Towards an Architectural Approach to Location-aware Business Processes

Nasreddine Aoumeur∗
José Fiadeiro
Cristóvão Oliveira†

Department of Computer Science
University of Leicester
LE1 7RH, UK
na80@le.ac.uk, jose@fiadeiro.org, co@di.fct.unl.pt

Abstract

We present first steps towards modelling location-aware business processes based on architectural primitives that support the separation of three important concerns: computation, coordination, and distribution. We focus in particular on the primitives that bring the expressive power of architectural connectors to the modelling of activity flows according to given business rules. Coordination primitives externalise, from business rules, the way business entities interact within given business activities. Location primitives externalise the dependencies that such rules put on the way activities are distributed and performed across communication networks.

1 Introduction

In recent work on architectural techniques for software-intensive systems, the main focus has significantly shifted from just supporting the construction of architectures in terms of components and connectors, towards concerns like evolution and context-awareness. That is, the current emphasis is shifting towards reconﬁgurable and pervasive (mobile/ubiquitous)-driven architectures [6]. This shift is being dictated by the need for companies and organisations to keep pace with the huge advances that are being made in Information and Communication technologies.

Business processes are among the domains that have extensively beneﬁted from and dramatically inﬂuenced these advances. Unanticipated and frequent changes, as well as context-awareness, are becoming the norm in business today. For instance, to remain competitive, banking systems are continuously creating/updating new services depending on the proﬁles of their customers, while at the same time supporting more and more advanced ‘devices/channels’ for banking such as ATM, Internet, PDA, Pay-TV, inter alia [9].

At the software architectural level, the operational view of business processes is still dominating (e.g. workflow techniques [1]). However, recent investigations are shifting the emphasis towards more abstraction through business rule-driven architectures [10]. These are considered to provide a more adequate level for coping with business volatility and, hence, dynamic-evolution at the business domain level.

In this paper, we argue and demonstrate that location-awareness, which includes related communication/mobility aspects, should be perceived at a higher architectural level, more precisely at the same level as for interaction and coordination aspects as typically captured by architectural connectors. More precisely, as first steps towards a location-aware software architecture for specifying and dynamically evolving activity flows, we propose separate conceptual primitives for
modelling coordination and location at the same level of abstraction. In particular, we show how both the coordination and location dimensions are required, and how they can coexist in harmony, complementing each other in the modelling of business activities.

Coordination primitives, as we have been promoting in recent work [5], provide a clean separation between computations, as performed by stable core business entities to ensure the functionality of basic business services, and the mechanisms that reflect how the interactions between these business entities should be coordinated according to given business rules. Such interactions are captured using the concepts of coordination laws and interfaces. In terms of Architecture Description Languages (ADL), coordination laws and interfaces correspond to connector types and roles. In terms of business modelling, they capture business rules that regulate and compose services provided by the core entities that instantiate the roles.

The location primitives that we are putting forward for the first time in this paper capitalise on our recent work on extending CommUnity, a formal approach that we have developing for architectural description, with primitives that capture distribution and mobility aspects [8]. More precisely, we borrow the definition of spaces of mobility (i.e. location structures) and corresponding contexts with the "be-in-touch" relationship as a preconditions for communicating entities.

The rest of this paper is structured as follows. In the second section, we recall some notions about business processes with specific emphasis on their activity flows, and we present our running example dealing with accounts in banking systems. In the third section, coordination concepts are illustrated using the example. In the fourth section, location concepts are presented and illustrated using the same example. In the last main section, we present the coordination-location based architecture for modelling and evolving location-aware business activities. We conclude with some remarks and an outline for future work.

2 Business processes: definition and running example

As already mentioned, we perceive business processes from a communication, software-driven activity perspective. That is, on the basis of the definitions forwarded in [7] and [2], we regard business processes as sets of activities constrained, among others, by causal or temporal relationships as well as specific business rules. The activities performed within a business process involve roles or actors related to a given virtual or real inter-organisational enterprise.

