Enabling Flexible Location-Aware Business Process Modeling and Execution

Xinwei Zhu[a,*], Seppe vanden Broucke[b], Guobin Zhu[a], Jan Vanthienen[b], Bart Baesens[b,c]

[a] *International School of Software, Wuhan University*
Luoyu Road 37, Hongshan, Wuhan, Hubei, China
[b] *Research Center for Management Informatics (LIRIS), KU Leuven*
Naamsestraat 69, B-3000, Leuven, Belgium
[c] *School of Management, University of Southampton*
Highfield Southampton, SO17 1BJ, United Kingdom

Abstract

Business Process Management (BPM) has emerged as one of the abiding systematic management approaches in order to design, execute and govern organizational business processes. Traditionally, most attention within the BPM community has been given to studying control-flow aspects, without taking other contextual aspects into account. This paper contributes to the existing body of work by focusing on the particular context of geospatial information. We argue that explicitly taking this context into consideration in the modeling and execution of business processes can contribute to improve their effectiveness and efficiency. As such, the goal of this paper is to make the modeling and execution aspects of BPM location-aware, i.e. to govern and constrain control-flow and process behavior based on location-based constraints. We do so by proposing a Petri net modeling extension

[*] Corresponding author
Email addresses: `xinwei.zhu@whu.edu.cn` (Xinwei Zhu), `seppe.vandenbroucke@kuleuven.be` (Seppe vanden Broucke), `gbzhu@whu.edu.cn` (Guobin Zhu), `jan.vanthienen@kuleuven.be` (Jan Vanthienen), `bart.baesens@kuleuven.be` (Bart Baesens)

which is formalized by means of a mapping to coloured Petri nets (CPN). Our approach has been implemented using CPN Tools and a simulation extension was developed to support the execution of location-aware process models. We also illustrate the feasibility of coupling business process support systems with geographic information systems by means of an experimental case.

Keywords: business process management, geographic information systems, location-aware processes, geospatial processes, process modeling, process execution, coloured petri nets

1. Introduction

Throughout the past two decades, Business Process Management (BPM) has emerged as one of the abiding systematic management approaches to align organizational business processes to the needs of clients (Vom Brocke and Rosemann, 2010). BPM encompasses a broad scope, including the design, modeling, execution, monitoring and optimization of business processes—the so-called BPM life cycle (van der Aalst et al., 2003). The main driving rationale for BPM is that it enables organizations to be more efficient and more capable to react to changes. From this viewpoint, BPM regards processes as core strategic assets of an organization, which hence need to be understood, managed, and improved to increase the value added by products or services delivered to clients.

The emergence of BPM has caused a shift in the realm of information systems and information technology from data-based information systems to process-aware ones, i.e. "Process-Aware Information Systems", or PAIS. The support provided by PAIS—be it for the modeling, execution, validation or monitoring of business processes—is only able to capture and describe an idealized or simplified version of reality. Traditionally, most attention within the BPM community has been focused on studying control-flow aspects of business processes, i.e. the aspects governing the flow of business activities (i.e. the sequence in which activities can be performed). In recent years, however, integrating other perspectives and "contexts" within this view has received increased attention, as support systems which adopt a control-flow-centric view are unable to adequately capture human behavior due to lack of descriptions of possible constraints against activity modeling. Similarly, support systems focusing only on data aspects fail to capture the flow and sequence aspects of the data as it moves through a business process. As such, many scholars have shifted towards studying various approaches that integrate control-flow with other contexts. In this paradigm, processes can be rapidly changed and adapted to a new external data-governed context (e.g., location, weather, etc.). It is recognized that contextualizing processes in this manner allows for a more explicit consideration of the environmental setting of a process (Rosemann et al., 2008).

This paper contributes to the research field of BPM by focusing on the particular *context of geospatial information*, an aspect which is becoming more and more important in all information system related areas, given the increased usage of mobile devices and tracking as well as other recent developments such as the Internet of Things or sensor-based data gathering. We hence argue that taking this context into account in the various life cycle steps of BPM can contribute to improve the effectiveness and efficiency of process management. Especially in environments where a need arises to apply both process-aware and Geographic Information Systems (GIS), it is highly valuable to combine and integrate these two perspectives, instead of considering them in isolation (Meeks and Dasgupta, 2004). The goal of this paper is thus to make the modeling and execution aspects of BPM "location-aware". We do so by proposing business process modeling language based on a formal Petri net extension which incorporates location aspects and ways to constrain the execution of activities by location-based constraints. Next, we formalize the execution semantics of our extension by describing a unambiguous mapping to coloured Petri nets. This also allows us to develop a prototype implementation of our approach using CPN Tools (Jensen et al., 2007), with which a simulation extension was developed to support the execution and validation of models created using our approach and to illustrate the feasibility of coupling business process support systems with geographic information systems.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work and preliminaries used throughout the paper. Section 3 outlines a running example which will be used to illustrate the developed artifacts. Section 4 introduces our proposed modeling language to design location-aware processes, after which Section 5 discusses the execution semantics of such models by means of a mapping to coloured Petri nets. Section 6 discusses the developed implementation. Section 7 concludes the paper and provides outlines for future work.

2. Preliminaries

2.1. Related Work

We regard location as one of the key variables in the wider context of a business process. In the layered process context model proposed by Rosemann et al. (Rosemann et al., 2008), location describes an important variable situated in the environmental context layer, which describes process-related variables that reside beyond the business network in which an organization is embedded, but still pose a contingency effect on the business processes. Scholars have argued that the inclusion of location contextual variables in business process management practices help to improve dependency aspects (constraining activity executions based on location aspects, for instance) (Decker, 2009), increase the adaptability and flexibility of running processes (by reconfiguring and modifying models and tasks based on location aspects) (Georgoulias et al., 2009; Chakraborty and Lei, 2004; Ali et al., 2006; Aoumeur et al., 2004), and improve the efficiency (performance and cost-effectiveness) of organizational processes (Zhu et al., 2014). Naturally, these concern are of an even greater importance for processes where mobility (that is, tracking changes in locations and adapting processes to these changes) is deemed to be an important factor (Jing et al., 2000).

The notion of location-awareness centers around the basic idea that location and location-based services can be sensed and adapted to within processes. Location-aware business process management thus encompasses the ability for a business process to sense the current process status in a specific location and to be aware of the whole process situation. Based on this, process owners can react or dynamically change the process execution to adapt the goal of the process. Examples of location sensitive services and applications can be observed in areas such as navigation and travel, device and human tracking, geosocial networking, retail and real-estate services, mobile workforce deployment, and many others. However, works around the connection of location services with principles of business process management are relatively scarce in the literature. That is, many researchers focus on on connecting spatial-based information with scientific workflows (Medeiros et al., 1996; Alonso and Hagen, 1997; Günther, 1997; Seffino et al., 1999;

Altintas et al., 2004; Ludäscher et al., 2006; Jäger et al., 2005), but not with business-oriented workflows. As a notable exception, (Leoni et al., 2008) discuss map metaphors that are used to visualize work items and resources in process-aware information systems (using the YAWL workflow language). This technique specifies that users could check geographical positions and distances based on a geographical map, but does not indicate how exactly geographical aspects can influence the flow of execution of the process. Decker et al. (Decker, 2009; Decker et al., 2010) have defined location constraints for individual workflow activities when modeling a workflow schema to restrict the location where an activity can be performed, but the location constraints lack comprehension and expressiveness. Our proposed modeling technique, on the other hand, is able to specify in an exact manner how location impacts the basic logical relationships in a process control-flow, i.e. sequence, parallel split/joins and exclusive choice split/joins.

