# Exploring Context Sensing in the Goal-Driven Design of Business Processes

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

***Abstract*— As more and more business processes execute in increasingly rich digital environments, there is a great opportunity for these processes to make use of the context data to better achieve business goals. Selecting what context data to employ and how to incorporate context sensing into a business process design is therefore of great interest. In this paper, we propose an approach that allows organizations to proactively identify and explore the space of context information that can be sensed and utilized in a business process, with the aim of selecting such context information that can deliver important business benefits. Then, given the selected context information, the approach derives BP design constraints that determine at which points in a business process the selected context information can be sensed and used. The approach builds upon earlier work on goal-driven design of context-aware business processes. Examples from the passenger transportation domain are used for illustration.**

***Keywords—goals, requirements, context sensing, process design***

## I. INTRODUCTION

Today's enterprises are operating in complex and ever more global business environments, which are also getting increasingly connected, dynamic, and competitive. Under these conditions, organizations need to be able to sense and analyze changes in their business domains (i.e., within their competitive boundaries) and respond appropriately, possibly by modifying their business processes (BPs) or even by changing the objectives they pursue. The properties of the external environment (the *context*) and how they affect processes have long been studied in BP management (e.g., [5][16]) and *context-aware* BPs are the processes that are able to automatically sense such external change at runtime and modify their behaviour accordingly. Their flexibility "consists of an extrinsic trigger for change and mechanisms for intrinsic process adaptation" [16]. Thus, the current conception of context-aware BP design takes a primarily *reactive* or passive stance towards external changes. Goal-driven methods for contextual requirements engineering (e.g., [1][10]) took a similar reactive approach and focused on capturing the effects of various contexts on requirements.

Today, with technologies like low-cost sensing and the internet of things (IoT), wireless and mobile devices, wearables, mobile, social, and cloud computing platforms, as well as big data analytics, more and more of the business context is available as data, waiting to be sensed, analyzed, and used to deliver better business results. Rather than responding to a few pre-conceived context variables, a BP designer should *proactively* seek out and explore the vast context space that can potentially be sensed and selectively incorporated into a BP design. In many cases, using more and finer-grained context data allows for more relevant and precise decisions and better action outcomes and ultimately delivers more business value. Moreover, taking an additional piece of information into consideration when achieving an objective can lead to an important innovation and possibly a competitive advantage. For example, for a ride-hailing service such as Uber, sensing and using real-time supply and demand data when calculating fares can create a better balance between the number of available drivers and passengers willing to use the service at any given time, thus increasing its efficiency and competitiveness.

In this paper, we propose an approach that aims at exploring the availability of data and data sensing technologies to complement the previous work in context-aware BP design and provide a different perspective on context awareness. Instead of focusing on contexts as the external change drivers, we treat any context information as an opportunity to improve the achievement of business goals. We take a goal-based view of business requirements (as in [9]) and elicit and refine business objectives using goal models. The approach then focuses on exploring the business domain in search for the context information that can be sensed and utilized to improve (in terms of the important non-functional criteria) the achievement of operational-level business goals. The context information that is deemed to provide enough benefits to justify its costs (and other potential drawbacks) is then selected for use in the business process implementation. Based on the dynamics of the selected context information, the approach helps to identify constraints on the positioning of the process elements that are responsible for sensing and utilizing that information in BPs.

The ideas in this paper fit well with the vision for the future BPM research outlined in [17], which itself has roots in the earlier business literature [13]. As opposed to the many current approaches that fall into the *exploitative* BPM category and are *inside-out* and reactive, focus on optimization and marginal improvements, and look to identify and eliminate negative deviants, the novel *explorative* BPM approaches should be endowed with the "*outside-in*, environment scanning capability" [17] in search of new, innovative ways to achieve business objectives. Explorative BPM focuses on identifying and assessing external

opportunities and our approach addresses one aspect of this – the exploration of the potentially relevant context information.

Our approach allows organizations, in addition to being reactive to changes in the context (which is supported by the earlier context-aware BP approaches), to be proactive and sense the properties of the environment that may not be significant enough to force a change in high-level business requirements and/or a BP structure, but important enough to support improvements in how business goals are achieved. We see information-rich domains, such as logistics, retail personalization, smart cities, etc., as being ripe for the application of our approach. Note that organizations operating in dynamically changing business domains would need to re-apply the method described in this paper periodically to identify and explore new context information and the opportunities to utilize it to achieve their goals. It would also be beneficial to run this analysis whenever the organizations' (functional and quality) goals or the priorities among them change: given the changed objectives, the evaluation of the benefits of using the various pieces of context information to achieve business goals may also change.

The rest of the paper is structured as follows. Sec. II discusses how potentially relevant context information is identified and analyzed. Sec. III describes the process of identifying constraints on BP design, while related work is discussed in Sec. IV. Sec. V concludes the paper and outlines future work.

## II. Exploring Contexts for Achieving Objectives

Given a business objective G, what context information can be used to improve how it is achieved? In this section, given such a goal, we use ER models [3] to identify the relevant information that can potentially affect important quality criteria associated with G. We then evaluate the costs and the benefits of taking each piece of context information into consideration and settle only on those context variables, which produce enough benefits to justify their costs, while rejecting the others.

### A. Goal-Driven Business Process Design

In this paper, we take the goal- and requirements-driven approach to BP design and use the general method from [9] to guide the elicitation and refinement of business goals. Fig. 1A presents a fragment of a goal model for a ride-hailing/ridesharing service (which can be called a Transportation Network Company (TNC)), such as Uber, Lyft, etc. These networks act as electronic platforms connecting transportation service consumers (passengers) and providers (drivers) by implementing app-based matching and filtering services, a driver/passenger ranking mechanism to instill trust and security into their interactions, payment processing services, etc. The fragment focuses on refining the objective Transport Customer, which gives rise to one of the main value-creating processes for such companies. The model consists of a functional goal decomposition hierarchy with AND/OR goal refinements with the obvious semantics and a set or a hierarchy of *softgoals*, which represent non-functional objectives (NFRs) associated with the functional goals being refined (see Fig. 1A). The third element of this model is the contribution links that *qualitatively* (using the help (+), hurt (–), make (++), or break (--) links) specify how the achievement of particular (soft)goals contributes to the *satisficing* of certain other softgoals. Then, a softgoal is satisficed if there is sufficient positive and little negative evidence for this claim. Fig. 1A shows how a service transports a customer, starting from picking him up at the present location and finishing with unloading him from a vehicle. Notice that there are two fare options in the model: the fixed one and the variable one. Relevant NFRs are elicited for evaluating these and other alternatives. In addition to increasing profits, the organization wants the fares to be fair to customers and to make sure that the drivers' earnings are also fair. Likewise, it wants to avoid complexity and make sure that there is enough supply of vehicles to satisfy customer demand. In the model, contribution links are used to show that fixed fares are simple, but too inflexible to support the right supply/demand balance and ensure fair fares and driver earnings. On the other hand, the variable fare option (the highlighted Calculate Fare goal) is more complex, but contributes positively to the rest of the NFRs.

