Towards Location-Aware Process Modeling and Execution

Xinwei Zhu[1], Guobin Zhu[1], Seppe K.L.M. vanden Broucke[2], Jan Vanthienen[2] and Bart Baesens[2]

[1] International School of Software, Wuhan University
Luoyu Road 37, Hongshan, Wuhan, Hubei, China
[2] Department of Decision Sciences and Information Management, KU Leuven
Naamsestraat 69, B-3000 Leuven, Belgium
xinwei.zhu@whu.edu.cn

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

Key words: business process management, geographical information systems, location-aware 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 in order to align organizational business processes and workflows to the needs of clients [1]. 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 view only are unable to adequately capture human behavior due to

lack of descriptions of possible constraints against activity modeling. 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.).

This paper contributes to the existing body of work by focusing on the particular *context of geospatial information*. We 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 were a need arises to apply both process-aware and Geographical Information Systems (GIS), it makes sense to combine and integrate these two perspectives instead of keeping them isolated. Such an approach would help to increase understandability and objectiveness of designed process models, govern and constrain control-flow and process behavior based on location driven constraints, and allow for better location based monitoring and feedback support during execution of such processes. The goal of this paper is thus to make the modeling and execution aspects of BPM "location-aware". We do so by proposing a Petri net modeling extension which incorporates location aspects and ways to constrain the execution of activities based on location-based constraints. Next, we formalize the execution semantics of our extension by describing a mapping to coloured Petri nets. Finally, we have implemented our approach using CPN Tools [2]; 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 geographical information systems.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work and preliminaries. Section 3 outlines a running example which will be used to illustrate the developed artifacts. Next, Section 4 introduces the Petri net modeling extension to model 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.

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. [3], 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) [4], increase the adaptability and flexibility of running processes

(by reconfiguring and modifying models and tasks based on location aspects) [5], and improve the efficiency (performance and cost-effectiveness) of organizational processes [6].

However, works around the connection of location services with principles of business process management are scarce in the literature. Many researchers focus on on connecting spatial-based information with scientific workflows [7, 8, 9, 10, 11] (i.e. describing a series of scientific-oriented computational or data manipulation steps), but not with business workflows. As a notable exception, [12] discusses 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 geographical aspects can influence the behavioral aspects of the process. Decker et al. [4, 13] 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.

Some existing BPM tool suites allow for the definition and capture of additional variables in the modeling of business processes [14, 15]. Such attribute fields could be used to capture location-based information, but only in the form of secondary constructs or text-based annotations for readers to understand the graphical diagram, which 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.

2.2 Definitions and Notations

This section outlines some preliminary concepts and definitions which will be utilized in the remainder of the paper. We assume readers to be familiar with the concept of Petri nets [16, 17] and their execution semantics. A particular subclass of Petri nets which we utilize hereafter are called WorkFlow nets (or WF-nets) **[18]**. A WF-net specifies the behavior of a single process instance in isolation and is defined as follows.

Definition 1. ***WF-net [18].*** *A Petri net (P, T, F) is a WF-net iff:*

- *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) [2]. CPNs are an extension of Petri net which allow for tokens of multiple types ("colors"), add guard transitions to constrain the execution of transitions and add arc expression to govern the input and output flow of tokens.

Definition 2. ***Coloured Petri net (see [2]).*** *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, $T = \{t_1, t_2, ..., t_{|T|}\}$ *with* $P \cap T = \emptyset$*; A the set of arcs,* $A = \{a_1, a_2, ..., a_{|A|}\}$*;* Σ *the set of color sets defined within the model; V the set of variables used in the model,* $V = \{v_1, v_2, \ldots, v_{|V|}\}$*;* $C : P \cup V \rightarrow \Sigma$ *returning the color set associated to a place or a variable;* $N : A \rightarrow P \times T \cup T \times P$ *mapping arcs to a place-transition or transition-place flow.* $G : T \rightarrow \{GExpr\}$ *is the guard expression function mapping a transition* $t \in T$ *to a boolean expression GExpr, denoting whether the transition is permitted to fire. Evaluating this expression yields a boolean result* $GExpr^* \in \{true, false\}$*.* $E : A \rightarrow \{AExpr\}$ *is the arc expression function mapping an arc* $a \in A$ *to an expression AExpr. Evaluating an arc expression yields a multiset of tokens,* $AExpr^*_{MS}$ *to be produced (transition to place arcs) or consumed (place to transition arcs).* $M : p \in P \mapsto C(p)_{MS}$ *is the marking function, returning the multi set of tokens contained in a place with* $\forall p \in P : [\forall \sigma \in M(p) : [\sigma \in C(p)]]$ *and* $I : P \rightarrow \{IExpr\}$ *is the initialization function, initializing places in the model with a state, expressed as colored tokens. The evaluation of an IExpr yields a token multi set:* $IExpr^*_{MS}$ *with* $\forall p \in P : [\forall \sigma \in IExpr^*_{MS} : [\sigma \in C(p)]]$*.*

