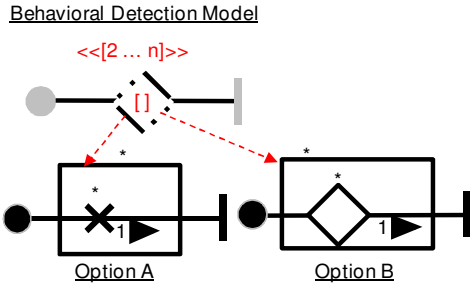

Figure 4. Task Composition – Behavioral detection model

Modeling repetition is necessary for this pattern and some of the other patterns (e.g., "Contact Reduction", "Parallelism", and "Case Manager"). Therefore, we require a *Repetition Container* model element allowing us to capture repetition. In AoURN, we address this requirement by adding a new stub type called "repetition stub" (◈). We use the array and stereotype syntax to show the expected number of repetitions. For instance, in this example pattern, we have used <<[2 … n]>> to show the behavioral detection model only matches, if at least two repetitions exist. Furthermore, the array syntax in combination with URN links (▶) allows each instance of the

repetition to be referenced in the intentional detection model. For example, in Figure 5, [1] and [n] are used to reference the first and n^{th} instance of the repetition and [1 … n] is used to indicate that the *Time KPI is defined for all instances of the repetition. We believe an array of model elements or a construct consisting of multiple model elements is required for flexibility and scalability to allow us to model cases like the "Task Composition" intentional detection model. Otherwise, for model elements like the *Time KPI in Figure 5, one KPI per instance must be added to the detection model.

In addition, since the repetitions may describe elements at a lower level of abstraction (i.e., responsibilities) or higher level of abstraction (i.e., stubs) or a mix of both, we need an approach to show alternative possibilities (i.e., Option A and Option B in Figure 4). This requirement is not limited to this example pattern. Any other pattern that has alternative workflow structure to match against has the same requirement. For instance, the "Extra Resource" pattern can be applied either to a process with one resource performing a task repeatedly or to a process with multiple resources processing a queue, which would be modeled in a different way, hence, requiring a different detection approach. Consequently, we are proposing another type of model element to behave as an *Options Container*. In AoURN, we translate this requirement to the "options stub" (). The stub illustrated in Figure 4 is a *Composite Container* supporting the semantics of both the repetition stub and the options stub, which is yet another requirement for language X. Note that these proposed model elements not only help with defining BPI patterns, but also improve the modularity of requirements models that describe more complex behavioral requirements of a system.

The intentional detection model in Figure 5 matches against a KPI () of type time (), with a name ending in Time (*Time), and associated with the matched responsibility or stub as indicated by the URN link (). A *Condition* (C: <#eachTaskTimeValue) constraints the match to those with a time value below a certain *Constant* (#). In general, an element that needs to be matched is identified by the conventional AoURN icon. Therefore, two more model elements need to be matched, i.e., the ones labeled StartTime and EndTime.

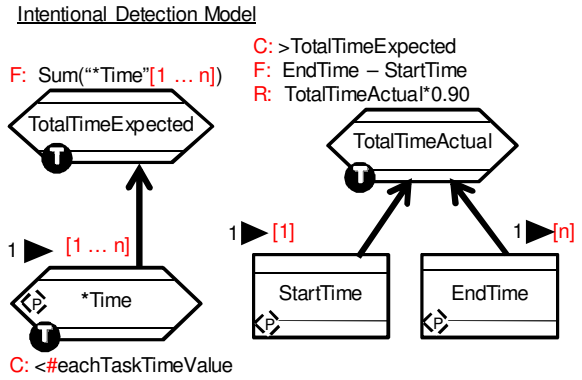

Figure 5. Task Composition – Intentional detection model

While modeling this pattern, we observed the lack of a model element allowing us to use *Raw Data Input* to calculate more complex KPI values in Figure 5. Therefore, we are proposing a new model element and visualize this element in AoURN as a "data model element" (). The difference between a KPI and a data model element is that, for a KPI, a conversion from a real-world value to a GRL satisfaction value takes place, while this does not happen for a data model element and its real-world value is used in the GRL model.