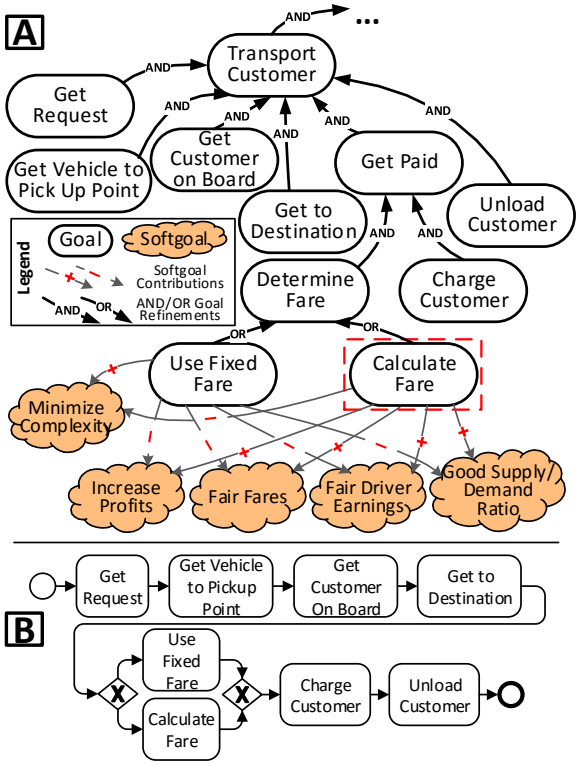


Fig. 1. A) A partial goal model for a ride-hailing service. B) The derived Transport Customer process.

In [9], goal refinement terminated when goals were simple enough to be achieved by a *process element (PE)* – an activity or a decision. By default, goal models do not represent the ordering of goals. To help map goal models into process models, the approach in [9] added control flow annotations (e.g., sequence and parallel) to goal models to capture dependencies among subgoals (in Fig. 1A, we simply order subgoals left to right based on their precedence). Then, leaf-level goals were mapped into the activities that achieved them. The activities were arranged in the newly created BP (Fig. 1B) based on the ordering of their corresponding goals. Note that while ORs in goal models are mostly viewed as being design-time choices for selecting the best implementation option, the goal refinement

choices can be preserved throughout the implementation to obtain flexible/customizable BPs (see [9]), as done in Fig. 1B with XOR gateways to keep the two fare calculation choices open.

In this paper, we modify the approach of [9] and propose a number of additional steps to be performed before mapping leaf-level objectives into PEs. These are aimed at exploring for such leaf-level goals a new type of variability – the choices about what potentially relevant context information to use when achieving them.

The use of contexts in goal-oriented requirements engineering (GORE) is not new. Previous approaches for handling contexts in GORE (e.g., [1][10]) focused on capturing and analyzing domain variability and its relationship with system requirements (goals). For example, [10] used special annotations to model how contexts affected goal models: whether objectives had to be achieved in all contexts or just in some of them, how goal refinement choices contributed differently to NFRs in different contexts, etc. E.g., when an e-commerce company is shipping goods, the goal Clear Customs would appear only in the context of international orders. This makes system requirements models dependent on changing contexts and represents an important step in requirements analysis for systems operating in different and dynamic environments, and eventually leads to context-sensitive BP specifications.

This paper proposes a different way to take context information into consideration in a goal-driven design of BPs. It is complementary to the context-driven approaches described above and can be applied after the effects of domain contexts on requirements have been captured.

Given a leaf-level goal G, such as Calculate Fare (as in Fig. 1A), we proactively look for the context information that can be sensed at runtime and taken into consideration to improve the outcome of attaining it w.r.t. the pertinent NFRs. We expect the mandatory information for achieving G (e.g., the origin and destination for calculating routes) to be specified explicitly through goal parameters (as suggested in [9]) and instead seek the *additional/optional* data to utilize for attaining G. Thus, any piece of contextual data can be omitted if it does not provide enough benefits to justify its costs. Note that in some cases, there is no information that is absolutely necessary for achieving an objective (e.g., when calculating trip fares – they can simply be fixed).

While our approach can be used for any leaf-level objective in a goal model, it makes most sense to apply it to those goals where the utilization of additional context information promises significant business benefits. The following steps of the approach will be applied for each such objective.

### B. *Identifying Potentially Relevant Contexts for a Business Objective*

At this point in the approach, a goal model refining some business objective has been developed and a method for capturing contextual requirements, such as [2][10], has optionally been applied as well. Given an objective G selected as outlined above, we now look at which domain information besides G's mandatory parameters can improve the outcome of achieving G.

We use domain models captured using ER diagrams (UML class diagrams [15] can also be used) to identify the potentially relevant context for G. To construct these domain models, we look at the entities in the domain and their attributes that can be used in the achievement of the goal. For instance, when calculating trip fares for a ride-hailing service (see Fig. 2), since Fare is an attribute of Trip, Trip is identified as an entity directly relevant to the goal. All of its properties (Distance, Duration, Date, etc.) are potentially relevant to fare calculation. Then, we recursively look for domain entities that are related to those already identified (in Fig. 2, these are Customer, Vehicle, and Driver) and model their attributes as potentially relevant for Calculate Fare. We call the identified attributes *context variables (CVs)* and sometimes use the dot notation (e.g., Driver.Rating) to refer to them. Let us see how some of the attributes of the identified entities can potentially be used when calculating fares:

- Trip: Distance and Duration are routinely used by the taxi industry to calculate fares; Origin/Destination can be used to support differentiated fares based on the location (e.g., trips from/to airports can be priced differently); Date can also be used to differentiate fares.
- Driver: fares can be differentiated based on the driver's Rating (e.g., lower fares for low-rated drivers to entice customers to give such drivers a chance to improve).
- Customer: Discount Group (e.g., child/senior) can be used to offer discounts to passengers; Rating – used the same as for drivers.
- Vehicle: Type can be used to charge fares based on the size (and class) of vehicles; their Fuel Consumption can be used to better offset fuel costs.

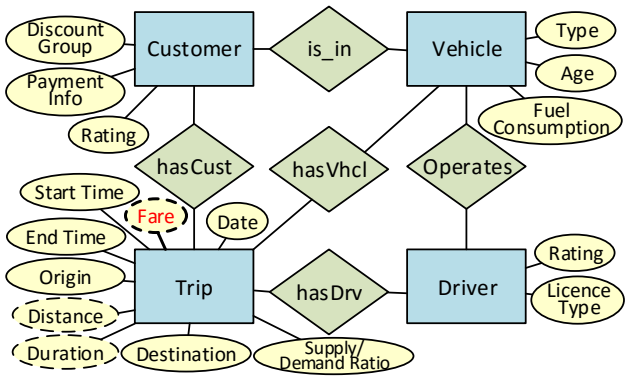


Fig. 2. ER diagram for the Calculate Fare objective. Note that Fare is a derived attribute.