The execution semantics of a CPN differ from those of a regular Petri net. For a transition $t \in T$ to be enabled, all expressions of the incoming arcs should be satisfied and the guard condition of the transition must evaluate to true, $G(t)^* = true$. When firing an enabled transition, output and input places are updated accordingly given the input and output arc expressions.

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 geographical information systems [19]. A feature is describes something that can be drawn on a map, i.e. something in the real world—a monument, a landmark or even moving objects such as cars or trains. 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 location in terms of points, lines, paths (multiple line segments) or polygons, associated with a well-defined coordinate reference system.

Definition 3. ***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). 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. We will also denote this using the shorthand* $f : ft$ *with* $f \in F_L$ *and* $ft \in FT_L$*. Let* $Geometry : f \in F_L \mapsto g$ *be a function which returns the geometry (g) for a given location with g a point, line, path or polygon and let* f^a *indicate some attribute a associated with a feature* $f \in F_L$*. This can be data of any type, e.g. a name or even other features.*

In our definition, a feature is only represented by one geometry type only, although this can be easily extended (and is already supported in an indirect

manner by construction of additional features which serve as attributes for the feature at hand).

Finally, we introduce the concept of a "geospatial relationship". By defining the concept of a geometry and establishing relationships over them, we are able to answer queries such as "Is one feature contained in another?" Many definitions for such relationships exist, but many of the most widely used GIS toolkits, e.g. GeoTools, PostGIS, ArcGIS, SQL Server etc., define geospatial relationships based on the "Topic 8" standard proposed by the OpenGIS Consortium (OGC) [19]. We will also utilize these relationships towards the construction of location constraints, as will be illustrated later. Note that, in some cases, geospatial relationships are categorized in separate sets, such as topological, measurement, sequential or complex relationships [6, 20], but for the sake of simplicity, we use one global moniker (a "geospatial relationship") in this work.

3 Running Example

To illustrate our location-aware Petri net modeling extension and its execution semantics, we will utilize a running example throughout this paper, inspired on the case examples provided in [13]. The basic WF-net is depicted 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 in a particular call center (Receive Customer Call, RCC and Accept Customer Call, ACC). The call center evaluates the complaint, based on which the user is remotely assisted (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, 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, 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 can occur multiple times). After the repair is finished, the repair team is called back (Release Repair Team, RTT). 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.

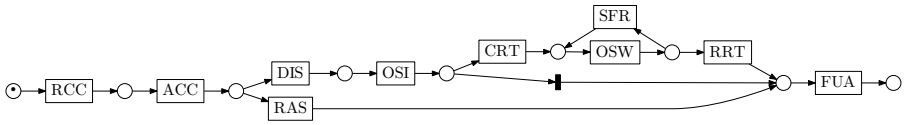

Fig. 1. WF-net model of "repair" process used as a running example throughout the paper.

The description of this process highlights some locational aspects which can not be captured by control-flow alone. In particular, we list the following locational concerns which need to be adhered to:

- Call centers may only handle customer calls when the customer is located within the region a call center is responsible for;
- The call center performing the follow-up work should be situated in a different region then call center handling the customer call;
- 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;
- Shop-floor repairs should be made in the repair station closest to the customer's location; this should be based on navigational routing information, not on beeline distance.

4 Location-Aware Workflow Modeling

This section discusses our proposed Petri net modeling extension to model location-aware processes. Our methodology includes two main extensions, namely *location dependent transitions* and *location constraints*.

Fig. 2 shows the running example modeled using our location-aware extension. Location dependent transitions are indicated with a flag (⚑), together with the feature name and type for the location which will be bound to the transition after executing. The shaded boxes represent location-aware constraints, used to constrain the locations which can be bound to a location dependent transition. Visually, the constraints are connected with all the transitions which bound location will be used as an input in the constraint, and with one transition which is bounded by the constraint (using dashed arcs).