Both of the data model elements in Figure 5 need to be matched, i.e., the StartTime for the first responsibility or stub and the EndTime of the last (n^{th}) responsibility or stub. These two elements are then combined into the TotalTimeActual KPI as defined by a *Formula* (F: EndTime – StartTime), which calculates the actual duration of the process from the first to the last matched responsibility or stub. In addition, a second KPI (TotalTimeExpected) and formula is defined (F: Sum("*Time"[1 … n])) that calculates the sum of all individual task durations. Another condition (C: >TotalTimeExpected) expresses the overall constraint that a combination of tasks takes longer than expected. Conditions, formulas, and constants are used very frequently for intentional detection models and are hence an integral part of the model.

Finally, an anticipated *Result* (R: TotalTimeActual*0.90) is specified for the TotalTimeActual KPI, meaning that the anticipated improvement is a 10% reduction of the duration of the process. This can be used to provide insight into relative advantages of several candidate/competing patterns that can be applied [23], e.g., preference is given to the pattern with the greater positive impact on the overall goals of the business process in Figure 3. In summary, language X needs to support the ability to specify *Conditions*, *Formulas*, *Constants*, and expected *Results*.

B. *Support for Partial Pattern Matching and Reporting*

As can be seen in the "Task Composition" example, the intentional detection model may be quite complex. Using typical pattern matching algorithms, the intentional detection model is matched in the base model, only if the exact same elements and conditions are present. Therefore, modelers are forced to have a very well defined base model to be able to detect and hence take advantage of BPI patterns, which defeats the purpose of assisting modelers in the process improvement journey. For example, the reason why the intentional detection model in Figure 5 is not matched could be that one of the conditions (e.g., C: <eachTaskTimeValue) is violated or because one of the data model elements (e.g., StartTime) does not exist in the base model. However, these two reasons represent two completely different outcomes. The first means that the pattern does not apply, while the second one means that the pattern potentially does apply but a lack of data in the base model prevents a final determination of the applicability at this point in time.

Therefore, language X is required to support a new matching algorithm with partial matching capabilities. The partial matching algorithm does not require the complete structure and conditions to be present in the base model to match. This algorithm matches in three stages. First, it matches against the behavioral detection model where an exact match is required. In the second stage, the algorithm looks for matches against the intentional detection model, but finds partial

matches as well. For instance, if only one of three KPIs defined in an intentional detection model is found in the base model, the algorithm detects a partial match. In the third stage, the algorithm matches against the conditions defined for each matched KPI (note that not all conditions may hence be evaluated). Finally, the result of the partial matching algorithm is a report with the number of matches against the behavioral detection model further broken down into the following categories based on the match against the intentional detection model:

- the number of overall successful matches;
- the number of unsuccessful matches due to violated constraints; and
- the number of potential matches sorted using match probabilities calculated based on the number of matched model elements.

This three-stage partial matching approach is useful because not all model elements required by the intentional detection model to match need to exist in the base model right from the start. In other words, a modeler requires less preparation time for existing process models to use the BPI patterns. In addition, finding partial matches educates an organization to gather the data and add the KPIs and data model elements to their models as required for business processes measurement, monitoring, and improvement. Therefore, this approach reduces the complexity of the application of BPI patterns previously observed [23].

The three-stage partial matching approach, however, has implications on the visualization of matches as full and potential matches need to be differentiated. This highlights another requirement for language X, which is a method for identifying the different types of matches in the base model. Similarly, the patterns in Group D and I also pose different visualization requirements. Therefore, it must be possible to indicate the group to which a pattern belongs in the model. In AoURN, this requirement is realized by allowing the type of the pattern (i.e., one of the two groups D or I) to be defined for each of the patterns.

To give an example of the application of a pattern from Group D, the "Task Composition" pattern is applied against the process described in Figure 1. As illustrated in Figure 4 – Option B, the behavioral detection model matches against a repetition of stubs with any names. Hence, the stubs Registration, Drop-off, Sort, Tag, and Hang are all matched. The behavioral detection model matches against any combination of two or more consecutive stubs, i.e., ten matches of the behavioral detection model are possible and reported as the result of the first stage of the partial matching algorithm.