Some existing BPM tool suites allow for the definition and capture of additional variables in the modeling of business processes (Recker, 2012; Recker et al., 2009; Jing et al., 2000; Ali et al., 2006). In theory, such attribute fields could be used to capture location-based information. For instance, in Business Process Model and Notation (BPMN) models, locations could theoretically be modeled through the use of swimlanes, text annotations or data elements. In Event-driven Process Chain (EPC) models, location variables may be grouped via organizational objects and in Yet Another Workflow Language (YAWL) models, static attributes could be attached to work items as additional text information. However, in all these approaches, location-based elements exist only as secondary constructs or text-based annotations for readers to understand the graphical diagram, and do not impact the semantics or execution of the modeled process in a direct way.

Our approach aims to make location-based constructs first-class citizens in the modeling and execution of process models: meaning that it is possible to govern the execution of a process based on location-based properties, and to signal changes to location properties based on the enactment of activities within a running business process.

2.2. Definitions and Notations

This section outlines preliminary concepts and definitions which will be utilized in the remainder of the paper.

Petri nets are a well-known representational language to model concurrent system, and have also been extensively applied to formalize business process model semantics (Murata, 1989; Peterson, 1981).

Definition 1. Petri net. A Petri net is a triple (P, T, F) (Murata, 1989; Peterson, 1981) with:

– P is a finite set of places, $P = \{p_1, p_2, \ldots, p_{|P|}\}$;

– T is a finite set of transitions, $T = \{t_1, t_2, ..., t_{|T|}\}$, with $P \cap T = \emptyset$;

– $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of directed arcs (flow relation).

A place (drawn as a circle, see for instance Fig. 1) $p \in P$ is called an input/output place of a transition (drawn as a box and labeled; unlabeled transitions are shown as a black box) $t \in T$ if there exists an arc from p to t or from t to p respectively. $\bullet t$ and $t\bullet$ denote the set of input and respectively output places for a transition $t \in T$. Similarly, $\bullet p$ and $p\bullet$ define the set of transitions having $p \in P$ as an input place and the set of transitions having $p \in P$ as an output place respectively.

Definition 2. Marking, marked Petri net. A marked Petri net is a triple (N, M, M_0) with:

– $N = (P, T, F)$ a Petri net (a marked Petri net can also be written in an expanded notation (P, T, F, M, M_0));

– $M : P \to \mathbb{N}_0^+$ is a marking function;

– $M_0 : P \to \mathbb{N}_0^+$ is the initial marking function.

Places $p \in P$ in a Petri net can contain zero or more "tokens" (drawn as black dots inside the places). The distribution of tokens over the places defines the state, denoted as the "marking" of the Petri net, represented by the marking function M, which maps each $p \in P$ to a natural, positive number, representing the amount of tokens contained in that place. The multiplicity of a place p in a marking M, i.e. $M(p)$, denotes the number of tokens that this place contains. The initial marking M_0 is used to initialize all places with an initial token count (in most cases, the initial marking is defined as follows: $M_0 : p \in P \mapsto 1$ if $\bullet p = \emptyset$ or 0 otherwise).

Definition 3. Petri net execution semantics. The number of tokens in a Petri net changes during the execution of a Petri net. The marking of a Petri net defines a state, based on which execution semantics can be formalized as follows:

– A transition $t \in T$ is said to be *enabled* under marking M iff each of its input places contains at least one token: $\forall p \in \bullet t : [M(p) > 0]$;

– An enabled transition $t \in T$ can be fired, which brings the Petri net from one state to another:

$$M_1 \xrightarrow{t} M_2 \text{ so that } \forall p \in P : [\begin{cases} M_2(p) = M_1(p) & \text{if } p \notin \bullet t \cup t\bullet \\ M_2(p) = M_1(p) - 1 & \text{if } p \in \bullet t \\ M_2(p) = M(p) + 1 & \text{if } p \in t\bullet \end{cases}].$$

Scholars modeling control-flow dimensions of a business process often utilize a subclass of Petri nets, called WorkFlow nets (or, WF-nets) (van der

Aalst, 1998). A WF-net specifies the behavior of a single process instance in isolation.

Definition 4. WF-net. A Petri net (P,T,F) is a WF-net (van der Aalst, 1998) when:

– There is a single source place $i \in P$ such that $\bullet i = \emptyset$;

– There is a single sink place $o \in P$ such that $o \bullet = \emptyset$;

– The net $(P, T \cup \{t'\}, F \cup \{(o,t'),(t',i)\})$ is strongly connected, i.e. every $x \in P \cup T$ lies on a path from i to o .

To define the execution semantics of our Petri net modeling extension, we will provide a formalized mapping to coloured Petri nets (CPN). CPNs are an extension of Petri net which are comparable to Petri nets, but add color sets to places and tokens to allow for tokens of multiple types, guard transitions to constrain the execution of transitions and arc expression to govern input and output flow of tokens (Jensen, 1987). Normally, guards and arc expressions are formalized in a particular language (CPN Tools for instances uses Standard ML to do so, a functional programming language). The definition we provide below is adapted from (Jensen, 1987), but defined in language-agnostic form, meaning that we assume the general availability of a language which allows to define expressions which can be evaluated and yield a result when done so.

Definition 5. Coloured Petri net. A CPN is a tuple $(P, T, A, \Sigma, C, V, N, G, E, M, I)$ with:

– P the set of places, $P = \{p_1, p_2, ..., p_{|P|}\}$;

– T the set of transitions, $T = \{t_1, t_2, ..., t_{|T|}\}, P \cap T = \emptyset$;

– A the set of arcs, $A = \{a_1, a_2, ..., a_{|A|}\}$;

– Σ the set of color sets defined within the model. A color set is a grouping of colors. A color is an attached value (i.e., a label) to a token. Regular Petri nets can be expressed as a CPN by defining a single color set with a single color value;

– V the set of variables used in the model, $V = \{v_1, v_2, \ldots, v_{|V|}\}$. Note that we indicate the actual value (i.e., color) of a variable $v \in V$ as v^*;

– $C : P \cup V \cup A \rightarrow \Sigma$ the function returning the color set associated to a place, variable, or arc in the CPN model; for arcs, the color set of the place associated to the arc is returned, i.e. $C : a \in A \mapsto C(P \cap N(a))$

– $N : A \rightarrow P \times T \cup T \times P$ the node function mapping arcs to a place-transition or transition-place flow expression. This function allows for the definition of multiple arcs between the same place-transition or transition-place pair;

– $G : t \in T \mapsto \mathit{GExpr}$ the guard expression function mapping a transition $t \in T$ to a boolean expression GExpr, denoting whether the transitions is permitted to fire. Evaluating this expression yields a boolean result value, indicated as $\mathit{GExpr}^* \in \{\mathit{true}, \mathit{false}\}$;