Sometimes, whether certain context variables should be used in achieving a given objective is determined by business-level decisions. E.g., a company may decide not to offer discounted fares. Then, the related attributes (e.g., Discount Group in Fig. 2) are removed from consideration and will not feature in the subsequent steps of the approach. The rest will be analyzed further.

### C. *Analyzing Context Variables*

At this point, we have, for a given goal G, a set of potentially relevant CVs. We now evaluate these CVs by identifying benefits, drawbacks, and costs of using them when achieving G. Then, a subset of the CVs with the positive net benefit is selected. We analyze the CVs using the steps outlined below.

**Step 1: Identify the relevant NFRs as the evaluation criteria**. The benefits/drawbacks of using a CV to achieve G are evaluated w.r.t. the relevant NFRs. There are a number of sources for such NFRs. First come the NFRs from the original goal model, such as those in Fig. 1A. They represent the organization's main domain-specific quality objectives.

Second are the domain-independent NFRs related to context awareness. Overall, using additional CVs leads to more context awareness. Being more context-aware leads to better *responsiveness to change*, *flexibility*, etc., which are important for organizations operating in dynamic environments. These benefits come at a cost of increased *complexity*, *unpredictability*, etc.

Third are the data sensing *costs* used to evaluate options for sensing CVs. CVs may require large upfront investments in the sensing and data analysis infrastructure. This includes hardware sensors and/or instrumentation of software systems residing inside and outside the organization to enable data collection. While such an investment will be amortized over a large number of individual sensing actions, avoiding it may be possible by using novel means of obtaining the required data. E.g., ride-sharing services let users supply credit card information upfront – online or through their mobile apps. This eliminates the extremely costly investment in mobile credit card terminals. In contrast, taxicab companies so far have not been able to avoid the significant investment in the reader infrastructure due to a different business model. On top of the sensing infrastructure costs we have the marginal costs incurred while sensing each piece of information. E.g., when getting a user feedback through paper questionnaires, their processing costs are not negligible.

Fourth are the data quality-related NFRs that are also important for evaluating context sensing options. These include *accuracy*, *correctness*, *relevance* (the steps of the approach outlined in Sec. II.A are designed to help in this regard), *completeness*, *validity* (we return to it in Sec. III), *consistency*, etc. Different means for sensing a CV may produce data of different quality, which can affect the overall benefit of using that CV to achieve a business goal. Thus, for a sensing option, the quality of data it produces must be traded off with its sensing costs.

Fifth – the (re)development of the goal achievement procedure to make use of additional CVs must be considered. Sometimes, using extra CV leads to significant changes in these procedures. The benefits of utilizing CVs must compensate for these development costs. Note that these costs are likely to be incurred for any CV, so for comparing them, we can establish a development cost baseline and concentrate on comparing the development costs for each CV to the baseline.

Note that the above NFRs are not equally valuable. Different organizations will have different preference orders among these NFRs and such preference orders may also change over time.

**Step 2: Determine which CVs and their combinations can be used for achieving G**. Let us look at the Calculate Fare goal and Trip.Duration and Trip.Distance as the selected potentially relevant CVs (see Sec. II.B). We need to evaluate the benefits of using each CV to calculate fares. What about using both variables? This is how most taxi services calculate fares. Do we have to additionally evaluate the situation where they are used in combination or can we just add their independently calculated costs and benefits to evaluate their combined use?

If we begin with a set of N potentially relevant CVs, then, to select which subset to actually implement, it seems that we need to exhaustively evaluate all the possible subsets of that set. There are $2^N$ such subsets. The time complexity of this process is exponential, which makes it quite impractical. One reasonable approach is to make the simplifying assumption that the N context variables can be evaluated independently and the evaluation of any combination of them will amount to calculating the sums of their costs and benefits. Formally: for any pair of CVs, $c_i$ and $c_j$, $cost(c_i + c_j) = cost(c_i) + cost(c_j)$ and $benefit(c_i + c_j) = benefit(c_i) + benefit(c_j)$. In other words, no combination of CVs results in emergent costs or benefits. This *independence assumption* states, for instance, that the combined use of both Vehicle.Type and Trip.Date to calculate fares results in no additional costs or benefits. Under this assumption the evaluation process becomes linear in N. If there exist combinations of variables that violate the assumption, then they should be evaluated separately, which in the *worst case* brings the time complexity back to being exponential.

Fig. 3A presents a general pattern for enumerating the options for evaluating potentially relevant CVs when achieving a goal G. It shows that G has a set of mandatory parameters MP and can be refined into N alternative goals, each adding one optional CV ($c_i$) to the mandatory parameters. Then, combinations of CVs that violate the independence assumption must be evaluated separately. Overall, the model presents a space of options for taking CVs into consideration when achieving G.

Fig. 3B instantiates the generic model for Calculate Fare. We first go through 9 CVs that we assume were deemed potentially relevant in the previous step of the process (Sec. II.B). Note that Calculate Fare has no mandatory parameters. Then, one combination of CVs violating the independence assumption is added (see the dashed oval): if both Trip.Destionation (fares differentiated by destination) and Customer.Discount (fares discounted for some customer groups) are used, then their combination can provide vastly superior benefits for some customers – e.g., for seniors, cheap and fast transportation to hospitals.

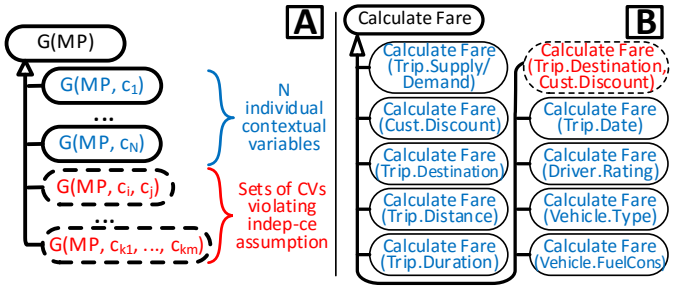


Fig. 3. A) Enumerating options for taking CVs into consideration for a goal G. B) The space of options for using CVs for achieving Calculate Fare.

**Step 3: Identify and evaluate sensing options for CVs**. CVs need to be sensed before their values can be utilized for achieving G. Sensing costs may preclude a CV from being a feasible option as its benefits may be marginal compared to these costs. There may also be multiple sensing options for a CV – each with its own costs and benefits. Also, different sensing

means may produce data of varying quality, thus affecting the potential benefits of that CV.

