Architectural Handling of Management Concerns in Service-Driven Business Processes

Ahmed Al-Ghamdi and José Luiz Fiadeiro

Department of Computer Science, University of Leicester
Leicester LE1 7RH, United Kingdom

Abstract. To be effective and meet organisational goals, service-driven applications require a clear specification of the management concerns that establish business level agreements among the parties involved in given business processes. In this paper, we show how such concerns can be modelled explicitly and separately from other concerns through a set of new semantic primitives that we call management laws. These primitives support a methodological approach that consists in extracting management concerns from business rules and representing them explicitly as connectors in the conceptual architecture of the application.

1 Introduction

Service-driven business processes have become one of the topical innovations through which organisations hope to cope with ever-changing business requirements. The shift that this new trend represents over typical product-oriented business models is characterised, among other, by much more flexible, loosely-coupled interactions that sustain high-levels of agility and adaptability to change.

From a technological point of view, the response to the challenges raised by such models is relying on the pervasiveness of Internet technologies, namely on Web-services. These are platform-independent components with explicit interfaces tailored to the Web that can be used in combination with others to form large-scale, evolvable business applications. Because the ultimate objective of the service-driven economy is to match customer requests *in-time* and with *optimal* services, either elementary or composed from simpler ones, it becomes clear why the notion of *quality of service (QoS)* plays a central role in this new paradigm [1]. It is not surprising that the notion of Service Level Agreement (*SLA*) has been placed at the heart of such Web technologies [2, 3]. Notions of quality addressed by these efforts include availability, accessibility, accuracy, throughput, and reputation, among others.

However, from the wider point of view of the organisations that adopt such services to promote their businesses, there are levels of quality that are more global and abstract than those that relate to the IT-platform that supports the software applications, which need to be ensured and to which service-driven business processes have to contribute. *Business SLAs* refer to agreements on *how* specific services are offered to their customers. They concern, primarily, the business semantics of such services,

Al-Ghamdi A. and Luiz Fiadeiro J. (2006).
Architectural Handling of Management Concerns in Service-Driven Business Processes.
In *Proceedings of the 4th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*, pages 111-120
DOI: 10.5220/0002500701110120

which cannot be captured through system or application level metrics; they concern business rules that deal with the way inter-enterprise services and their customers can be managed in an optimal way.

Most frameworks addressing QoS and SLAs remain at the infrastructural level provided by proposed IT platforms: an example is SLAng [4], an XML-language defined over the UML for addressing a number of performance/resource-related quality metrics. Instead, business-level notions of quality need to be addressed at a higher-level of abstraction in which they can be formulated independently of any IT platform. This is essential to enable the participation of the wider community of partners in the organisation, or across different organisations, that are concerned with business quality factors (e.g. users, managers, designers, marketers), not just the programmers [5].

In this paper, we take one step in the direction of enabling business concerns to be represented explicitly in the conceptual models of service-oriented applications. We put forward a set of semantic primitives that extend the CCC-architectural approach proposed in [6] with connectors that capture the management aspects of the applicable business rules. More precisely, CCC focuses on architectural connectors that coordinate the interactions between business partners that ensure that given business procedures are followed and policies upheld; we introduce a different kind of connector through which we can superpose the concerns that relate to the way these interactions ensure required levels of management quality such as deadlines, handling and financial aspects of business transactions.

Having this in mind, the paper is structured as follows. The next section presents a running example previously used in literature – selling PCs [7]. The third section recalls how interaction concerns are modelled in [6, 8] using coordination laws. In section 4, we start by showing how management concerns can be approached at the business level and in terms of business rules. Then, we put forward new architectural concepts to extract management concerns directly from business rules. We close by discussing the additional research steps that we are taking for addressing Business SLAs.

2 The Case Study

Our case study deals with a service-driven business process for selling PCs "online" (PcSelling). We distinguish four categories of services involved in this process: customer, provider, shipment, and banking services.