– $E : a \in A \mapsto \mathit{AExpr}$ the arc expression function mapping an arc $a \in A$ to an expression AExpr. Evaluating an arc expression yields a multiset of tokens, indicated as AExpr^*_{MS} which is to be produced (for transition to place arcs) or consumed (for place to transition arcs). The expression itself can use one or multiple variables in V, the color sets of the input and outputs of the arc expression must correspond to the color sets of the places the arcs connects to, formalized, the following holds: $\forall a \in A : [\exists \sigma \in \Sigma : [\forall \tau \in E(A)^*_{MS} : [\tau \in \sigma] \wedge C(P \cap N(a)) = \sigma]]$;

– $M : p \in P \mapsto C(p)_{MS}$ the marking function, returning the multiset of tokens contained in a place with $\forall p \in P : [\forall \tau \in M(p) : [\tau \in C(p)]]$;

– $I : p \in P \mapsto \mathit{IExpr}$ the initialization function, this function initializes places in the model with a state, expressed as colored tokens. The evaluation of an *IExpr* yields a token multiset, indicated as IExpr^*_{MS} with $\forall p \in P : [\forall \tau \in \mathit{IExpr}^*_{MS} : [\tau \in C(p)]]$.

The execution semantics of a CPN differ from those of a regular Petri net.

Definition 6. Coloured Petri net execution semantics. Let $p : A \to P$ be a function returning the place attached to an arc and t: $A \to T$ a function returning the transition attached to an arc, i.e. $t : a \in A \mapsto T \cap N(a)$. For a transition $t \in T$ to be enabled, the following criteria need to hold:

– All expressions of the incoming arcs should be satisfied: $\forall a \in A, t(a) = t : [E(a)^*_{MS} \neq \emptyset]$;

– The guard condition of the transition must evaluate to true, $G(t)^* = \mathit{true}$.

Enabled transitions can be fired. Output and input places are updated accordingly given the input and output arc expressions. Firing an enabled transition brings a marking $M_1 \xrightarrow{t} M_2$ as follows. Let $A^I_t = \{a \in A | N(a) = (x, y) \wedge y = t\}$; $A^O_t = \{a \in A | N(a) = (x, y) \wedge x = t\}$ and $A_t = A^I_t \cup A^O_t$, then $\forall p \in P : [M_2(p) = M_1(p) \uplus \{\tau \in E(a)^*_{MS} | a \in A^O_t \wedge p(a) = p\} \backslash \{\tau \in E(a)^*_{MS} | a \in A^I_t \wedge p(a) = p\}]$.

Next, we shift our attention to the formalization of locations. The definition of our concept of location corresponds with a so called "feature" as applied by most geographic information systems. A feature describes something that can be drawn on a map, i.e. something in the real world—a mountain, landmark or even moving objects such as cars. Additionally, it is reasonable to group certain features together if they share a number of properties. For example, China, Belgium and Germany can all be regarded as features of the type "Country". In addition, apart from a semantic description (a name and other properties), features can also be represented in physical terms, i.e. as a mathematical expression of an object's geometry in terms of points, lines, paths (multiple line segments) or polygons (associated with a well-defined coordinate reference system and scale or granularity).

Definition 7. Feature, feature type. Our definition of location corresponds with a so called "feature". Features are defined in terms of a geometry, a feature type, and an arbitrary number of additional attributes (such as a human-readable name), i.e.:

– Let FT_L be a set of feature types, $FT_L = \{ft_1, ft_2, ..., ft_{|FT|}\}$;

– Let F_L be a set of features, $F_L = \{f_1, f_2, ..., f_{|L|}\}$;

– Let *Type* : $F_L \rightarrow FT_L$ be a function mapping features to a type. In the visual modeling notation (e.g. in Fig. 2 later below), we will also denote this using the shorthand $f : ft$ with $f \in F_L$ and $ft \in FT_L$;

– Let *Geometry* : $f \in F_L \mapsto g$ a function which returns the geometry g for a given feature $f \in F_L$;

– Let f^a indicate some attribute a associated with a feature $f \in F_L$. This can be any type of data, e.g. a name, a date, a number or even other features.

Many standards and vendor implementations exist to define geometry types (OGC, 1999; Lake et al., 2004; Clementini et al., 1993, 1994). In our implementation, we define a geometry g to be one of the following types, but this can be extended or modified based on end-user needs:

– A "point", a single coordinate: $(x \in \mathbb{R}, y \in \mathbb{R})$

– A set of points, a “multipoint”: $\{(x_1 \in \mathbb{R}, y_1 \in \mathbb{R}), \ldots, (x_n \in \mathbb{R}, y_n \in \mathbb{R})\}$

– A “linestring”, a sequence of points connected by line segments (note that linestrings can form a closed loop or not): $((x_1 \in \mathbb{R}, y_1 \in \mathbb{R}), \ldots, (x_n \in \mathbb{R}, y_n \in \mathbb{R}))$

– A set of linestrings, a “multilinestring”

– A “polygon”, defined by an closed outer boundary (described as a linestring) and zero or more closed inner boundaries: $(((x_1^o \in \mathbb{R}, y_1^o \in \mathbb{R}), \ldots, (x_n^o \in \mathbb{R}, y_n^o \in \mathbb{R})), \{((x_1^{i1} \in \mathbb{R}, y_1^{i1} \in \mathbb{R}), \ldots, (x_n^{i1} \in \mathbb{R}, y_n^{i1} \in \mathbb{R})), \ldots, ((x_1^{i2} \in \mathbb{R}, y_1^{i2} \in \mathbb{R}), \ldots, (x_n^{i2} \in \mathbb{R}, y_n^{i2} \in \mathbb{R}))\})$

– A set of polygons, a “multipolygon”

Finally, we define the concept of a geospatial relationship. By establishing relationships over geometries, we are able to answer queries such as “Is one feature contained in another?” In some cases, geospatial relationships are categorized in separate sets, such as topological, measurement, sequential or complex relationships (Zhu et al., 2013, 2014), but for the sake of simplicity, we define a “geospatial relationship” here by means of a single, global moniker as follows:

Definition 8. Geospatial relationship. A geospatial relationship is a function of the general form $\mathit{Relationship}(fg[, fg_1, \ldots, fg_n][, a_1, \ldots, a_n])$ with:

– *fg*: the geometry used as the main function argument. If a feature is passed in (i.e. $fg \in F_L$), the function is simply applied on *Geometry*(*fg*), the geometry of the feature;

– $fg_1, \ldots, fg_n$: additional geometry or feature arguments (optional). Just as with fg, if a feature is passed in, the function is applied on the geometry of the feature;

– $a_1, \ldots, a_n$: additional arguments of any type other than features or geometry (optional);

– a return type which can be of any type, e.g. another feature, geometry, binary value, number or invariant.