Identification/evaluation of sensing means has to be performed for every option identified in Step 2. Thus, in our ride-hailing example, we have to do this for every alternative represented in Fig. 3B. Goal models capturing a Calculate Fare variant, ways to sense the CV(s) utilized in it, and the relevant NFRs are used for such evaluation. Fig. 4A shows how this is done for Calculate Fare(Trip.Supply/Demand Ratio) that utilizes the ratio of the current transportation services supply and demand.

In Fig. 4A, the root goal refinement shows that to calculate fares using the supply/demand ratio, the ratio needs to be sensed and then used in the calculation. Fig. 4A captures the two options for sensing the CV (Fig. 4B in Step 4 focuses on the ways to use the CV to calculate fares): sensing the supply/demand ratio within a particular ride-hailing service and sensing the overall ratio – across multiple competing services and possibly even including the available public transit options. These two options are then evaluated against two NFRs: Minimize Cost(Sensing) and Data Completeness shown in bold in Fig. 4A. These softgoals cover two NFR categories from Step 1 (data sensing costs and data quality). Note that Minimize Cost can be further refined into Minimize Upfront Cost and Minimize Marginal Cost as per the discussion in Step 1, but we omit this for brevity.

Contribution links show the evaluation of the sensing options. Sensing the overall supply/demand ratio fully *satisfices* (++) Data Completeness since the data covers all service providers. This option's sensing costs are high since it has to sense and aggregate information coming from many systems. On the other hand, sensing the internal supply/demand ratio only covers the supply within the service and thus is cheaper to sense, but the data is much less complete. Data completeness may play a role if the ride-hailing service uses the information to vary prices based on the supply/demand ratio. E.g., based only on its internal data, a ride-sharing company may decide that given the current high demand and low supply, fares need to go up to improve the ratio. However, the incomplete data fails to account for the possibly high availability of other transportation services. In this situation, the price hike will inevitably lead to unfairly high fares and consequently to customers choosing alternative service providers to meet their transportation needs.

Fig. 4A also shows that data-related softgoals can in turn contribute to higher-level NFRs representing the company's overall objectives. This helps evaluate the impact of sensing options on business goals. Here, Minimize Cost (Sensing) positively contributes to Increase Profits. This link helps evaluate the impact of each sensing option on profits.

The aim of this process step is to identify the best way to sense a CV, so each option's benefits need to be evaluated against its costs. Very frequently the NFRs used to evaluate options cannot be optimized for at the same time, which requires a trade-off analysis. E.g., in Fig. 4A, the preferred option for sensing the CV will depend on the relative importance of Data Completeness and Minimize Cost or those high-level NFRs that they contribute to. However, sometimes it is impossible to determine which CV sensing option is outright better. In this case, the analysis needs to be deferred to Steps 4 and 5 of the process where each sensing option can be analyzed separately. E.g., in Fig. 3B (Step 4), Calculate Fare(Trip.Supply/Demand) can be replaced by Calculate Fare(Internal Trip.Supply/Demand) and Calculate Fare(Overall Trip.Supply/Demand) that correspond to the two ways of sensing the CV. Then, in Step 5, which looks at *all* potentially relevant CVs and can take into account the overall priorities of the organization, the best option will be selected.

**Step 4: Employ high-level NFRs to evaluate the benefits of using a CV to achieve the business goal**. Evaluating sensing options for a CV (as in Fig. 4A) is important, but simply sensing it does not provide any benefits. We use goal models to explore how the sensed data can be used to achieve the given objective G and analyze what value it can deliver (compared to achieving G while using just the mandatory parameters). Fig. 4B continues with the Calculate Fare example, but focuses on the options for Do Fare Calculation(Trip.Supply/ Demand). Here, we elicit and model alternatives for utilizing this information to calculate fares. One option here is to raise fares to increase the supply of drivers – this is called "surge pricing" by Uber. This option is presented in Fig. 4B. All the options need to be evaluated with respect to the business-level quality objectives (e.g., from Fig. 1A). We do not explore these options due to the lack of space, but show how Surge Pricing stacks up against the NFRs. We estimate that this practice helps balance supply and demand, has a positive effect on driver earnings, but contributes negatively to fair fares. Whether the overall outcome is positive or not will depend on the relative priorities of these quality objectives.

**Step 5: decide which CVs to use for achieving the objective G**. We now go back to evaluating, for each CV (or a combination of CVs where the independence assumption does not hold), whether it provides enough business value to justify its

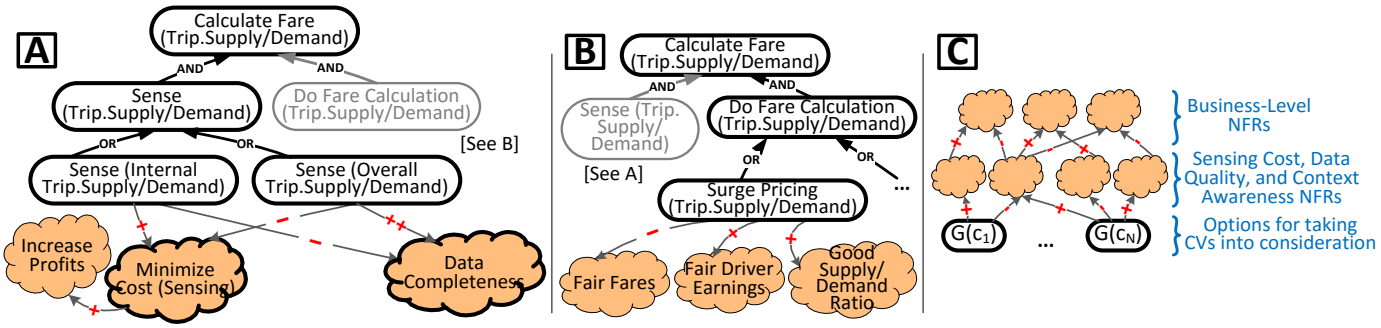


Fig. 4. A) The sensing options for Calculate Fare(Trip.Supply/Demand). B) Analyzing options for using the sensed data for Calculate Fare(Trip.Supply/Demand). C) The general pattern for analyzing the use of context variables in achieving an objective G.

costs and therefore whether it should be utilized when achieving the given business goal. The output is a set of CVs selected for implementation. Adding a partial order on this set based on the value the CVs provide is beneficial for the discussion in Sec. III.

To evaluate whether each CV is worth implementing, standard goal model analysis techniques [6], which primarily focus on propagating goal satisfaction labels through multiple layers of (soft)goals, can be used. Fig. 4C shows the main layers of (soft)goals this analysis has to go through in our approach. Tabular representations can be used when many options/NFRs are present. For finer-grained analysis, qualitative contribution labels can be replaced with numerical values to indicate the strengths of the contributions. For a large number of alternatives, optimization techniques can also be used to identify the best CVs to use. E.g., given a limited budget, finding a set of CVs that deliver the best business value on this budget is an example of a well-known *knapsack problem*.