The running example we are adopting concerns the withdrawal of money in a banking system. This simple business process consists of several activities, including specifically:

- The customer identification and authentication;
- The customer request for a specific amount;
- The withdrawal/denial of the requested amount from the corresponding account

These activities must be performed according to given business rules. For instance, a withdrawal request can only be issued after the customer has been properly identified.

This high-level description of the process omits certain aspects of the business rules that apply and determine how each of the activities is actually performed. For instance, it is clear that the withdrawal activity at the core of the process is subject to the business rules that apply to the specific customer and account involved. Depending on the nature of the account and of the customer, certain constraints may apply that determine if, for a given amount, the withdrawal is authorised and, if so, what operations it involves on the bank account itself. The aspects that relate to the way business rules determine how the business entities involved in a business activity interact fall under what we call "coordination".

On the other hand, depending on the location of the customer, the business activities may be subject to different rules. For instance, the identification-authentication activities can be: (1) a simple "hello" when the customer is at the desk of the branch where the account has been held for 20 years; (2) the presentation of a personal identity document if the clerk has only recently joined that branch or if the customer is at a different branch; (3) a complex transaction involving debit cards and codes if the customer is at an ATM; (4) a collection of passwords, security codes and predetermined personal questions if the customer is using the internet; (5) etc.

To the best of our knowledge there is no conceptual modelling approach that addresses location-awareness
in business processes in the sense that we have illustrated, except for the work in [3], one of our main sources of inspiration. This work invokes the notion of 'channel' for addressing location-awareness. It is, altogether, rather "operational", not as declarative as we wish to be, because it uses state machines as a modelling tool. It does not cope with the evolutionary side either, and it has not been integrated within an architectural approach that provides explicit connectors that can handle location-dependency aspects.

Our goal in this paper is to show how we can capture the coordination and location concerns involved in business rules through modelling primitives that can be separately and dynamically superposed over business activities such as those above. In terms of the underlying business process, these primitives introduce architectural connectors on the run-time configuration for enforcing the specific activity flows that implement the business rules that apply to the business entities involved in the execution.

3 The coordination dimension

The way coordination aspects can be captured as architectural connectors has been developed in previous publications (e.g. [5]). From the conceptual modelling point of view, they are captured in semantic primitives that we call Coordination Laws.

Coordination Law: captures the way a business rule requires given classes of business entities (partners) to interact; the partners are not identified at instance level: the law identifies the roles played by these partners through generic "coordination interfaces"; the way interactions between partners are coordinated according to the business rule is captured in the form of event-condition-action specifications; auxiliary attributes and operations may also be defined.

Coordination Interface: A coordination interface identifies what is normally called a role; it consists of the set of required services, events and invariants that have to be provided by the business entities to become coordinated as described by the interaction rules of the law.

Given the limited space available, we are going to present a short example of the definition of Coordination Laws, omitting the corresponding Coordination Interfaces.

Example 3.1 For the withdrawal business activity, we may consider different Coordination Laws, each of which captures a different business-rule relating a customer with an account. For instance, one may distinguish a "standard"-withdrawal in which the customer can only request amounts less than the current balance, and a "VIP"-withdrawal under which a credit-limit previously agreed with the bank can be used if the current balance is not sufficient.

At the instance, run-time level, each coordination law is perceived as a "contract" between the customer and the specific account involved in the business activity. Such contracts are external to the components, i.e. the interactions that they impose are not "hard-wired" in the code that implements, say, the bank account, as it would happen if using traditional object-oriented techniques (see [5] for more discussion on these aspects). Instead, coordination contracts are like architectural connectors that can be superposed dynamically over the current configuration to establish the interactions required by the business rules that apply to the business entities involved in a given activity. It is not the purpose of this paper to illustrate the way coordination laws get instantiated to specific contracts between business components. See [4] instead. The fact that, in a given configuration, a customer and an account are bound by a standard or VIP contract, results from the activation of configuration rules that reflect agreed business contracts between business entities and enforce business policies; these are modelled by a different set of semantic primitives.