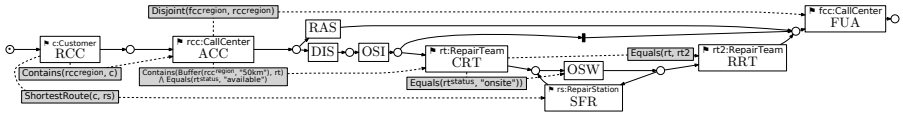

Fig. 2. The running example of Fig. 1 modeled using our location-aware extension.

Definition 4. ***Location-aware WF-net (LAWF-net).*** *Formally, 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 before);* $T_L \subseteq T$ *the set of location dependent transitions* C_L *the set of location constraints (a set of expressions);* $CF_L \subseteq (T_L \times C_L) \cup (C_L \times T)$ *a finite set of directed arcs linking location dependent transitions to a constraint.*

We also define the function $LM : T_L \rightarrow F$ to get the feature bound to a particular location dependent transition (the location marking). Initially, i.e. before execution of a transition, $\forall t \in T_L : [LM(t) = \emptyset]$. Most location constraints are formulated directly in the form of a geospatial relationship (when the relationship returns a boolean result), but generally there are no strict criteria for the definition of a location constraint $c \in C_L$ except for the following.

Definition 5. ***Location constraint.*** *A location constraint is an expression $c \in C_L$ which evaluates to a boolean result $c^* \in \{true, false\}$. The expression involves exactly one output transition, i.e. $\exists!(x, y) \in CF_L : [x = c \wedge y \in T]$. $C_L^t = \{c \in C_L | \exists!(x, y) \in CF_L : [x = c \wedge y = t]\}$ is used as a shorthand to return all constraints defined on $t \in T$. The expression can involve zero or more location dependent input transitions. The feature bound to such input transitions, given by LM will be used as an input for the expression at the time of evaluation.*

As stated in the preliminaries section, we mainly apply the geospatial relationships as defined by the "Topic 8" standard proposed by the OpenGIS Consortium (OGC) [19] towards formulating location constraints. As an example, in Fig. 2, the constraint "$Contains(rcc^{region}, c)$" contains one output transition (ACC) and one input transition (RCC)". Instead of using the transition labels in the expression, we use a short name ("rcc" or "c") as a way to indicate bound features for location dependent transitions directly. "rcc^{region}" should thus be read as "the region attribute (another feature) of rcc (the feature bound to the ACC transition). We also define a feature type for each location dependent transition indicating the type of the features which can be bound to the transition. This means that, even when no constraints are modeled, an intrinsic "$Type(LM(t)) = x$" constraint is present for any $t \in T_L$ with $x \in FT_L$ the defined feature type. We can thus also define $Type : T_L \rightarrow FT_L$ as a shorthand function returning the feature type for a location dependent transition. Note also that constraints can also be defined over a non-location dependent transition, as is the case with "$Equals(rt^{status}, "onsite")$". Such constraints govern the execution of non-location dependent transitions without binding a location to them.

The execution semantics of LAWF-nets are similar to those of a normal Petri net.

Definition 6. ***LAWF-net execution semantics.*** *A transition $t \in T$ is enabled in a LAWF-net iff: the control-flow properties for being enabled are satisfied (i.e. a token in all input places) and all constraints $c \in C_L^t$ are satisfied, i.e. $\forall c \in C_L^t : [c^* = true]$ if $t \in T \backslash T_L$. If $t \in T_L$ (i.e. t is a location-dependent transition), evaluating the satisfyability of the constraints involves checking whether there exists a feature which can be bound to the transition, i.e. $\exists f \in F_L : [\forall c \in C_L^t : [c^* = true]]$*[1].

[1] Naturally, when evaluating the conditions for a transition to be enabled, the currently bound feature to that transition reflects the previous (or unset) feature, whereas the evaluation of the constraint satisfyability requires a location marking

Firing a location dependent transition causes a normal token movement and additionally brings the location marking in a new state $LM_1 \xrightarrow{t} LM_2$ such that $LM_2(t) = f$ and $LM_2(x) = LM_1(x)$ for all $x \in T_L \wedge x \neq t$. That is, a satisfiable feature is bound to the location dependent transition.