In the second stage of the matching algorithm, the intentional detection model in Figure 5 is used, requiring a KPI of type time to be found in the model (i.e., the *Time KPI with the Ⓟ and Ⓣ icons). In this case, the lead time KPIs measuring the execution duration of a sub-process are present for each stub in the GRL model in Figure 3. However, the StartTime and EndTime data model elements do not exist in the GRL model. Therefore, the resulting matches can only be partial. In the third stage, the conditions are evaluated. Therefore, only those combinations where each lead time for each stub is less than the #eachTaskTimeValue constant will partially match against the intentional detection model. The other condition is not attempted to be matched, because not all structural model elements required for the condition could be matched during the second stage. Assuming that the lead times of the first three stubs do not violate the condition, the partial matching algorithm hence reports that there are no overall successful matches, seven unsuccessful matches, and three potential matches.

The potential matches are illustrated in Figure 6 using conventional AoURN aspect markers (◆) to denote the beginning (i.e., a1, b1, and c1) and end of a match (i.e., a2, b2, and c2). Therefore, the stub combinations Registration/Drop-off (a), Registration/Drop-off/Sort (b), and Drop-off/Sort (c) are potential matches.

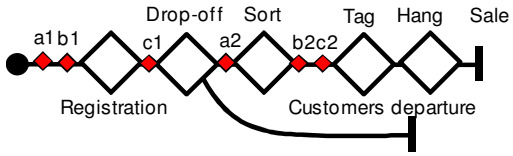

Figure 6. Task Composition – Matches in the UCM base model

Given that these are partial matches, the modeler may now turn his/her attention to what is required for a full match. Therefore, the intentional detection model must be composed with the base GRL model in a way that helps the modeler with this task. In AoURN, the matched, missing, and added model elements are highlighted in the composed model with AoURN aspect markers (◆), in dark red color, and in light gray color, respectively, as shown in Figure 7. The figure shows all three potential matches simultaneously with the help of an array notation and clearly indicates that the StartTime and EndTime data model elements are missing in the current GRL model and need to be added to it. At this point, there are three options. Either these data model elements have already been collected by the organization, in which case they can simply be added to the GRL model, or they are not available. The latter case is an indication for the organization to gather more measurements to be able to improve its business processes, or alternatively, the organization may decide not to collect further measurements, maybe because doing so is too expensive, and hence the decision is made not to improve the business process in this case.

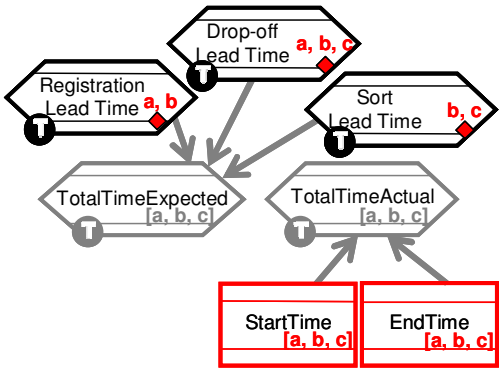

Figure 7. Task Composition – Matches in the GRL base model

In Figure 7, the model elements are marked by similar labels (i.e., a, b, and c) as are used for the matches in the UCM model in Figure 6 to identify the matches that belong to the corresponding aspect markers in Figure 6. Each label is associated with exactly one matched instance of the intentional detection model, e.g., only those elements that are relevant to the Registration/Drop-off (a) match are labeled with (a) in the composed GRL model. Note that if a modeler wants to focus on a single match, then it is straightforward to show this match individually instead of all matches at once.

After all the required model elements have been added to the GRL model, only those cases among the three potential matches found earlier that now satisfy also the condition of the added KPIs (i.e., TotalTimeActual>TotalTimeExpected) will be matched.