Again, many definitions exist that catalog geospatial relationships. ISO19107 for instance defines a topological model based on the "Geography Markup Language" (Lake et al., 2004), whereas other sources, such as the IntesaGIS project (Amadio et al., 2004) base themselves on an authority database. Many of the most widely used GIS toolkits, i.e. GeoTools, PostGIS, ArcGIS, SQL Server etc. define geospatial relationships in accordance with the "Topic 8" standard proposed by the OpenGIS Consortium (OGC) (OGC, 1999). In (Brodeur et al., 2000), a UML-based metamodel is proposed based on the same standard. In most cases, however, all standards include relations to describe the basic topologic geospatial relations between two geometries as described by de DE-9IM standard (Clementini et al., 1993, 1994; Egenhofer and Franzosa, 1991; Egenhofer and Herring, 1990).

Table 1 provides a listing of all the geospatial relationships we consider in the remainder of our work and were also implemented in the developed prototype, described below, based on the DE-9IM, OGC Topic 8 and ISO19107 standards. These will be used as building blocks towards the construction of location constraints, as will be illustrated later.

Relation	**Return Type**	**Description**
$Centroid(fg)$	Point	Returns the geometric center of a geometry fg.
$Length(fg)$	Length (m)	Returns the length of the geometry fg if it is a linestring or multilinestring.
$Area(fg)$	Area (m^2)	Returns the area of the geometry fg if it is a polygon or a multipolygon.
$Perimeter(fg)$	Length (m)	Return the length measurement of the boundary of fg if it is a polygon or multipolygon.
$ClosestPoint(fg, fg_1)$	Point	Return closest point on or in fg_1 that is closest to fg.
$Distance(fg, fg_1)$	Length (m)	Returns the cartesian minimum distance between fg and fg_1.
$Equals(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometries fg and fg_1 represent the same geometry.
$Disjoint(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometries fg and fg_1 have no points in common.
$Intersects(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometries fg and fg_1 have at least one point in common.
$Touches(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometries fg and fg_1only touch edges and do not overlap in any way.
$Crosses(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometries fg and fg_1 touches and overlap edges.
$Within(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometry $fg_1$is completely within fg (no touching edges).
$Contains(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometry fg contains fg_1.
$Overlaps(fg, fg_1)$	$\{true, false\}$	Returns true if the given geometries fg and fg_1 have points in common but not all points in common.
$Buffer(fg, fg_1, a_1)$	Polygon	Returns a polygon geometry that represents all points whose distance from fg is less than or equal to distance a_1.
$ShortestRoute(fg, fg_1)$	Linestring	Returns a linestring geometry representing the shortest route from fg to fg_1 based on routing topology.

Table 1: Geospatial relationships considered in our approach. We apply these relationships as building blocks towards the construction of location constraints, as will be illustrated later.

3. Running Example

To illustrate our location-aware modeling approach and its execution semantics, we will utilize a running example throughout this paper, extending an example provided in (Decker et al., 2010). Looping and parallel behavior was added to show how our approach can be applied on more complex control-flow constructs. The basic process model (no location-awareness) is depicted as a WF-net in Fig. 1. The example process describes a technical maintenance service, which is executed as follows. The process is started once a customer call is received (Receive Customer Call, RCC) and handled in a particular call center (Accept Customer Call, ACC). The call center evaluates the complaint of the customer, based on which the user is remotely assisted for trivial problems (Remote Assist, RAS), or an inspector is dispatched from the call center to the customer's location to investigate the problem on-site (Dispatch, DIS). Based on the results of the investigation (On-site Inspection, OSI), the inspector can solve the problem whilst investigating (No Repair Required, NRR), or calls-in a mobile repair team to perform on-site repair work (Call Repair Team, CRT and On-site Work, OSW). If the repair cannot be performed on-site (e.g. due to some broken components), the repair team heads to a repair shop to perform repairs there (Shop-floor Repair, SFR), before returning to the customer and continuing the on-site work (this cycle can occur multiple times before the problem is fixed). After finishing the work, both repair team and inspector are released and sent back (Release Repair Team, RRT and Release Inspector, RIN). Next, in case an inspector was sent out, they need to write up a report at the call center (Write Report, WRE). In case a repair was performed, a follow-up check is required as well (Perform Follow-up Check, PFC). Finally, independent of the nature of the solution offered, some administrative follow-up work (Follow-up Administration, FUA) needs to be performed to close the case.

The description of this simple process highlights many locational aspects which cannot be captured by control-flow alone (see Fig. 1). In particular, we list the following locational concerns which the process needs to adhere to:

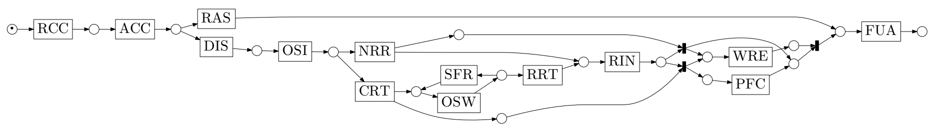

Figure 1: WF-net model of the "repair" process used as a running example throughout the paper. In its basic form, the process model is unable to explicitly include location based concerns in its design nor the take them into account during its execution.

- Call centers may only handle customer calls when the customer is located within the region a call center is responsible for;
- Inspectors can only be dispatched when they are available, and only when their "home base" call center matches the call center that handles the customer call;
- Inspectors can only write their report when they are at their call center;
- Requests can only be made to repair teams which are located in the customer's region or 50km around it. Naturally, a repair team which is already working for another customer cannot be requested;
- The same holds for repair teams performing a follow-up check, but a repair team different from the original one needs to perform the check
- Shop-floor repairs should be made in the repair station closest to the customer's location; this should be based on navigational routing information;
- The call center performing the follow-up work should be situated in a different region then the call center handling the customer call.

4. Location-Aware Process Modeling

This section discusses our proposed modeling approach to model location-aware processes. Our methodology is based on an extension of Petri nets, incorporating two new constructs, namely *location dependent transitions* and *location constraints*.

Fig. 2 shows the running example modeled using our location-aware extension. Using this modeling method, transitions can be made *location dependent* (indicated visually with a flag, ⚑), which means that a feature, belonging to a specified feature type, will be bound to the transition after executing. Modelers are free to decide which transitions should be made location dependent, but in general, the follow guidelines hold:

– Transitions which involve a location that should be bound by constraints should be made location dependent, so as to define location constraints over them.

– Transitions which involve a location that will be used in a following activity (i.e. to constrain it) should be made location dependent.

– Transitions of which their execution involves a change in the properties of a geospatial construct should be made location dependent, so as to track their bound feature.