## III. Context-Driven Process Design

In this section, we look at the dynamics of the relevant context variables – when they become defined, when they become available for sensing, etc. – and its implications on when the CVs can be sensed and utilized to achieve objectives.

We now switch from discussing what context information can potentially help in achieving a particular leaf-level business goal, such as Calculate Fare, to discovering and analyzing the constraints that the previously selected CVs impose on the PE achieving that goal. The PE, which we call the *Main PE*, represents the implementation of a leaf-level business goal that takes into consideration the previously selected context(s). The constraints on the Main PE restrict when it can be executed in a business process. We will also discuss the *Sensing PEs*, which are needed to support the sensing of CVs.

### A. Identifying Process Design Constraints

At this point in our approach, we have identified, for some objective G, a set of context variables that will be utilized for achieving G. Then, for each CV, we identify the constraints that it imposes on the Sensing PE and Main PE. The information can be sensed only when it is defined (i.e., has a value) and available (for sensing) and the Main PE can be executed only after the information has been sensed and until the CV changes its value.

The way we approach constraints identification is by looking at the lifecycles of the context variable and its sensed value available in the system. The statechart that captures both of these lifecycles is shown in Fig. 5A. It is rather generic, but makes certain assumptions about the behaviour of the CV. Particularly, the model assumes that the CV is undefined initially and that once it acquires its value, it is initially unavailable for sensing, but then may cycle through being available and unavailable. Such a model is needed for each context variable selected in Sec. II.C and will reflect its particular dynamics.

In the statechart in Fig. 5A, we use two orthogonal regions to capture the behaviour of the ActualValue and SensedValue of a context variable. In the model, we have the following triggers that trigger state transitions:

- TDfn – causes the actual variable to get initialized (i.e., to become defined). E.g., when the (final) route for a trip is known, the Trip Distance variable acquires its value.
- TAvl – makes the context variable available for sensing. E.g., when a truck exits a tunnel, this enables its GPS-based position sensing.
- TUnvl – makes the context variable unavailable for sensing. E.g., when a truck enters a tunnel, this makes it impossible to sense its location or when a customer leaves a store, this makes it impossible to sense his feedback on the transaction that has just happened.
- Changed – indicates that the ActualValue has changed.
- VarSensed – shows that the ActualValue has been sensed.

These triggers are generic and will be instantiated for each CV. E.g., for tracking trucks, PositionLost will replace TUnvl). Thus, when instantiated, the triggers capture what causes particular CVs to acquire/change their values, what makes them available/unavailable for sensing, etc. More than one domain-specific event can trigger transitions in the model and all of them should be captured in the statechart.

Regarding changes in the variable's actual value, it might not always be possible to know when they happen. In some cases, this information is pushed into the system by its environment. In others, it has to be pulled from the environment by using sensing/probing. It is possible that we have an idea about the volatility of the CV – i.e., about its change cycle. We may know it or may be able to predict it. In this case, the rate of change can be represented in the model (see Fig. 5B where Changed is replaced by a time-based trigger). In this paper, we assume that there are always ways to evaluate Changed triggers – i.e., to know when CVs have changed.

Continuing with the subject of change, when evaluating how a CV influences a given objective, we need to compare the cycle

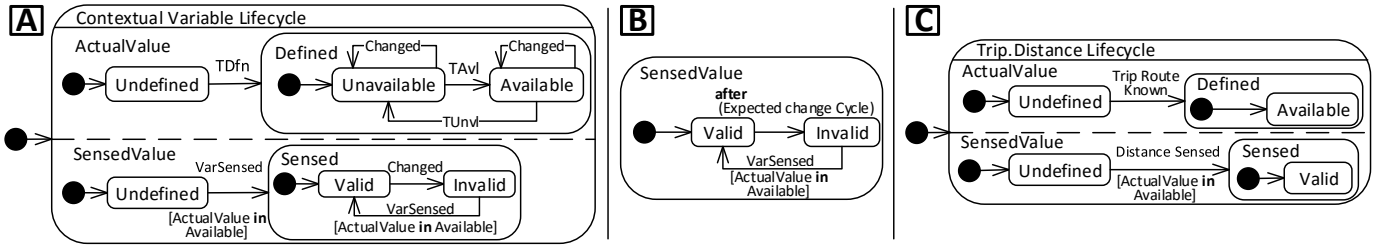


Fig. 5. A) A statechart showing the lifecycle of a contextual parameter. B) A statechart fragment for SensedValue showing a context variable with a known change cycle. C) The lifecycle of Trip.Distance (actual and sensed values).

time of the process where the goal is to be achieved and the volatility (the rate of change) of the CV. Many domain attributes change too slowly for a relevant BP. In this case, while the CV can be an important piece of information for achieving the objective, it can be assumed to be constant within that process, thus simplifying the statechart representing its lifecycle.

The crucial element of the model in Fig. 5A that integrates the behaviour of the variable's actual and sensed values is the guard condition on VarSensed: [ActualValue **in** Available]. It guarantees that sensing can only be done when the context variable is available for sensing. Moreover, for the sensed context information to be usable when achieving an objective (i.e., for executing the Main PE for that goal), SensedValue must be in the Valid state – when it truly (or closely enough) reflects the real-world data. Thus, for the model in Fig. 5A, to execute the Main PE while using the sensed value of the context variable *c*:

- A sensing activity (Sensing PE) must be executed in a BP when *c* is in the available state. Based on Fig. 5A, this must happen after the events TDfn and TAvl have occurred, but before TUnvl. We call the resulting window the *sensing window (SW)* for the context variable.
- The Main PE must be executed when the *c*'s SensedValue is in the Valid state: i.e., after TDfn and TAvl occurred and the Sensing PE has been executed, but before the *c*'s ActualValue has changed. This defines the CV's *validity window (VW)*.

Note that if a context variable is always available for sensing and its value is not expected to change, then developing statecharts for it is not necessary since the only constraint is that its value must be sensed before the Main PE is executed.

To illustrate the above approach, we now present a number of example statecharts. First, we look at the lifecycle of Trip.Distance (Fig. 5C), which captures the total distance of passenger trip. The actual value of this variable is not available until the trip route is known. For taxi and similar trips, where the exact route is not known until the end of the trip, the Trip Route Known transition will be triggered when the trip is over. Thus, in the case of trips with routes that are unknown upfront, any PE that utilizes the trip distance information (e.g., Calculate Fare) can only be executed starting from the end of the trip. However, for trips with a fixed route (e.g., on railroads) the transition will be triggered even before the trip begins, thus giving a process designer much more flexibility in terms of the positioning of the fare calculation activity.