C. *Case Manager Pattern*

"Case Manager" is an "Improvement" pattern that detects situations requiring a managerial review or check at the end of a series of tasks or sub-processes performed by various roles in an organization. Although the pattern tends to improve the quality metrics of the processes, it may have negative effects on time and cost.

Figure 8 shows the detection and improvement models for this pattern. The intentional detection model of the pattern matches against quality KPIs (◎) with a value lower than a defined constant called #QualityTarget (note that quotes are used to reference the * KPI name to avoid confusion with the multiplication symbol). The anticipated result (R) of the "Case Manager" pattern is an improvement of 20%. The KPI needs to be defined for a process consisting of a series of tasks as the URN link (▶) refers to the whole behavioral detection model and not an individual element.

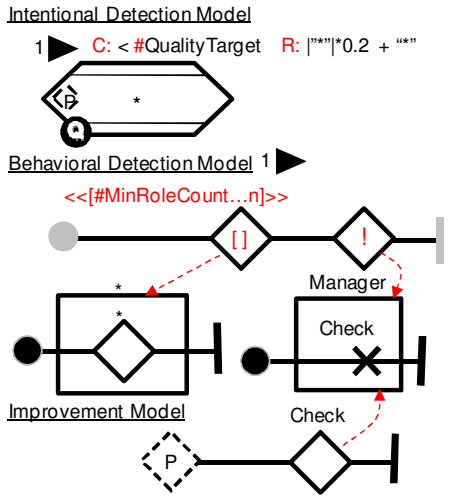

Figure 8. Case Manager – Intentional and behavioral detection models, as well as improvement model

The behavioral detection model uses the repetition stub (◇) proposed earlier. In this case, the repetition stub is used to show a series of at least #MinRoleCount sub-processes (i.e., stubs) executed sequentially in the process. This series of stubs, however, must not already be followed by a managerial check, necessitating an *Absent Container* model element allowing us to describe model elements that must not be matched. We address this requirement in AoURN with another new stub type called "absent stub" (◇).

The improvement model in Figure 8 uses conventional AoURN syntax to describe the effect of applying the "Case Manager" pattern. The pointcut stub (◇) signifies the locations where the pattern can be applied in the process model because the detection model can be matched, i.e. it represents the detection model. Since the check stub is shown after the pointcut stub in the improvement model, it is applied after the matched locations in the process model. Note that the check stub in the improvement model actually reuses the same UCM sub-model used in the absent stub.

Figure 9 shows where the "Case Manager" pattern is detected in the process model assuming that the #MinRoleCount constant is five. The suggested improvement is inserted into the process model again with the help of the conventional aspect marker (◆), which is automatically linked to the improvement model by the conventional AoURN composition mechanism.

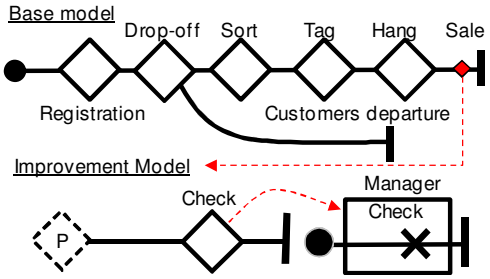

Figure 9. Case Manager – Detection and suggestion in UCM base model

D. *Task Automation Pattern*

Automation is a widespread practice for process improvement that can be as minor as automating an approval process in an organization or as major as changing a brick and mortar to an online business model. Therefore, coming up with a generic "Task Automation" pattern to improve all the potential automation opportunities in an organization is not feasible. However, providing a dictionary of opportunities based on best practices can be viable solution.

As this dictionary evolves over time, it is useful to separate it from the specification of the detection model. Hence, there is a need to specify a *Reference* to an external source as well as an *Action* to be performed with this external source. Figure 10 shows how this requirement is addressed in AoURN for the "Task Automation" pattern. In the behavioral detection model, a responsibility is given a new stereotype (<<orList AutomationDictionary.#Term>>). orList indicates that the external source contains a list of items that can be matched. AutomationDictionary references the external source and #Term accesses a particular data attribute of this external source. The same approach can be used for the "Outsourcing" and "Trusted Party" patterns to find the potential sub-processes that can be outsourced or information that can be acquired from trusted parties.