Coordination Law Withdraw_Standard

partners a : Interface-Account,
            c : Interface-Customer
when c.withdraw(n, a) with a.owner(c) and (a.bal() ≥ n)
do a.debit(n)
end Coordination Law

Coordination Law Withdraw_VIP

partners a : Interface-Account,
            c : Interface-Customer
attribute credit : money
when c.withdraw(n, a) with a.owner(c) and (a.bal() + credit ≥ n)
do a.debit(n)
end Coordination Law
4 The location dimension

In this section we introduce the concepts and constructions that reflect our understanding of location-awareness with respect to activity flows in business processes, with illustrations from the running example. As for coordination, we aim at externalising, as first-class entities, the dependencies that business rules put on the way given business activities are performed at given business locations, what are sometimes called business "channels".

The motivation is, again, one of supporting incremental refinement and compositional evolution. In our opinion, location-dependency should not be "hard-wired" in the code that implements the software components involved in a business information system. Otherwise, the introduction of a new location (business channel) would require every component to be reprogrammed to incorporate the specific rules that apply to the new location. This is a costly, time consuming and error prone process that leaves business knowledge scattered in the code that implements the information system. By supporting a clear separation between computation, coordination and distribution at the architectural level, we are able to make explicit the "contracts" that enforce business rules in each dimension and determine how new business products need to be reflected, separately, in each dimension.

Towards this end, we perceive any communication-aware "business entity" as a (an embedded/physical/software) component or device that may include (hidden) computational activities but, more importantly, can also play different roles in communication-aware modes of composition with other components. Depending on the current location of the communication device, roles may have proper data and operations and—as for coordination—provide services and/or trigger business rules to react to certain events. We handle this new concept of role as Location Interfaces within Location Laws.

One basic difference between coordination and location modelling primitives is that, as expected, the latter involve "locations". It is important to stress that these are "business locations", not necessarily mapped to "unique" network addresses or other physical means of communication. Indeed, as already claimed, our purpose is to bring location-awareness to the conceptual modelling level through architectural primitives that relate them to the other business concerns within the same layer.

For this purpose, capitalising on the experiments developed over CommUnity [8], we adopt a generic data type LOC for modelling the positions of the space of locations that can be adapted to the business domain at hand. The services and triggers that are identified in Location Interfaces are assigned variable that take values in this space, which makes them location dependent.

The location-dependent aspects of business rules are modelled in Location Laws that relate activities performed in different locations as captured through Location Interfaces. The definition of Location Law that we present borrows, once more, from the extensions that we have been developing in CommUnity. Depending on the position of an entity (e.g. communication device), we perceive its topological context through the special observable BT : 2^LOC. This observable account for the fact that communication among components can only take place when they are located in positions that are "in touch" with each other, which is modelled by BT.

Like for Coordination Laws, Location Laws may include proper data types (attributes, operations) and invariants when needed. However, in contrast to coordination rules, besides usual triggers, conditions, and initiation of services, their ECA-like rules also involve:

1. An explicit dependency on the availability of communication (i.e. be-in-touch) between participating location roles.

2. Depending on this availability, either a full composition of all features of participating partners is performed, or just composition of located features are performed. That is, any required composition is postponed until next available communication.

Example 4.1 Taking, like for coordination, the withdraw activity within the withdrawal business process, we will consider two location laws capturing banking at an ATM or at a branch. In Figure 1 we have depicted these two Location Laws, namely LLaw Withdraw_ATM-BANK and LLaw Withdraw_Branch-ATM. Like for coordination laws, the instantiation of a location law superposes a connector (location contract) between the business entities that play the roles involved in the business activity. In the case of the location law Location Law Withdraw_ATM-BANK, these are a customer at an ATM and a bank account at
the bank. According to the location rules of the law, the location contract will always check if the requested amount is less than the maximum value allowed by the ATM. When communication with the bank is available while the customer is withdrawing from the ATM (i.e. $BT(\text{atm}, \text{bank}) = \text{true}$), the contract makes sure that the requested amount is given to the customer (i.e. $\text{Give(amount)}$) and a corresponding charge is debited from the account (at the bank location). However, if there is no communication between the ATM and the bank at that moment, the requested amount is still given to the customer up to an established default value, and the rest of the withdraw transaction is delayed (through the assumed primitive $\text{Delay}$) until communication is again established.