Next, the shaded boxes in Fig. 2 represent *location constraints*, used to constrain the locations which can be bound to a location dependent transition, or to constrain the execution of non-location dependent transitions based on previously bound locations, without binding a location to those transitions themselves. Visually, the constraints are connected (using dashed arcs) with all the transitions of which their bound location will be used as

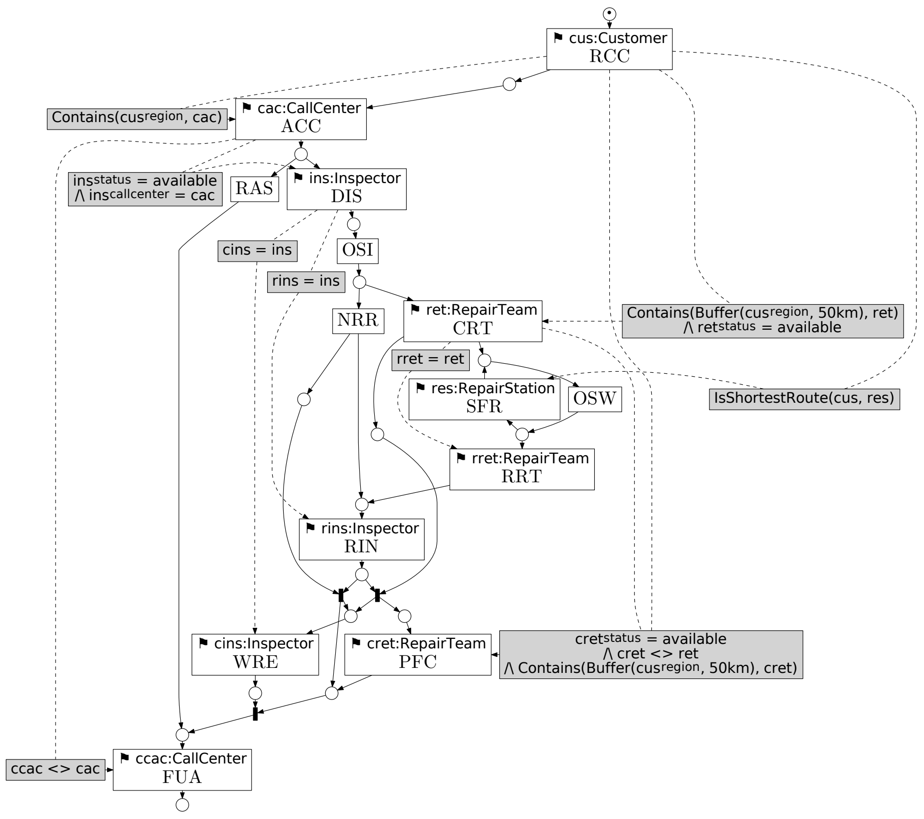

Figure 2: The running example of Fig. 1 remodeled using our location-aware approach. Location dependent transitions are shown with a flag, whilst location constraints are modeled as shaded boxes.

an input in the constraint, and with the one transition which is bounded by the constraint.

The following definitions formalise the constructs of location dependent transitions and location contraints.

Definition 9. Location-aware WF-net (LAWF-net). A location-aware WF-net (LAWF-net) is represented as a tuple $(P, T, F, F_L, FT_L, T_L, C_L, CF_L)$ with:

– P, T and F unchanged with regards to the definition of a WF-net (places, transitions and flows);

– F_L and FT_L the sets of features and their type (see Subsection 2.2);

– $T_L \subseteq T$ the set of location dependent transitions (shown visually with a flag (⚑) and a $f : ft$ label with $f \in F_L$ and $ft \in FT_L$);

– C_L the set of location constraints (a set of expressions, shown visually as a shaded box);

– $CF_L \subseteq (T_L \times C_L) \cup (C_L \times T)$ a finite set of directed arcs linking location dependent transitions to a constraint (shown visually as a dotted arc).

We also overload the function $Type : T_L \rightarrow FT_L$ introduced in Definition 7 to return the feature type for a location dependent transition and the function $M_L : T_L \rightarrow F$ to get the feature bound to a particular location dependent transition (the "location marking") so that $\forall t \in T_L : [Type(M_L(t)) = Type(t) \vee M_L(t) = \emptyset]$. Initially, that is before execution of any transition, no locations are bound, i.e. $\forall t \in T_L : [M_L(t) = \emptyset]$.

Definition 10. Location constraint. A location constraint is an expression $c \in C_L$ so that:

– The constraint expression c evaluates to a boolean result, depicted as $c^* \in \{true, false\}$;

– The constraint expression c involves exactly one output transition, i.e. $\forall c \in C_L : [\exists!(x, y) \in CF_L : [x = c \wedge y \in T]]$. $C_L^t = \{c \in C_L | (c, t) \in CF_L\}$ is used as a shorthand to return all constraints defined on $t \in T$;

– The constraint expression c can involve zero or more location dependent input transitions $t_i^{input} \in T_L$. The feature bound to such input transitions, given by $M_L(t_i^{input})$ will be used as an input for the expression at the time of evaluation. Note that it is possible, in theory, to use a location dependent transition as an input for the same transition.

Note that location constraints can be formulated directly in the form of a geospatial relationship (when the relationship returns a boolean result), though Fig. 2 contains one additional constraint of the form *IsShortestRoute* which is defined as follows: $\mathit{IsShortestRoute} : \mathit{fg} \in F_L \times \mathit{fg}_1 \in F_L \mapsto \mathit{Length}(\mathit{ShortestRoute}(\mathit{fg}, \mathit{fg}_1) = \underset{\substack{\mathit{fg}_2 \in F_L | \\ \mathit{Type}(\mathit{fg}_2) = \mathit{Type}(\mathit{fg}_1)}}{\mathit{min}} \mathit{Length}(\mathit{ShortestRoute}(\mathit{fg}, \mathit{fg}_2)$, i.e. this contraint return true when feature fg_1 is the feature which lies on the shortest path to fg for all features of its type $\mathit{Type}(\mathit{fg}_1)$. Naturally, one can also define a similar constraint for the shortest cartesian distance, $\mathit{IsShortestDistance} : \mathit{fg} \in F_L \times \mathit{fg}_1 \in F_L \mapsto \mathit{Distance}(\mathit{fg}, \mathit{fg}_1) = \underset{\substack{\mathit{fg}_2 \in F_L | \\ \mathit{Type}(\mathit{fg}_2) = \mathit{Type}(\mathit{fg}_1)}}{\mathit{min}} \mathit{Distance}(\mathit{fg}, \mathit{fg}_2)$. Fig. 2 also merges locations constraints pertaining to the same output transition as a conjunction ($\wedge$) to keep the figure readable, but modelers are free to split these up into multiple, separate location constraints. To provide an additional example, in Fig. 2, the constraint "$\mathit{Contains}(\mathit{cus}^{\mathit{region}}, \mathit{cac})$" contains one output transition (ACC) and one input transition (RCC). Instead of using the full transition labels in the expression (i.e., writing "$M_L(\mathit{RCC})$"), we use a short name ("*cus*" or "*cac*") as a way to indicate bound features for location dependent transitions directly. As we have defined a feature type for each location dependent transition, this means that, even when no constraints are modeled, an intrinsic "$\mathit{Type}(M_L(t)) = \mathit{Type}(t)$" constraint is present for any $t \in T_L$.