5 Execution of 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, we also provide a mapping from LAWF-nets to CPN models, due to 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, it also allows for easier integration with existing GIS platforms. Third, by providing an approach which is fully compatible with CPN, we can build open a large existing body of work concerning validation of such models (i.e. ensuring the correctness of the designed model). Finally, formulating location-aware process models in terms of CPN models also allows for integration with other contexts, i.e. timing or social (organizational) aspects.

Fig. 3 shows the result of the conversion of a LAWF-net to a CPN model. The following definitions provides the formalization of this mapping.

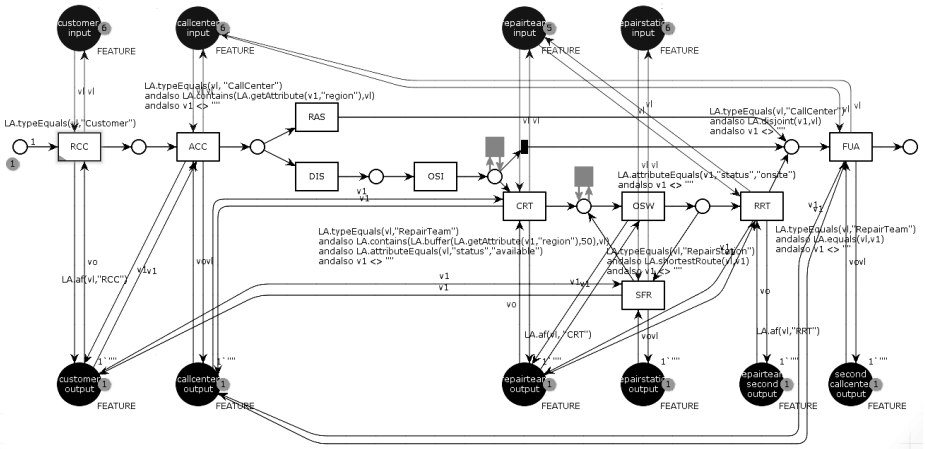

Fig. 3. LAWF-net of Fig. 2 converted to a CPN model.

where the feature under consideration is bound to the location dependent transition. To resolve this, every $f \in F_L$ is evaluated under a location marking $LM_{t,f}$ such that $LM_{t,f}(t) = f$ and $LM_2(x) = LM_1(x)$ for all $x \in T_L \wedge x \neq t$. In case that all constraints hold under this candidate location marking and this feature is chosen to be bound, $LM_{t,f}$ will be finalized as the new marking after firing the transition.

Definition 7. ***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* $U = \{unit\}$ *(color sets);*

– $\forall p \in P^L : [p \in P]$ *with* $C(p) = U$. $I(p) = \{unit\}$ *iff* $\bullet p = \emptyset$, $I(p) = \emptyset$ *otherwise and* $\forall t \in T^L : [t \in T]$ *(control-flow places and transitions);*

– $\forall t \in T_L^L : [p_L^t \in P]$ *with* $C(p_L^t) = F_L^L$ *and* $I(p_L^t) = \emptyset$ *(the places in the bottom row labeled "output" in Fig. 3) (location output places);*

– $\forall ft \in FT_L^L : [p_L^{ft} \in P]$ *with* $C(p_L^{ft}) = F_L^L$ *and* $I(p_L^{ft}) = \{f \in F_L^L | Type(f) = ft\}$ *(the places in the bottom row labeled "input" in Fig. 3) (location input places);*

– $\forall (x, y) \in F^L : [a_{(x,y)} \in A]$ *with* $N(a_{(x,y)}) = (x, y)$ *and* $E(a_{(x,y)}) = unit$ *(control-flow);*

– $v_L \in V$ *with* $C(v_L) = F_L^L$ *and* $v_O \in V$ *with* $C(v_O) = F_L^L$ *(binding and overriding variable);*

– $\forall t \in T^L : [a_t^{input}, a_t^{return}, a_t^{output}, a_t^{override} \in A]$ *with* $N(a_t^{input}) = (p_L^{Type(t)}, t)$, $N(a_t^{return}) = (t, p_L^{Type(t)})$, $N(a_t^{output}) = (t, p_L^t)$ *and* $N(a_t^{overrid}) = (p_L^t, t)$. $E(a_t^{input}) = E(a_t^{return}) = E(a_t^{output}) = v_L$ *and* $E(a_t^{override}) = if\ |M(p_L^t)| = 0$ *then empty else* $M(p_L^t)$ *(this arc consumes the feature token from the output place if it is present) (input, output, return and override arcs);*