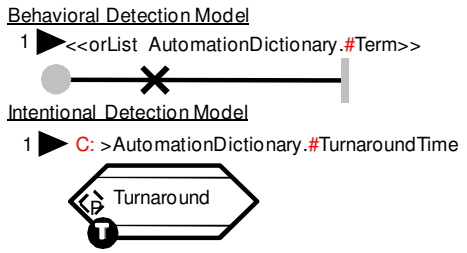

Figure 10. Task Automation – Behavioral and intentional detection models

As indicated by the URN link (▶), the responsibility must also satisfy a condition on its Turnaround KPI. In the condition, another data attribute of the external source is referenced. Note that several actions may be defined (e.g., orList, andList…). Furthermore, general issues of name-based matching such as how to match typos or synonyms are orthogonal to the discussed subject matter and have been discussed in a recent taxonomy [19].

Figure 11 shows where the "Task Automation" pattern is detected in the base model. This example shows only potential matches because the Turnaround KPI does not exist in the base GRL model. Two responsibilities in the registration sub-process (i.e., Fill out the form and Registers) are in the automation dictionary and hence are matched. These matches are again indicated with the help of aspect markers and missing KPIs in red color with the array [a, b] to indicate KPI instances as explained for previous examples.

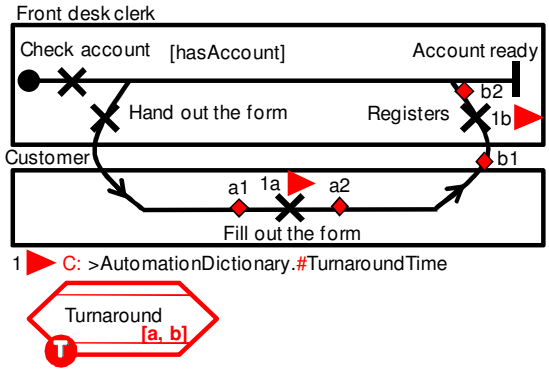

Figure 11. Task Automation – Detection in UCM and GRL base models

IV. Requirements Summary

While the ability to match intentional and behavioral detection models is a basic requirement and is supported by some modeling languages, many other requirements related to the specification and matching of BPI patterns, as discussed with the help of examples in previous sections, are yet to be supported by modeling languages. These new requirements for language X are categorized in TABLE IV.

We prioritize these requirements using five criteria (TABLE III). Each requirement is tagged by these priority criteria in TABLE IV. The priority score of a requirement is the total number of points that the requirement acquires based on these tags. All of the criteria have fixed points except for Patterns (P), whose points are specified based on the number of patterns impacted by the evaluated requirement. The highest possible number of points for P is 15, which is the total number of patterns in Groups I and D. Due to space constraints, we only elaborate the tagging rationale for two of these requirements to convey the thought process we went through for this prioritization activity.

TABLE III Prioritization Criteria

Criteria	Description	Points
Patterns (P)	must have for modeling *N* number of the BPI patterns (maximum for N is 15)	N
Framework (F)	must have for the success of the framework	5
Usability (U)	improves the usability of the framework and language	3
Expressivness (E)	increases the expressivness power of the modeling language	2
Scalable (S)	improves the scalability of the language	1

If "Partial matching and reporting" is not supported by the language, either the defined patterns have to be very generic to match against incomplete base models, which causes many false positives and noise, or the base models have to be very comprehensive and developed having the patterns in mind, which defeats the purpose of the patterns. Therefore, we consider this requirement a must have for the success of the framework (F). If this requirement is supported, the patterns can be modeled including all the KPIs and be as specific as required without worrying about missing potential matches. Therefore, this requirement influences how all patterns are modeled (P = 15). Moreover, the reporting approaches discussed in section III.B increase the framework's usability (U) by allowing modelers to complete their base model and find potential improvement opportunities. Finally, the reporting approach also helps with the scalability (S) of the framework because reporting a high number of matches in a concise manner allows the framework to be used against larger size models and does not rely on human eyes to detect full and partial matches. The total priority score for this requirement as shown in TABLE IV is hence 24 (15 for P, 5 for F, 3 for U, and 1 for S).