- **Customer services (CS)** provide PcSelling with front-end interactions with customers. That is, CS allow customers to post their requests to buy PCs, to pay their dues, and to cancel/accept offers.
- **Provider services (PS)** are used for returning offers to customers that place requests. They also control delivery and provide key functions for dealing with refunds and penalties.
- **Shipment services (SS)** are used when there is a need for shipping the goods, allowing the shipment method to be selected.
- **Banking services (BS)** accomplish any required payment procedures, including shipment and any necessary insurance costs.

In a typical scenario, when a provider service receives requests for buying PCs from a customer service, it checks for availability and for customer insurance. At the same time, the customer service checks quality-level requirements such as delivery time and payment options. When the different requirements of both parties are agreed upon, an offer is made by the provider service to the customer service.
The business activities involved take the form:

- **Request**: This is the first step in the process during which the customer requests a given number of PCs.
- **Offer**: This activity consists of an offer made from the provider in reply to the customer's request, including details related to management levels such as delivery time and costs.
- **Delivery**: It notifies the provider to start delivering the goods according to agreed business levels of quality.
- **Cancellation**: This business activity may follow if the customer wants to cancel the order or just part of it. Penalties may apply.
- **Shipment**: This activity is launched when customer's location requires shipment instead of collection.
- **Payment**: This activity handles cost-related transactions such as direct payments, refunds, and penalties.

3 Rule-Based Modelling

In order to reach the higher-levels of abstraction required for modelling business processes, interest has grown in adopting business rule-driven approaches[9]. These are understood as "projections of organisations' constraints and declarations of (internal/external) policy/conditions that must be satisfied for doing business". They specify actions to be taken in the event of particular events, including "state of being" changes concerning individuals, infrastructure, consumables, informational resources and business activities in general.

Rule-driven approaches offer a number of advantages that are crucial for coping with dynamically evolving complex business processes. They support the specification of business models independently of the specific processes that happen to be running at any one time. They focus on primary requirements and address business domain descriptions in a declarative rather than an operational way. Applications of rule-driven approaches to web-services composition have also been proposed [10] and [11] but, as far as we know, without including the management aspects that concern us in this paper.

Our architectural approach is also rule-based but takes the separation between business rules and activity flow one step further: it separates the concerns that relate to the way interactions among business entities need to be coordinated to fulfil given business goals from the management concerns that reflect business quality level agreements. The remaining sections detail the way we model each concern separately and how they can be brought together to provide an integrated model of a business process.

4 Coordination Concerns

Semantic modelling primitives have been put forward in [6, 8] that rely or architectural connectors for separating the coordination of interactions between business entities from the computations that entities perform to ensure required services. More precisely, so-called coordination laws and contracts externalize as first-class entities any intra- or cross-organisational interactions between business components. This clean separation permits changes to business rules to be performed at the level that is required without affecting the other aspects.

- **Coordination Laws** capture the way business rules require given classes of business entities (partners) to interact. 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 (ECA) rules.
- **Coordination Interfaces** identify what are normally understood as the roles of a connector. They consist of the sets of services, events and invariants that have to be provided by business entities to become coordinated as described by the rules of the law.

As an example, consider the coordination aspects involved in the request activity: the provider has to check for the availability of the quantity and type of the requested PCs to decide whether to accept or reject the request. The business rule that applies may be informally described as follows:

1. The customer asks for a specific number of particular types of PCs.
2. After checking the availability and types, the provider is entitled to accept or reject the request.
3. By accepting the request the provider designates the request as pending by assigning it a reference and notifies the customer in consequence.
4. By rejecting the request, the provider cancels the request and notifies the customer.

A formalisation of this business rule in terms of coordination laws requires the following interfaces:

```
coordination interface CustReqCI
partner type CUSTOMER
import type  Item,Request
events request(i:Item,Qt:nat)
services cancelled(), accepted(r:Request)
coordination interface ProvReqCI
partner type PROVIDER
import type  Item, Request
services
  makePending(i:Item,Qt:nat,Rq:Request)
  typePrv(i:Item):bool
  availQt(i:Item):nat
```

The coordination interface *CustReqCI* captures the behaviour required of a customer so that it can interact with a provider, namely it produces the event *request* with parameters *i* for the kind of PCs that are requested and *Qt* for the quantity requested. In the coordination interface *ProReqCI*, we require the operations that a provider needs for interacting with the customer: (1) make a request pending – *makePending(i,Qt);* (2) cancel a request – *cancelRequest(i,Qt)*; (3) the kinds of PCs offered by the provider – *typePrv(i)*; and the quantity of PCs currently available for a given kind –*availQt(i)*.