Another example (Fig. 6A) shows the lifecycle of the customer payment (e.g., credit card) information in the context of a taxi trip. Here, the actual value of the information is not defined until the customer is known. Unlike ride-sharing companies, taxis are not able to store their customers' payment information: only when a customer is actually in a cab can this information be sensed. Then, once a customer exits the vehicle, his payment information is no longer available. On the SensedValue side, the payment information sensing happens when the payment is made through a credit card reader. This can only happen when the information is available, i.e., while the customer is in the car.

As already discussed, CVs are the sources of constraints on the Sensing PE and Main PE. However, in addition to the optional context parameters, there can be constraints due to the goal's mandatory parameters. These are identified in a similar fashion. Extra constraints on the Main PE may also be present (as exemplified by the need to charge taxi customers before they exit their cabs). We use the CV and its SW(s) to derive the constraints on the placement of the corresponding Sensing PE in a BP. Both the variable's SWs and VWs are then used to identify the *Main PE Window (MW)* – the window for positioning the activity used to achieve the objective under consideration.

Fig. 6B illustrates the positioning windows that the payment information lifecycle imposes on the sensing activity (Sense Through Reader) and the Main PE (Charge Customer). The diagram also shows the corresponding SW and VW.

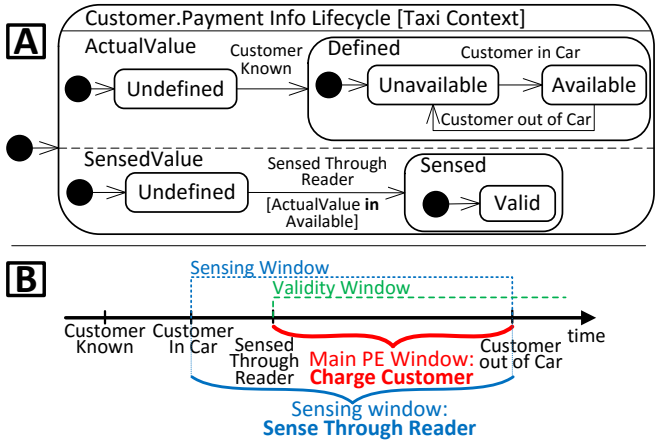


Fig. 6. A) The lifecycle of Customer.Payment Info in the taxi context. B) A timeline diagram showing constraints on sensing payment information and on Charge Customer.

### B. *Combining Constraints from Multiple Variables*

In the previous section, we looked at how the Sensing and Main PE windows are derived for a CV. In general, a number of such variables can be utilized for achieving a goal, with each variable *c* producing two windows, $SW_c$ and $MW_c$, or *two sets* of such windows if the information is not always available for sensing (see below for examples). Each $SW_c$ is used to constrain the positioning of the Sensing PE for *c*. Generally, these sensing activities are independent for each CV. So, for N variables, we will have N Sensing PEs placed somewhere within their corresponding $SW_c$ windows. On the other hand, every $MW_c$ window refers to the *same* Main PE, which means that to determine the overall constrains on the Main PE we need to combine the $MW_c$ windows from all the used variables. The outcome of such integration is the window MW (note the absence of the subscript).

When integrating constraints from multiple CVs, we can do it in two ways. One is to produce a window MW, which represents the intersection of all the $MW_c$ windows. The other is to produce a single position for executing the Main PE, which is compatible with all the constraints. Both options lead to the correct positioning of the Main PE. The benefit of a window is the added flexibility. We can use further analysis at a later time to determine the best place for the Main PE within that window –

e.g., whether to postpone or advance its execution. This provides additional benefits and allows for fine-tuning the position based on important NFRs [11]. Note that the position of the Main PE within the window can change periodically if it is deemed that the BP needs to be reconfigured. While producing the MW instead of a single position adds some extra complexity to the approach, we believe the added flexibility justifies it. Thus, below we focus on how to obtain the integrated window MW.

As the MW is the intersection of $MW_c$ from all the variables, it is possible that not all such windows actually intersect. This creates a conflict and implies that there is no consistent way to position the Main PE within a BP while utilizing all the selected CVs. The solution is to drop one or more CVs from consideration so that the windows from the remaining CVs intersect.

Since the selected CVs can be ordered based on their net benefits, the primary conflict resolution strategy would involve dropping one or more conflicting CVs to maximize the benefits provided by the remaining ones. Options may include, for instance, dropping several lower-impact variables or one higher-impact one. This is a common optimization problem.

Since we would also like to support flexibility in BP design, the secondary objective of the conflict resolution process may be to maximize the size of MW. This would lead to the CVs with the more restrictive $MW_c$ windows being removed first.

Recall that the $SW_c$ windows are defined based on the availability (for sensing) of context information and each $MW_c$ is defined based on the validity of the sensed information. Since the information cannot be valid before it has been sensed and the Main PE can only be executed while the sensed CV value is valid, the beginning of each $MW_c$ *always* corresponds to the position of the Sensing PE for the context *c*. Hence, the position of the Sensing PE in $SW_c$ directly impacts $MW_c$. So, when choosing the best place to execute our Main PE, we also have to consider when to sense the required information: the larger the $MW_c$ windows are for each context *c* the more flexibility we will have when integrating them to produce MW. One strategy to address this is to sense context variables as soon as they become available for sensing, thus moving the starting points of the $MW_c$ windows as much to the left as possible. It is a good heuristic when the sensed information is stable (at least in the time frame relevant to the process, as is the case with the credit card information). However, the dynamics of context is also important. E.g., if we want to plan a route based on traffic conditions, we cannot sense this information arbitrarily early because of its volatility and hence the limited validity of the corresponding sensed value. This is illustrated later in the section.

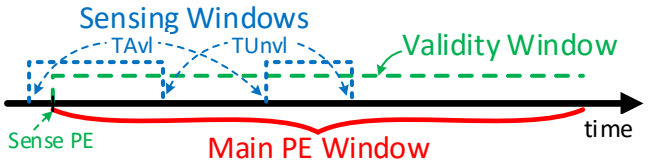


Fig. 7. Intermittenlty available, but stable context variable.

Similarly, intermittent availability of context information is an important factor when sensing variables (e.g., think of the data from a wireless sensor on a truck that occasionally travels through tunnels). In this case, the context variable will have many sensing windows (see Fig. 7). These windows are defined by the TAvl/TUnvl triggers, as previously discussed. Spotty availability of context data coupled with stable data values is not a serious complication as the context can simply be sensed in any of the SWs. As discussed above, sensing early potentially increases the size of MW, which helps flexibility.

Let us now look at an example of how the constraints coming from several CVs can be integrated. In particular, this example focuses on the volatility of context data. We look at the Plan Route goal for a ride-hailing service. To achieve it we need to know at least the trip's origin and destination. Yet, there are many other pieces of information that can potentially improve the route. We assume that in the context identification process (Sec. II) we identified four CVs for this goal (see TABLE I. ).