**Figure 1. Location laws for the Withdraw activity.**

```
LLawWithdraw_ATM-Bank
Partners $\text{ratm} : \text{Loc-Int.-ATM-wdr}$, $\text{rbank} : \text{Loc-Int.-Bank-1}$
Locations $\text{atm}$, $\text{bank}$
attribute default : money
operation $\text{min(money,money)}$
Rule: Withdraw_ATM
  when (withdraw(n))
  $BT(\text{atm}, \text{bank})$: with (n < atm.max-amount())
    do atm.give(n) and bank.debit(charge())
  ~ $BT(\text{atm}, \text{bank})$: with (n < atm.max-amount())
    do atm.give(min(n,default())) and $\text{Delay(bank.debit(min(n,default())+charge()))}$
end Location Law

LLawWithdraw_Branch-Bank
Partners $\text{rbranch} : \text{Loc-Int.-Branch}$, $\text{rbank} : \text{Loc-Int.-Bank-2}$
Locations $\text{branch}$, $\text{bank}$
Rule: Withdraw_Branch:
  when withdraw(m)
  $BT(\text{branch,bank})$: branch.take(m) and bank.debit(m)
  ~ $BT(\text{branch,bank})$: skip
end Location Law
```

Notice that, when the customer is banking at a branch, there are less constraints that need to be observed but, on the other hand, less provision for off-line transactions.

These location laws capture only location-dependent aspects of the business rules. At any given configuration, the same business entities will be engaged in location and coordination contracts that, together, dictate how the activity is flowing. This model is briefly discussed in the next section.

## 5 The Coordination-Location based Architecture

As motivated in the introduction, this section proposes an overview of the architectural approach that we are developing and that brings together the three intrinsically related parts of process-oriented information systems—activity flows, Coordination, and Location.

As already mentioned, our approach is activity-oriented, i.e. for each activity within a business process, we identify which are the location and coordination concerns that apply to the business entities involved, and how they are put together to respect the business process logic (e.g. the activity ordering). In general, there is a 0-N correspondence between each business process activity and Coordination / Location Laws. That is, depending on the semantics of each activity, we may have no coordination laws (which is the case of identification in the example) or one or more coordination laws (case of withdraw); and the same for location laws.

Besides this correspondence, which shows how business process activities are intrinsically related to coordination and location laws, we have to emphasize that, depending on the business entities involved in a specific activity, only specific laws apply at each configuration. Determining which laws should apply and, for those that apply, how the business entities play the relevant roles as instances of the interfaces (location and coordination), and how the corresponding contracts connect the entities, is beyond the scope of this paper. See [5] for the case of coordination.
Example 5.1 Figure 2 illustrates the coordination-location architecture for the case of withdrawal using an ATM, with a customer in VIP contract with an account. The component customer@ATM encapsulates the computational aspects required for performing this activity on the side of the ATM, while the component Account@Bank encapsulates the computations that need to be performed at the bank side.

6 Conclusions

We proposed first steps towards a location-aware architecture particularly suited for business processes modelling and dynamic evolution. The architecture is centered around semantic modelling primitives that capture coordination and location concerns.

To consolidate this architecture we are working on case studies in other areas that can help us validate the approach. We are also collaborating with ATX Software, the IT company with whom we developed the Coordination primitives, on the use and methodological aspects of the approach. Extensions to modelling languages like the UML are also being developed at Leicester. Finally, foundational work is also progressing in collaboration with the University of Lisbon.

Acknowledgement

We would like to thank our colleagues P. Kosiuizenko and A. Lopes for their insights and suggestions on the work reported in the paper. We are also grateful to the anonymous referees for their helpful comments.

References