Although the primary goal of LAWF-net is to provide a comprehensible modeling language, it is possible to define execution semantics, similar to those of a normal Petri net. That is, a transition $t \in T$ is enabled in a LAWF-net under a marking (M, M_L) when: the control-flow properties for being enabled are satisfied (i.e. a token in all input places) and when all location constraints $c \in C_L^t$ are satisfied. Note that evaluating the location constraints differs for location dependent and non-location dependent transitions. For a non-location dependent transition $t \in T \backslash T_L$ the constraints are satisfied iff $\forall c \in C_L^t : [c^* = \mathit{true}]$. For a location dependent transition $t \in T_L$, the constraints are satisfied iff there exists *at least one candidate feature* f^{can} which can be bound to the transition, so that $\mathit{Type}(f^{can}) = \mathit{Type}(t) \wedge \forall c \in C_L^t : [c^*_{M_L^{can}} = \mathit{true}]$. When evaluating candidate features, however, it is important to remark that the currently bound feature to the transition at hand reflects the previous (or unset, empty) feature, whereas the evaluation of the constraints requires a location marking where the candidate feature is bound to the location dependent transition. To solve this, every candidate feature is evaluated under a candidate location marking M_L^{can} where the candidate feature is bound to the transition under evaluation. Finally, the actual execution of a location dependent transition causes a normal token movement $M_1 \xrightarrow{t} M_2$ (see before) and additionally brings the location marking in a new state $M_{L,1} \xrightarrow{t} M_{L,2}$ such that $\forall t' \in T_L : [M_{L,2}(t') = f^{selected}$ if $t' = t$ or $M_{L,1}(t')$ otherwise] with $f^{selected}$ a satisfiable feature chosen to be bound to the fired location dependent transition.

5. Executing Location-Aware Process Models

Our LAWF-net modeling extension provides a straightforward and understandable means to merge location aspects with control-flow concerns. Although we have provided execution semantics in the section above in accordance with those of a WF-net, we also define a mapping from LAWF-nets to CPN models, driven by the following reasons. First, as we will see later, mapping LAWF-nets to CPN models enables to use existing tools to drive the execution of location-aware processes. Second, as we will show below, this also allows for easier integration with existing GIS platforms. Third, by providing an approach which is fully compatible with CPN, we can apply the existing body of work concerning validation of such models, both at run-time, i.e. ensuring the process runs as designed and constraints are being enforced during execution as discussed above, but also at design-time, by performing behavioral soundness checks which investigate whether a model is sound based on a number of conditions. For instance, a model is sound if it does not allow "deadlocks" to occur: states in which it is not possible anymore to finish a running case. Finally, formulating location-aware process models in terms of CPN models also allows for more straightforward integration with other contexts, i.e. timing or social (organizational) aspects.

The following definitions provides the formalization of the mapping from LAWF-nets to CPN.

Definition 11. LAWF-net to CPN mapping. A LAWF-net $(P^L, T^L, F^L, F^L_L, FT^L_L, T^L_L, C^L_L, CF^L_L)$ is mapped to a CPN model $(P, T, A, \Sigma, C, V, N, G, E, M, I)$ as follows:

– $\Sigma = \{U, F^L_L\}$ with $U = \{unit\}$ the color set containing one control-flow oriented color and F^L_L the set of features, which also acts as a color set *(color sets)*;

– Each transition and place in the LAWF-net is also a transition in the CPN model: $\forall p \in P^L : [p \in P]$ with $C(p) = U$. $I(p) = \{unit\}$ if $\bullet\, p = \emptyset$ or $\emptyset$ otherwise and $\forall t \in T^L : [t \in T]$ *(control-flow places and transitions)*;

– For each location dependent transition in the LAWF-net, a location output place is created in the CPN model: $\forall t \in T^L_L : [p^t_L \in P]$ with $C(p^t_L) = F^L_L$ and $I(p^t_L) = \emptyset$ (the places in the bottom row in Fig. 3) *(location output places)*;

– For each feature type in the LAWF-net a location input place is created in the CPN model: $\forall ft \in FT_L^L : [p_L^{ft} \in P]$ with $C(p_L^{ft}) = F_L^L$ and $I(p_L^{ft}) = \{f \in F_L^L | \mathit{Type}(f) = ft\}$ (the places in the top row in Fig. 3) *(location input places)*;

– All control-flow of the LAWF-net is also represented in the CPN model: $\forall(x, y) \in F^L : [a_{(x,y)} \in A]$ with $N(a_{(x,y)}) = (x, y)$ and $E(a_{(x,y)}) = \mathit{unit}$ *(control-flow)*;

– A binding and overriding variable are created in the CPN model, used to bind a location to a location dependent transition and to override a previously bound transition respectively: $v_L \in V$ with $C(v_L) = F_L^L$ and $v_O \in V$ with $C(v_O) = F_L^L$ *(binding and overriding variable)*;

– For each location dependent transition, four arcs are added: one bringing the feature from the input place to the transition, one bringing the feature to the output place, a return arc returning the feature to the input place, and an override arc to remove previously bound features present in the output place: $\forall t \in T^L : [a_t^{input}, a_t^{return}, a_t^{output}, a_t^{override} \in A]$ with $N(a_t^{input}) = (p_L^{\mathit{Type}(t)}, t), N(a_t^{return}) = (t, p_L^{\mathit{Type}(t)}), N(a_t^{output}) = (t, p_L^t)$ and $N(a_t^{overrid}) = (p_L^t, t)$. $E(a_t^{input}) = E(a_t^{return}) = E(a_t^{output}) = \{v_L\}$ and $E(a_t^{override}) = \emptyset$ if $M(p_L^t) = \emptyset$ or $M(p_L^t)$ otherwise (this arc consumes the feature token from the output place if it is present) *(input, output, return and override arcs)*;

– For each location dependent transition, the guard expression in the CPN model is defined as a conjunction of all the location constraints in the LAWF-net with this transition as the output: $\forall t \in T^L : [G(t) = \bigwedge_{c \in C_L^{L^t}}(c)]$ *(guards)*;

– For location constraints involving input transitions, a variable is defined as well as two arcs to move the feature from the respective output place to the constrained transition and move it back after firing the constrained transition: $\forall t \in T^L : [\forall c \in C_L^{L^t} : [\forall (x,y) \in \{(x,y) \in CF_L^L | x \in T_L^L \wedge y = c\} : [v_L^{(x,y)} \in V, a_i^{(x,y)}, a_o^{(x,y)} \in A]]]$ with $C(v_L^{(x,y)}) = F_L^L$ and with $N(a_i^{(x,y)}) = (p_L^x, t)$ and $N(a_o^{(x,y)}) = (t, p_L^x)$. The expressions read: $E(a_i^{(x,y)}) = E(a_o^{(x,y)}) = \{v_L^{(x,y)}\}$ *(constraint input arcs).*

Fig. 3 shows the result of the conversion of a LAWF-net to a CPN model. Note that it is possible to directly model location-aware business processes as a CPN model, although the proposed LAWF-nets clearly offer a better understandable and maintainable means to do so, whilst still offering execution and validation support by means of the mapping outlined above.