TABLE I. VARIABLES IN THE PLAN ROUTE EXAMPLE.

| Variable | Volatility | Availability |
|---|---|---|
| **Weather Info** | Continuous change. 30 min to significant change | Always available |
| **Traffic Info** | Continuous change. 10 min to significant change | Always available |
| **Driver Prefs** | Low (constant with respect to the current process) | Available from Driver Known |
| **Customer Prefs** | Low (constant with respect to the current process) | Available from Customer Known |

Weather and traffic information is, obviously, useful when planning a route. Similarly, driver and customer preferences (Driver Prefs and Customer Prefs respectively) can also be integrated to produce a better route that can achieve additional quality objectives relevant to either actor (e.g., the route being *scenic* for customers). Once the customer's identity is known, his preferences can be sensed; the same is true for the driver (and we also assume that the driver is known before the customer is). Weather and traffic conditions change constantly. We assume that the sensed information becomes obsolete (i.e., invalid) after 30 min for the weather and after 10 min for the traffic (which is more volatile). These assumptions may be different based on many factors – e.g., the general climate and typical weather patterns in the area, the current date/time characteristics (day of the week, holiday period), etc.

Fig. 8A illustrates the lifecycle of Traffic Info (Weather Info has a similar lifecycle). The model shows that the variable is always available for sensing. Also, significant changes happen every 10 minutes and sensing must be done at the same rate to keep the sensed value valid. The statecharts for Driver Prefs and Customer Prefs are rather simple and are not shown.

Once the constraints are identified for each variable, they are integrated to produce the overall constraint on Plan Route. To better understand how the constraints from various CVs are related, an appropriate timeline diagram can be developed. Fig. 8B shows the one for our Plan Route example. Here, we capture the SWs for all the variables (the weather and traffic info are always available) and for the volatile ones (traffic/weather) we also model their VWs, which in this example are defined relative to the position of the Plan Route PE since this data has to be used within a limited amount of time after it has been sensed.

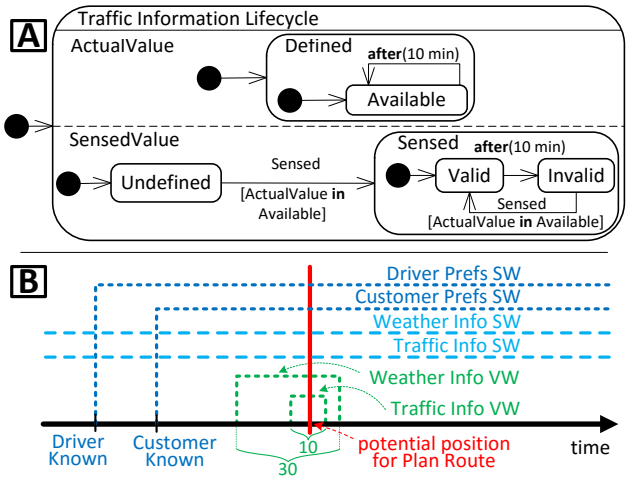


Fig. 8. A) Traffic information context variable's lifecycle. B) Bringing together constraints on Plan Route. The variables' Sensing Windows (SW) and Validity Windows (VW) are shown.

We then use inequalities on time to capture these constraints. We look at the SWs, which define constrains on sensing actions (Sense(variable)), and at how the positioning of Plan Route is related to these sensing PEs. We assume that events are instantaneous and that t(Event) captures the time of the event. Actions (including sensing ones) take time, so t_s(Action) and t_f(Action) represent an action's start and finish times respectively.

1) t(Customer Known) < t_s(Sense(Customer Prefs)) < t_f(Sense(Customer Prefs)) < t_s(Plan Route)
2) t(Driver Known) < t_s(Sense(Driver Prefs)) < t_f(Sense(Driver Prefs)) < t_s(Plan Route)
3) 0 < t_s(Plan Route) – t_f(Sense(Traffic)) < 10
4) 0 < t_s(Plan Route) – t_f(Sense(Weather)) < 30

The inequalities 1) and 2) combine the constraints on both the sensing PE and Plan Route and define the lower bound on the timing of the latter action, while 3) and 4) state that Plan Route has to start executing within 10 and 30 minutes *after* sensing traffic and weather, respectively. Note that the upper bound on the MW for Plan Route does not depend on CVs and will be determined based on the existing functional dependencies among business objectives and/or PEs in the business process.

Now that we have seen an example of a volatile context information sensing, it is easy to see that if, for some CV *c*, such volatility is coupled with an intermittent availability, it creates even more complex conditions for sensing. Here, not only do we need to sense context variables close enough to the Main PE so that the sensed data is still valid when the Main PE is executed, but we also need to make sure the context sensing is done within one of the validity windows of *c*. Note that it is possible that no compatible window exists. In this case, *c* or some other context(s) may be dropped from consideration, as previously discussed. We do not illustrate this case due to the lack of space.

For another illustration, we go back to the customer payment example from Fig. 6A to see how Customer.Payment Info constraints affect the positioning of the Charge Customer PE. Unlike the previous example, the inequalities below define both upper and lower bounds for the Charge Customer's MW:

5) t(Customer Known) < t_s(Sense Through Reader)
6) t(Customer In Car) < t_s(Sense Through Reader) < t_f(Sense Through Reader) < t(Customer Out Of Car)
7) t_f(Sense Through Reader) < t_s(Charge Customer) < t_f(Charge Customer) < t(Customer Out Of Car)

A SW for a context variable may depend on the means of sensing it. So, if relevant, a particular instance of a sensing PE should be specified. Here, we used Sense Through Reader instead of a generic sensing PE Sense(Payment Info).

### C. *Using the Identified Constraints in Process Design*

For an objective G being explored using the approach (e.g., Calculate Fare in Fig. 1A) and starting with a base BP, such as the one in Fig. 1B, we use the timeline diagrams and inequalities presented earlier to help position the Sensing PEs (one for each selected CV) and the Main PE in that BP in a way that makes sure that all the data availability/validity constraints are satisfied. Note that the events featured in these diagrams/inequalities (e.g., Customer Out of Car in Fig. 6B) are matched with the activities that have those events as postconditions – e.g., Customer Out of Car is the result of executing Unload Customer.

The constraints identified here are over and above the functional and data dependency constraints that exist in BPs. By focusing on windows (not concrete positions) and by using inequalities, we aim at producing the least restrictive constraints on the positioning of the Sensing and Main PEs. This gives process designers flexibility when creating process specifications.