– $\forall t \in T^L : G(t) = \bigwedge_{c \in C_L^{L^t}}(c)$ *(a conjunction of all the constraints with this transition as the output) (guards);*

– $\forall t \in T^L : [\forall c \in C_L^{L^t} : [\forall (x, y) \in \{(x, y) \in CF_L^L | x \in T_L^L \wedge y = c\} : [v_L^{(x,y)} \in V, a_i^{(x,y)}, a_o^{(x,y)} \in A]]]$ *with* $C(v_L^{(x,y)}) = F_L^L$ *and with* $N(a_i^{(x,y)}) = (p_L^x, t)$ *and* $N(a_o^{(x,y)}) = (t, p_L^x)$. $E(a_i^{(x,y)}) = E(a_o^{(x,y)}) = v_L^{(x,y)}$ *(constraint input arcs).*

6 Implementation and System Integration

The converted running example shown in Fig. 3 was implemented as a CPN model using CPN Tools [2]. Note that, due to the limitations of this tool, the CPN model in contains some additional constructs which are not part of the formalization. First, the addition of "dummy" invisible transitions before CRT and OSW. This is due to the fact that CPN Tools only performs a check for enabling tasks immediately after a marking change, and does not repeat this check when the user requests to execute a particular transition. Therefore, the invisible transitions are used as a workaround to force CPN Tools to perform the check again. Second, we can modify the arc expression of 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—as we do here—fire an event to an underlying GIS system which sends a repair team on their way. Third, as it is impossible to formulate the expression for the $a_t^{override}$ arcs 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 (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).

Modelers are free to extend a converted CPN model. If so desired, for instance, modelers might opt to use one global location input pool place instead of creating an input place per feature type. Such system would make it possible for instance to allow more than one feature type to become bound to location dependent transitions. 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 other 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.

The question remains how the various location constraints 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 CPN Tools' built-in SML language is twofold. First, both practitioners and academics are more familiar with Java (the language of the simulation extension) than SML, allowing for easier understanding and extension. Second, this approach allows to easily integrate location-aware business process models with existing GIS systems, both for evaluating the geospation 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[2]. Fig. 4 shows the running CPN model running with a GIS system. A map is shown based on real-life shapefiles which were imported in the set-up. As illustrated, 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). Vice versa, 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 Fig. 4). This illustrates the feasibility and validity of our proposed methodology[3].

7 Conclusion

For the most part, the modeling and execution of business process models has been studied and performed in practice in a rather limiting environment, dealing mainly with control-flow aspects only, without taking other contextual aspects into account. In this paper, we have focused our attention towards making the modeling and execution of business processes location-aware, i.e. on the particular context of geospatial information. A Petri net modeling extension was

[2] See: `http://geotools.org`. This toolkit offers support for all geospatial relationships defined by the "Topic 8" standard proposed by the OpenGIS Consortium (OGC) [19].

[3] Full source code of the developed implementation can be found at: `http://processmining.be/locationaware`.

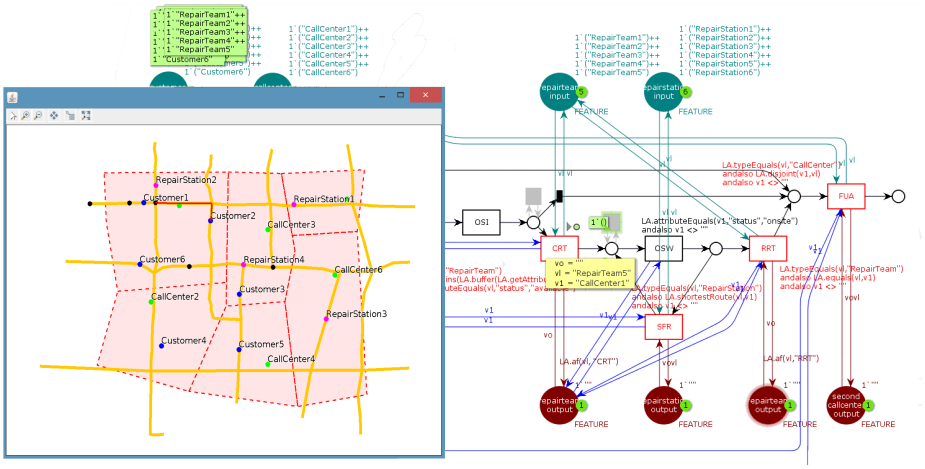

Fig. 4. CPN model running in tandem with a GIS system. The GIS system is able to impose geospatial constraints restricting the control-flow of the process. Vice versa, execution of activities can impose effects on features in the GIS system.