6. Implementation and System Integration

The converted running example shown in Fig. 3 was implemented as a CPN model using the well-known CPN Tools program (Jensen et al., 2007). Due to some limitations of this tool, the CPN model in Fig. 3 contains some additional constructs which are not part of the formalization. First, the addition of “dummy” unlabeled transitions before some location dependent transitions. For CRT, for instance, this is necessary due to the fact that repair teams might take some time to be in the vicinity of the customer's site. However, CPN Tools only performs a check for transition enabling immediately after a marking change. The added unlabeled transitions can be used to refresh the marking (which is not actually changed in terms of token distribution) and force CPN Tools to perform the enabling check again. Second, as stated above, the execution of location dependent transitions might trigger events to be handled by an external system. The same reasoning holds for the assignment of features. Hence, we can add an arc expression to

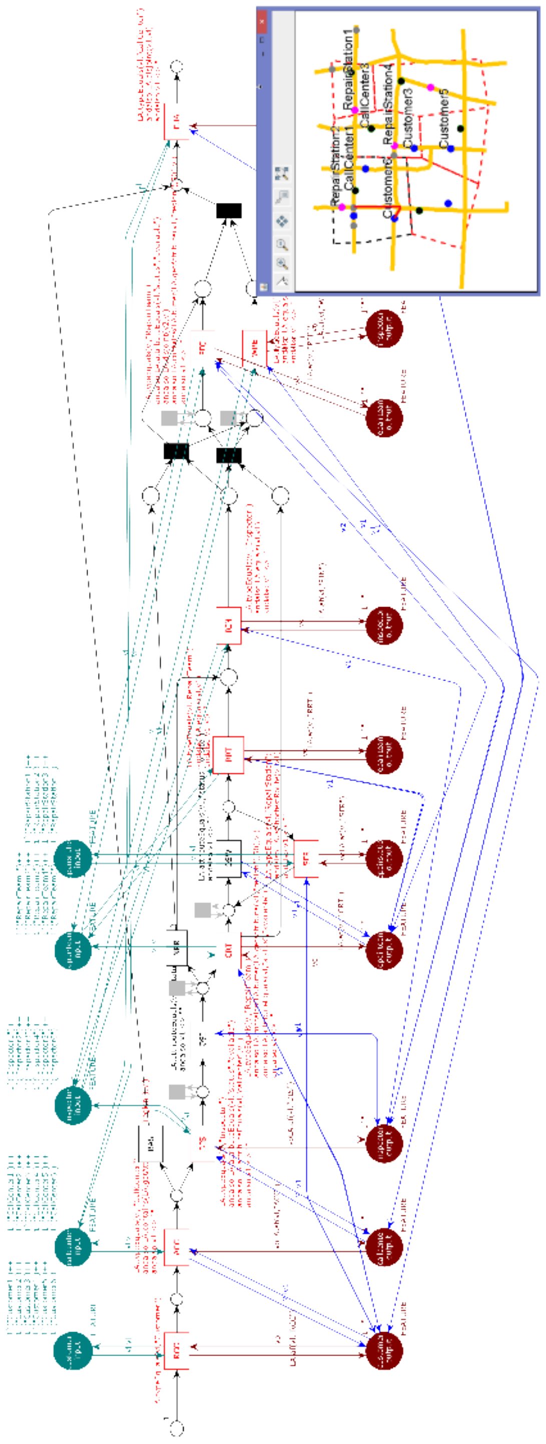

Figure 3: LAWF-net of Fig. 2 converted to a CPN model, running in tandem with a GIS system. The GIS system is able to impose geospatial constraints restricting the control-flow of the process and execution of activities can impose effects on features in the GIS system, displayed here through a map window superimposed over the model. Note that it is possible to directly model location-aware business processes as a CPN model, although the proposed LAWF-nets offer a better understandable and maintainable means to do so—as evidenced by the complexity of the CPN model above.

output arcs a_t^{out} for any $t \in T^L$ which still returns a multiset of tokens equal to v_L, but also triggers external events. This might be useful for logging purposes, or to dispatch updates to an underlying GIS system, for example to send a repair team on its way. Third, as it is not possible to formulate the expression for the $a_t^{override}$ arcs directly using CPN Tools, we instead initialize each location output place with a dummy (empty) placeholder, and add constraints to prevent this dummy feature to be used as an input for any transition using this feature as an input (we also explicitly perform a feature type check in the guard of each location dependent transition, but this is just for the sake of clarity).

Possibilities exist to extend a converted CPN model in a number of ways. If so desired, modelers might opt to use one global location input pool place instead of creating an input place per feature type (or group multiple input places). Such system would make it possible for instance to allow more than one feature type to become bound to location dependent transitions (the possible types can still be restricted by adding a constraint). Secondly, end users might opt to remove overriding $a_t^{override}$ arcs for some location dependent transition, for example to keep track of multiple bound transitions in the case of recurrent transitions. Finally, modelers might also desire for location dependent (or any) transitions to output additional features apart from the one being bound to the location dependent transition. This can also easily be achieved by adding more output places and formulating appropriate arc expressions.

Finally, the question remains how the various geospatial relationships were implemented in the CPN model. To do so, a simulation extension was developed using CPN Tools' RPC (remote procedure call) functionality. The reason this approach was chosen instead of using the built-in Standard ML language is twofold. First, both practitioners and academics are more familiar with Java (the language of the simulation extension) than Standard ML, allowing for easier understanding and extensibility. Second, this approach allows to easily integrate location-aware business process models with existing GIS systems, both for evaluating the geospatial relationships (constraining and driving the process) and to react to activities as they are being executed (within the GIS system). To illustrate this, we have created an experimental set-up using the GeoTools Java package[1], which provides support for all geospatial relationships listed before.

Fig. 3 also shows a map window which is being driven by the GIS system as it is running together with CPN Tools. The map was constructed using real-life shapefiles which were imported in the set-up. The GIS system is able to impose geospatial constraints restricting the control-flow of the process (some activities can only be started once the repair team is on-site, for instance), and execution of transitions in the process also drives changes in the GIS system, e.g. sending out a request to a repair team causes this repair team to head to the customer's location using the shortest route available (as is shown in the figure). This illustrates the feasibility and validity of our proposed methodology[2].

7. Conclusions

For the most part, the modeling and execution of business process models has so far been confined to a rather limiting environment, focusing mainly on control-flow aspects only, without taking rich contextual aspects into

[1]See: `http://geotools.org`

[2]Source code of the developed implementation together with additional documentation regarding the architecture can be found at: `http://processmining.be/locationaware`.

account. In this paper, we have focused our attention towards making the modeling and execution of business processes location-aware, focusing on the particular context of geospatial information. A Petri net modeling extension was proposed which incorporates location aspects and ways to constrain the execution of activities based on location constraints. This approach was formalized by means of a formalized mapping to coloured Petri nets and implemented in combination with an experimental GIS setup to illustrate the feasibility of coupling business process support systems with geographic information systems.

We believe our contribution to be a valuable step towards incorporating location aspects in business processes, with application areas in logistics, transportation and many others. Indeed, the ability to make processes flexible and adaptive in terms of their ability to react to road, traffic or weather conditions is put forward as a promising area of study. Regarding future work, we plan to set up a number of case studies and to further enable the application of our approach in real-life contexts. To do so, we aim to develop a visual modeling tool which allows for the design and execution of LAWF-nets, whilst providing information regarding potential design problems (by performing the mapping to CPN models in the background). Additionally, the tool should allow to supply domain-specific knowledge (definition of features, maps, etc.) in a user-friendly manner. Next, concerning modularity, it is worthwhile to investigate how the integration with GIS systems other than the one applied in this work can be facilitated. Finally, we also plan to expand on our methodology, both by investigating more location-based patterns that play a role in business process environments (the focus here was mainly on geospatial constraints) and how these aspects can be combined with other contextual aspects other than geographic information as well.

Acknowledgments

This work is supported by the National Key Technology R&D Program, China (grant 2012BAH01F02), by the KU Leuven research council (grant OT/10/010) and the Flemish Research Council (Odysseus grant B.0915.09).