On the other hand, while lack of the "Options Container" prevents us from modeling some of the patterns (P = 3), it does not completely prevent us from using the framework (not tagged with F). It, however, is tagged with "E" because it increases the expressiveness power of language X with the ability to describe groups of alternative model elements. Therefore, the total priority score for this requirement as shown in TABLE IV is 5 (3 for P and 2 for E).

V. Conclusion

Business process improvement has been important for businesses for many years. Yet, the frameworks and approaches used are labor intensive, often remain at the level of guidelines, and fail to provide further assistance to analysts. Among other reasons, the lack of a process modeling language, referred to as language X in this paper, that provides the required capabilities for such a framework plays a key role in this deficiency.

TABLE IV REQUIREMENTS FOR LANGUAGE X

Requirement *[AoURN Implementation]* **Description of Requirement** *(Priority Criteria → Priority Score)*
Pattern types identification *[Aspect types]* Language X shall identify the type (i.e., "Detection" and "Improvement") of a BPI pattern to allow for differentiated treatment. *(P = 15, F, U, E → 25)*
Conditions, formulas, constants, and results *[C, F, #, and R]* Language X shall support the specification and evaluation of conditions, formulas, and expected results for KPIs including the ability to use constants. *(P = 15, F, U, E → 25)*
Partial matching and reporting *[Partial AoURN pattern matching]* Language X shall report the number of partial matches and clearly visualize model elements that could not be matched in a partial match to reduce the dependency of the pattern expression on existing KPIs in the base model. *(P = 15, F, U, S →24)*
Identify multiple matches *[Annotated Aspect Markers]* Language X shall support the visualization of multiple matches (i) in a distinguishable way in the intentional and behavioral base models and (ii) ideally with only one model element for multiple, added instances in the intentional base model. *(P = 15, F, E → 22)*
Match KPI types and conditions *[Improved AoURN pattern matching]* Language X shall support matching against the type of a KPI as well as the condition of a KPI. *(P = 15, F → 20)*
Detection highlighting *[Improved AoURN pattern matching]* Language X shall support the visualization of the matches of a BPI pattern in the "Detection" group even though there is no implementation model to be applied to the base model. *(P = 9, F, U, S → 18)*
Repetition Container *[Repetition Stub]* Language X shall support the specification and matching of a group of repeated model elements in behavioral models. *(P = 5, E, S → 8)*
Actions and References *[AoURN Actions and References]* Language X shall support a set of advanced commands and the ability to reference external sources in the specification and matching of detection models. *(P = 5, E, S → 8)*
Array of model elements *[<<[n…m]>>]* Language X shall support the specification and matching of similar intentional model elements that need to be repeated several times as an array of model elements for improved ease of modeling and scalability. *(P = 2, E, S → 5)*
Options Container *[Option Stub]* Language X shall support the specification and matching of groups of alternative model elements in behavioral models. *(P = 3, E → 5)*
Absent Container *[Absent Stub]* Language X shall support the specification and matching of a group of model elements that should not be in the base model for a successful match of behavioral models. *(P = 1, E → 3)*
Composite Container *[Composite Stub]* Language X shall support the combination of various container semantics (e.g., Repetition Container with Options Container) for behavioral models. *(P = 1, E → 3)*
Raw data input *[Data Model Element]* Language X shall support the specification and matching of raw data in intentional models to be used as input for the calculation of KPI values. *(P = 1, E → 3)*

P: Patterns, F: Framework, U: Usability, E: Expressiveness Power, S: Scalability

We first discuss the minimum basic requirements for language X that can be used in a framework taking advantage of BPI patterns. We argue that, to the best of our knowledge, AoURN is the only language that addresses the minimum requirements among a set of well-known behavioral and intentional modeling languages. Hence, we use AoURN to demonstrate the specification and matching of business process improvement patterns, while still collecting requirements for language X independent of AoURN.