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 geographical information systems.

We believe our contribution to be a first valuable step towards incorporating location aspects in business processes, hence allowing stake holders to execute such processes in a more effective and efficient manner, with application areas in logistics, transportation and 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. Therefore, in future work, we plan to set up a number of case studies to elaborate on the feasibility and robustness of our approach. In addition, we plan to expand on our methodology, both by investigating more location-based patterns which 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 geographical information as well.

Acknowledgment

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

1. Vom Brocke, J., Rosemann, M.: Handbook on Business Process Management: Strategic Alignment, Governance, People and Culture. Springer (2010)
2. Jensen, K., Kristensen, L.M., Wells, L.: Coloured petri nets and cpn tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer **9**(3-4) (2007) 213–254
3. Rosemann, M., Recker, J.C., Flender, C.: Contextualisation of business processes. Int. Journal of Business Process Integration and Management **3**(1) (2008) 47–60
4. Decker, M.: Modelling location-aware access control constraints for mobile workflows with uml activity diagrams. In: UBICOMM '09. (Oct 2009) 263–268
5. Georgoulias, K., Papakostas, N., Chryssolouris, G., Stanev, S., Krappe, H., Ovtcharova, J.: Evaluation of flexibility for the effective change management of manufacturing organizations. Robot. Comp.-Integr. Manuf. **25**(6) (2009) 888–893
6. Zhu, X., Recker, J., Zhu, G., Santoro, F.: Exploring location-dependency in process modeling. Business Process Management Journal **20**(6) (2014)
7. Medeiros, C., Vossen, G., Weske, M.: Geo-wasa-combining gis technology with workflow management. In: Computer Systems and Software Engineering, 1996., Proceedings of the Seventh Israeli Conference on. (Jun 1996) 129–139
8. Alonso, G., Hagen, C.: Geo-opera: Workflow concepts for spatial processes. In: Advances in Spatial Databases. Volume 1262 of LNCS. (1997) 238–258
9. Seffino, L.A., Medeiros, C.B., Rocha, J.V., Yi, B.: Woodss—a spatial decision support system based on workflows. DSS **27**(1) (1999) 105–123
10. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the kepler system. Concurrency and Computation: Practice and Experience **18**(10) (2006) 1039–1065
11. Jäger, E., Altintas, I., Zhang, J., Ludäscher, B., Pennington, D., Michener, W.: A scientific workflow approach to distributed geospatial data processing using web services. In: SSDBM05, SSDBM (2005) 87–90
12. Leoni, M., Aalst, W., Hofstede, A.: Visual support for work assignment in process-aware information systems. In: BPM. Volume 5240 of LNCS. (2008) 67–83
13. Decker, M., Che, H., Oberweis, A., Sturzel, P., Vogel, M.: Modeling mobile workflows with bpmn. In: Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on. (Jun 2010) 272–279
14. Recker, J.C.: "modeling with tools is easier, believe me" : the effects of tool functionality on modeling grammar usage beliefs. Information Systems **37**(3) (May 2012) 213–226
15. Recker, J.C., Rosemann, M., Indulska, M., Green, P.: Business process modeling : a comparative analysis. Journal of the Association for Information Systems **10**(4) (Apr 2009) 333–363
16. Murata, T.: Petri nets: Properties, analysis and applications. In: Proceedings of the IEEE. Volume 77. (1989) 541–580
17. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. New Jersey: Prentice-Hall, Inc. (1981)
18. van der Aalst, W.M.: The application of petri nets to workflow management. Journal of circuits, systems, and computers **8**(01) (1998) 21–66
19. OGC: The opengis abstract specification–topic 8: Relationships between features (1999)
20. Zhu, X., Zhu, G., Guan, P.: Exploring location-aware process management. In: Geo-Informatics in Resource Management and Sustainable Ecosystem. Volume 399 of Communications in Comp. and IS. Springer Berlin Heidelberg (2013) 249–256