## IV. Related Work

While a business process is generally seen as a collection of activities that achieves some business purpose or objective aiming to create value for customers, most approaches to BP modeling work at the workflow level and focus on activities, flows, etc. [14]. Still, explicitly captured business goals have been used in BP modeling for a long time [7][8][9]. These approaches help take business objectives, systematically refine them using various goal-based modeling notations, and then map the resulting leaf-level (operational) goals into BP elements – activities or, sometimes, subprocesses. Goal models developed in this way support elicitation and analysis of goal refinement alternatives and provide rationale for workflow-level activities. These approaches are complementary to ours. We use the method from [9] to refine high-level business goals into operational objectives and then apply our approach to selected leaf-level goals.

There are approaches that support adding certain types of constraints on relationships among goals to goal models. E.g., in [9], control flow annotations are heavily used in goal models and then mapped into BPEL constructs. In [12], preconditions are graphically represented in goal models. These approaches are lighter-weight and less powerful than the declarative BP design methods [1], but have the advantage of using a similar notation to our approach, which makes them easy to integrate.

Approaches for designing context-aware BPs (e.g., [5][16]) focus on capturing and analyzing the effects of domain properties (contexts) on process behaviour, which is done at design time. Once designed, context-aware BPs are generally capable of sensing context information at runtime and thus a runtime BP

infrastructure is needed to support changing processes behaviour. This is frequently done through the use and management of process fragments (e.g., as in the Provop approach [5]). On the other hand, our method, given an operational goal, proactively seeks context variables that could be sensed and utilized to achieve this goal better. This analysis is done at design time. While in our approach CVs are sensed at runtime and then utilized for achieving business objectives, the process itself does not change. What changes is the outcome of process elements that have CVs as their inputs – e.g., routes will be calculated differently given the changing traffic/weather conditions. Thus, the contextual variability in our approach is hidden within process activities. Therefore, we view our method as complementary to the mainstream context-aware BP approaches.

## V. Conclusion and Future Work

The growing complexity, dynamism, and competitiveness of today's business environments coupled with the increasing availability of data and the sensing and analytics technologies enables the shift from the reactive context-awareness that sensed and analyzed a few important context variables, which had a significant impact on organizations' BPs, to its proactive counterpart where organizations seek to take as much contextual information into consideration as possible, provided the sensed and utilized contextual variables deliver increased business value.

Unlike the previous approaches for designing context-aware BPs, which focused on specifying how organizations and their processes would react to changing business contexts, this paper describes the approach for proactively seeking relevant context information to improve the achievement of operational business goals. Given such an objective, we explore the space of potentially relevant context variables that can be sensed and then used for achieving it. We outline the process of how the costs and benefits of each CV can be evaluated and how the set of best-performing CVs is selected for implementation. Based on these CVs and their lifecycles we identify the constraints on when these variables must be sensed and where the PE achieving the given business goal is to be executed in a BP.

In its current form, the approach provides some guidance on the elicitation of various types of NFRs (e.g., as described in Sec. II.C and elsewhere in the paper). We are working on improving upon this and plan to provide detailed patterns for linking various NFRs used in the evaluation of sensing and implementation options discussed in Sec. II.

We aim at applying this approach to BP architectures (BPAs) [11] to analyze the situations where, e.g., context sensing and the use of the sensed information are performed in different *stages* (i.e., processes with distinct execution cycles). This points to the need to identify additional data-related PEs, such as those for data aggregation, storage, etc., and to more complex positioning constraints.

Similarly, as the approach relies heavily on goal model-based analysis, we are looking at automating this portion of the approach – e.g., by using existing goal modeling tools.

In the proposed method, we use some lightweight formalization (e.g., when specifying lifecycle statechart triggers) and generate simple, yet useful constraints. Declarative (e.g., Declare [1]) and hybrid [3] BP modeling approaches have much more powerful languages that can also be used in our approach for constraint formulation and eventually for BP design automation. We are currently exploring these options.

A limitation of the approach is that it only applies to leaf-level goals and assumes that they are always implemented by a single PE. We aim at coping up with a generalized version of the method where these constraints are removed.

The approach is currently being evaluated in the domains of transportation (portions of this case study were used as examples in this paper), retail, and DevOps.

## References


[1] W. van der Aalst, M. Pesic, H. Schonenberg. Declarative workflows: Balancing between flexibility and support. Computer Science – Research and Development 23(2):99-113, 2009.

[2] R. Ali, F. Dalpiaz, P. Giorgini. A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering, 15(4):439-458, 2010.

[3] P. Chen. The Entity-Relationship Model - toward a unified view of data. ACM Transsactions on Database Systems, 1(1): 9-36, 1976.

[4] G. De Giacomo, M. Dumas, F. Maggi, M. Montali. Declarative Process Modeling in BPMN. In Proc. CAiSE 2015, Stockholm, Sweden, 2015.

[5] A. Hallerbach, T. Bauer, 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.

[6] J. Horkoff, E. Yu. Comparison and evaluation of goal-oriented satisfaction analysis techniques. Requirements Engineering, 18(3):199-222, 2013.

[7] V. Kavakli, P. Loucopoulos. Goal-Driven Business Process Analysis Application in Electricity Deregulation. Information Systems, 24(3):187-207, 1999.

[8] R. Kazhamiakin, M. Pistore, M. Roveri. A Framework for integrating business processes and business requirements. In Proc. EDOC 2004, Monterey, USA, 2004

[9] A. Lapouchnian, Y. Yu, J. Mylopoulos. Requirements-driven design and configuration management of business processes. In Proc. BPM 2007, Brisbane, Australia, Sep 24-28, 2007.

[10] A. Lapouchnian and J. Mylopoulos. Modeling domain variability in requirements engineering with contexts. In Proc. ER 2009, Gramado, Brazil, Nov 9-12, 2009.

[11] A. Lapouchnian, E. Yu, A. Sturm. Re-Designing process architectures: towards a framework of design dimensions. In Proc. RCIS 2015, Athens, Greece, May 13-15, 2015.

[12] S. Liaskos, A. Lapouchnian, Y. Yu, E. Yu, J. Mylopoulos. On Goal-based Variability Acquisition and Analysis. In Proc. RE 2006, Minneapolis, USA, Sep 11-15, 2006.

[13] J. March. Exploration and exploitation in organizational learning. Organization Science, 2:71-87, 1991.

[14] Object Management Group. Business Process Model and Notation (BPMN), Version 2.0.2, 2013. Retrieved April 29, 2016 from www.omg.org/spec/BPMN/2.0.2/PDF.

[15] Object Management Group. Unified Modeling Language (UML), Version 2.5, 2015. Retrieved April 29, 2016 from www.omg.org/spec/UML/2.5.

[16] M. Rosemann and J. Recker. Context-aware process design: exploring the extrinsic drivers for process flexibility. In Proc. BPMDS 2006, Luxembourg, 2006.

[17] M. Rosemann. Proposals for future BPM research directions. In Proc. AP-BPM 2014, Brisbane, Australia, July 3-4, 2014.