An examination of all improvement patterns illustrated by a representative set of example patterns (i) shows clearly that currently popular business process and requirement modeling languages are not yet capable of capturing BPI patterns on a broad scale and (ii) results in a list of requirements for language X prioritized based on five criteria including their impact on the BPI patterns. These requirements need to be fulfilled to support the specification and matching of business process improvement patterns. In addition, we propose solutions for how some of these requirements can be met with AoURN. These requirements are not only applicable to the BPI space but also can be used as a guideline to improve modeling languages in general to better capture behavioral and intentional requirements of a system.

Although the discussed requirements and proposed solutions were applied to a retail store process to illustrate their application, further validation on more patterns and case studies is required to assess the feasibility of the proposed requirements, which is planned as our future work. One considerable risk with such a comprehensive matching mechanism for language X is always performance and scalability in large models, which needs to be investigated in scenarios that are more realistic. Moreover, the formulation of requirements related to the capturing of instance level information, which would enable us to model even more improvement patterns, is another potential avenue for future work. Finally, while the usability and learning curve of aspect-oriented approaches is already a concern [24], this approach adds even more model elements and syntax that need to be understood by the user. Therefore, hiding the details of the languages under a higher level of abstraction and an intuitive user interface is another topic that needs to be discussed in the future. Furthermore, while we endeavored to not be influenced by AoURN, language X requirements should be revisited in the context of at least one more modeling language to ensure they are truly independent from AoURN.

In conclusion, although currently there are no languages that address all the requirements discussed in this paper for BPI frameworks, by using some of the enhancements suggested in this paper, AoURN is currently in the best position to be extended and become language X.

REFERENCES

[1] W.M.P. van der Aalst and A.H.M ter Hofstede, "YAWL: yet another workflow language", *Information Systems*, vol. 30, 2005, pp. 245–275.

[2] C. Alexander, *A Pattern Language*, Oxford University Press, 1979.

[3] D. Amyot and G. Mussbacher, "User Requirements Notation: The First Ten Years, The Next Ten Years", *Journal of Software (JSW)*, 6(5), 2011, pp. 747–768.

[4] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating Goal Models within the Goal-oriented Requirement Language", *International Journal of Intelligent Systems (IJIS)*, Vol. 25, Issue 8, August 2010, pp. 841–877.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, "Tropos: An agent-oriented software development

methodology", *Autonomous Agents and Multi-Agent Systems*, Vol. 8 Issue 3, 2004, pp. 203–236.

[6] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal Directed Requirements Acquisition", *Science of Computer Programming*. Vol. 20, 1993, pp. 3–50.

[7] A.H. Eden, A. Yehudai, and J. Gil, "Precise specification and automatic application of design patterns", *12th IEEE Int. Conf. Automated Software Engineering,* 1997, pp. 143–152.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994

[9] J. Horkoff, A. Borgida, J. Mylopoulos, D. Barone, L. Jiang, E. Yu, and D. Amyot, "Making Data Meaningful: The Business Intelligence Model and its Formal Semantics in Description Logics", *Proc. 11th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2012)*, LNCS 7565, Springer, 2012, pp. 700–717.

[10] ITU-T, Recommendation Z.151 (10/12) *User Requirements Notation (URN) - Language definition*, Geneva, Switzerland, 2012

[11] jUCMNav, version 5.3, University of Ottawa; http:// softwareengineering.ca/jucmnav (acc. Jan. 2013)

[12] J. Krogstie, *EEML2005: Extended Enterprise Modeling Language*, Norwegian University of Science and Technology, Norway, 2005.

[13] B. List and B. Korherr, "An Evaluation of Conceptual Business Process Modelling Language", *Proc. 21st ACM Symposium on Applied Computing*, ACM, 2006, pp. 1332–1539.

[14] R. Mayer, C. Menzel, M. Painter, P. Witte, T. Blinn, and B. Perakath, *Information Integration for Concurrent Engineering - IDEF3 Process Description Capture Method Report*, Knowledge Based Systems, Incorporated, Texas, 2005.