References

Ali, S., Torabi, T., Ali, H., 2006. Location aware business process deployment. In: Computational Science and Its Applications - ICCSA 2006, Int'l Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part IV. Vol. 3983 of Lecture Notes in Computer Science. Springer, pp. 217–225.

Alonso, G., Hagen, C., 1997. Geo-opera: Workflow concepts for spatial processes. In: Advances in Spatial Databases. Vol. 1262 of LNCS. pp. 238–258.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S., Jun 2004. Kepler: an extensible system for design and execution of scientific workflows. In: Scientific and Statistical Database Management, 2004. Proceedings. 16th Int'l Conference on. pp. 423–424.

Amadio, G., Cannafoglia, C., Corongiu, M., Desideri, M., Rossi, M., 2004. Intesagis: the basis for a national spatial data infrastructure. In: proceedings of the 10th EC-GI&GIS Workshop, Warsaw, Poland.

Aoumeur, N., Fiadeiro, J. L., Oliveira, C., 2004. Towards an architectural approach to location-aware business processes. In: 13th IEEE Int'l Workshops on Enabling Technologies (WETICE 2004), Infrastructure for Collaborative Enterprises, 14-16 June 2004, Modena, Italy. IEEE Computer Society, pp. 147–152.

Brodeur, J., Bédard, Y., Proulx, M.-J., 2000. Modelling geospatial application databases using uml-based repositories aligned with international standards in geomatics. In: Proceedings of the 8th ACM Int'l Symposium on Advances in Geographic Information Systems. GIS '00. pp. 39–46.

Chakraborty, D., Lei, H., 2004. Pervasive enablement of business processes. In: Proceedings of the 2nd IEEE Int'l Conference on Pervasive Computing and Communications (PerCom 2004), 14-17 March 2004, Orlando, FL, USA. IEEE Computer Society, pp. 87–100.

Clementini, E., Felice, P. D., van Oosterom, P., 1993. A small set of formal topological relationships suitable for end-user interaction. In: SSD. Vol. 692 of Lecture Notes in Computer Science. Springer, pp. 277–295.

Clementini, E., Sharma, J., Egenhofer, M. J., 1994. Modelling topological spatial relations: Strategies for query processing. Computers & Graphics 18 (6), 815–822.

Decker, M., Oct 2009. Modelling location-aware access control constraints for mobile workflows with uml activity diagrams. In: UBICOMM '09. pp. 263–268.

Decker, M., Che, H., Oberweis, A., Sturzel, P., Vogel, M., Jun 2010. Modeling mobile workflows with bpmn. In: Mobile Business and 2010 9th Global Mobility Roundtable (ICMB-GMR), 2010 9th Int'l Conference on. pp. 272–279.

Egenhofer, M. J., Franzosa, R. D., 1991. Point-set topological spatial relations. International Journal of Geographical Information System 5 (2), 161–174.

Egenhofer, M. J., Herring, J., 1990. A mathematical framework for the definition of topological relationships. In: 4th Int'l Symposium on Spatial Data Handling. Zurich, Switzerland, pp. 803–813.

Georgoulias, K., Papakostas, N., Chryssolouris, G., Stanev, S., Krappe, H., Ovtcharova, J., 2009. Evaluation of flexibility for the effective change management of manufacturing organizations. Robot. Comp.-Integr. Manuf. 25 (6), 888–893.

Günther, O., 1997. Environmental information systems. SIGMOD Rec. 26 (1), 3–4.

Jensen, K., 1987. Coloured petri nets. Springer.

Jensen, K., Kristensen, L. M., Wells, L., 2007. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer 9 (3-4), 213–254.

Jäger, E., Altintas, I., Zhang, J., Ludäscher, B., Pennington, D., Michener, W., 2005. A scientific workflow approach to distributed geospatial data processing using web services. In: SSDBM05. SSDBM, pp. 87–90.

Jing, J., Huff, K. E., Hurwitz, B., Sinha, H., Robinson, B., Feblowitz, M., 2000. WHAM: supporting mobile workforce and applications in workflow environments. In: Proceedings of the 10th Int'l Workshop on Research Issues on Data Engineering: Middleware for Mobile Business Applications and E-Commerce, San Diego, California, USA, February 27-28, 2000. IEEE Computer Society, pp. 31–38.

Lake, R., Burggraf, D. S., Trninic, M., Rae, L., 2004. Geography mark-up language (GML). Foundation for the GeoWeb. Wiley, Chichester.

Leoni, M., Aalst, W., Hofstede, A., 2008. Visual support for work assignment in process-aware information systems. In: Business Process Management. Vol. 5240 of LNCS. pp. 67–83.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., Zhao, Y., 2006. Scientific workflow management and the kepler system. Concurrency and Computation: Practice and Experience 18 (10), 1039–1065.

Medeiros, C., Vossen, G., Weske, M., Jun 1996. Geo-wasa-combining gis technology with workflow management. In: Computer Systems and Software Engineering, 1996., Proceedings of the 7th Israeli Conference on. pp. 129–139.

Meeks, W. L., Dasgupta, S., 2004. Geospatial information utility: an estimation of the relevance of geospatial information to users. Decision Support Systems 38 (1), 47–63.

Murata, T., 1989. Petri nets: Properties, analysis and applications. In: Proceedings of the IEEE. Vol. 77. pp. 541–580.

OGC, 1999. The opengis abstract specification - topic 8: Relationships between features.

Peterson, J. L., 1981. Petri Net Theory and the Modeling of Systems. New Jersey: Prentice-Hall, Inc.

Recker, J. C., May 2012. "modeling with tools is easier, believe me" : the effects of tool functionality on modeling grammar usage beliefs. Information Systems 37 (3), 213–226.

Recker, J. C., Rosemann, M., Indulska, M., Green, P., Apr 2009. Business process modeling : a comparative analysis. Journal of the Association for Information Systems 10 (4), 333–363.

Rosemann, M., Recker, J. C., Flender, C., 2008. Contextualisation of business processes. International Journal of Business Process Integration and Management 3 (1), 47–60.

Seffino, L. A., Medeiros, C. B., Rocha, J. V., Yi, B., 1999. Woodss—a spatial decision support system based on workflows. Decision Support Systems 27 (1), 105–123.

van der Aalst, W. M., 1998. The application of petri nets to workflow management. Journal of circuits, systems, and computers 8 (01), 21–66.

van der Aalst, W. M. P., Hofstede, A. H. M. t., Weske, M., 2003. Business process management: A survey. In: Proceedings of the 2003 Int'l Conference on Business Process Management. BPM'03. Springer-Verlag, Berlin, Heidelberg, pp. 1–12.

Vom Brocke, J., Rosemann, M., 2010. Handbook on Business Process Management: Strategic Alignment, Governance, People and Culture. Springer.

Zhu, X., Recker, J., Zhu, G., Santoro, F., 2014. Exploring location-dependency in process modeling. Business Process Management Journal 20 (6).

Zhu, X., Zhu, G., Guan, P., 2013. Exploring location-aware process management. In: Geo-Informatics in Resource Management and Sustainable Ecosystem. Vol. 399 of Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 249–256.