[15] M. zur Muehlen and M. Léonard, "How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation", *CAiSE 2008*, LNCS 5074, Springer, 2008, pp. 465–479.

[16] G. Mussbacher and D. Amyot, "Assessing the Applicability of Use Case Maps for Business Process and Workflow Description", *MCETECH'08*, IEEE CS, 2008, pp. 219–222.

[17] G. Mussbacher, D. Amyot, and M. Weiss, "Formalizing Architectural Patterns with the Goal-oriented Requirement Language", *Proc. Fifth Nordic Pattern Languages of Programs Conference*, 2006, pp. 13–36.

[18] G. Mussbacher, D. Amyot, and M. Weiss, "Formalizing Patterns with the User Requirements Notation", T. Taibi (ed.), *Design Pattern Formalization Techniques*, IGI Global, 2007, pp. 302–323.

[19] G. Mussbacher, D. Barone, and D. Amyot, "Towards a Taxonomy of Syntactic and Semantic Matching Mechanisms for Aspect-oriented Modeling", *Proc. 6th Workshop on System Analysis and Modelling (SAM 2010)*, LNCS 6598, Springer, 2010, pp. 241–256.

[20] G. Mussbacher, *Aspect-Oriented User Requirements Notation*, PhD thesis, University of Ottawa, Ottawa, Canada, 2010.

[21] J. Mylopoulos, L. Chung, and E. Yu, "From object-oriented to goal-oriented requirements analysis", *CACM*, 42(1), 1999, pp. 31–37.

[22] C. Petri and W. Reisig, "Petri net", *Scholarpedia*, 2008, doi:10.4249/scholarpedia.6477, 3(4):6477.

[23] A. Pourshahid, G. Mussbacher, D. Amyot, and M. Weiss, "Toward an Aspect-Oriented Framework for Business Process Improvement", *International Journal of Electronic Business (IJEB)*, 8(3), 2010, pp. 233–259.

[24] A. Pourshahid, D. Amyot, A. Shamsaei, G. Mussbacher, and M. Weiss, "A Systematic Review and Assessment of Aspect-oriented Methods Applied to Business Process Adaptation", *Journal of Software (JSW)*, 7(8), Academy Publisher, 2012, pp. 1816–1826.

[25] A. Pourshahid, G. Richards, and D. Amyot, "Toward a Goal-Oriented, Business Intelligence Decision-Making Framework", *E-Technologies: Transformation in a Connected World*, LNBI, vol. 78, Springer, 2011, pp. 100–115.

[26] A. Pourshahid, L. Peyton, S. Ghanavati, D. Amyot, P. Chen, and M. Weiss, "Model-Based Validation of Business Processes", *Business Process Management: Concepts, Technology, and Application*, IGI Global, 2012, pp. 165-183.

[27] A. Pourshahid, P. Chen, D. Amyot, A.J. Forster, S. Ghanavati, L. Peyton, and M. Weiss, "Business Process Management with the User Requirements Notation", *Electronic Commerce Research*, 9(4), 2009, pp. 269–316.

[28] H. Reijers, "Process Design and Redesign", *Process-Aware Information Systems: Bridging People and Software Through Process Technology*, John Wiley and Sons, 2005, pp. 207–234.

[29] H. Reijers and S. Liman-Mansar, "Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics", *Int. J. Manage. Sci.* vol. 33, 2005, pp. 283–306.

[30] A.W. Scheer, O. Thomas, and O. Adam, "Process modeling using event-driven process chains", *Process-Aware Information Systems*, Wiley, 2005, pp. 119–146.

[31] M. Subramanian, W. Larry, and C. Hossein, "Business Process Reengineering: A Consolidated Methodology", *Proc. 4th Annual International Conference on Industrial Engineering Theory Applications and Practice*, U.S. Department of the Interior, 1999, pp. 8–13.

[32] T. Taibi (ed.), *Design Pattern Formalization Techniques*. IGI Global, 2007.

[33] E. Yu., "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", *Proc. 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*, IEEE CS, 1997, pp. 226–235.