The coordination law that regulates the interaction between customer and provider during the request activity is described as follows.

```
coordination law RequestCL
partners Ct:CustReqCI; Pr:ProvReqCI
import type  Item, Request
attribute ReqNo:Request
when Ct.request(i,Qt) do
  if Pr.availQt(i)≥Qt and Pr.typePrv(i)
  then Pr.makePending(i,Qt)
           and Ct.accepted(ReqNo)
  else Ct.cancelled()
```

Accordingly, when the customer places a request, the provider checks for availability of type and quantity; if available, it assigns the request a number, notifies the customer and makes it pending; otherwise, it notifies the customer that the request cannot go through. More details about coordination laws, how they are instantiated and implemented following the CCC-architecture, can be found in [6, 8].

5 Management Concerns

In this section, we address *business management concerns* that arise between customers and providers at an abstract, domain level, such as business response/delivery time, costs and credibility of business partners.

We propose to describe such management concerns in terms of specific *business rules* dealing with how inter-enterprise activities are adequately managed. In this way, management concerns can be evolved and improved at the business level in conjunction with the relevant stakeholders. This is why we decided to build on the CCC-approach by proposing an additional architectural dimension in which the main characteristics of business management concerns can be identified, specified and analysed as *first-class* entities.

By investigating several process-aware applications and business processes, we have come up with the following characteristics, which we argue will lead to better management of business processes. We do not claim that these characteristics are complete or exhaustive; more investigation on extensions and refinements is still going on.

- **Customer and provider preferences and degrees of trust** are at the heart of service-based business processes. To satisfy customer, preferences have to be carefully addressed. In our case study, when requesting PCs the customer may choose some PC providers over others depending on a number of factors. In addition, the preferences of the provider also have to be considered.
- **Timing issues** are also essential elements when it comes to satisfying both the customer and the provider. One should not confuse business-time issues with system performance-related ones. The latter have been extensively considered in different approaches [5, 12]but do not address the higher business level agreements. In the case study, the customer, when requesting goods, has the right to set a time limit for a reply, as well as a time limit for accepting delivery of the goods. On the other hand, the provider also has the right to set time requirements such as the duration of the validity of any offer, or the time for shipment in relation to the delivery time agreed upon with the customer.

- **History of the state of activities** is essential for dealing with management issues while modelling complex business activities. The aim here is to set appropriate formulae and primitives at the business level, so that appropriate management strategies can be built. In our case study, for instance, the formulas which are of most interest include how many requests or offers have been accepted, rejected or ignored; delivery or payment delays, etc.

In the example, when a customer requests a batch of PCs, he/she may first prefer specific providers over others, possibly depending on previous experience with them or simply because he prefers their products. Secondly, to obtain a quicker response, the customer may set a desired response time to be respected by the provider. From the provider's side, he has to consider his response time limit, which has to be smaller than the requested reply time. The day and the time at which the request is 'posted' by the customer may also play an important role. For instance, dealing with any request during the weekend or outside normal working hours is more expensive. The extracted business rule may then be expressed as follows:

1. The customer asks for a specific number of particular types of PC.
2. The customer has the right to opt for a preferred provider and this may depend on the history of similar operations.
3. The customer also specifies the response time to be respected, which implies that the day and the time of the request are to be included.
4. When the customer's requirements have been fulfilled, the request is considered to be in a managed state.
5. When the conditions are not met, the request is added to the history for assessment in possible future transactions.

Such concerns represent a sort of management-driven *contracts* between different business partners such as customers, providers, and suppliers. This suggests that architectural techniques similar to coordination laws and contracts should be investigated to provide mechanisms for specifying business management concerns. More precisely, we propose to specify such concerns as architectural connectors called *management laws*.

6 Management Laws

As for coordination laws, management laws require interfaces through which one can identify the features (operations and events) that partners are required to provide to engage in a given business activity. A data type is also associated with every management interface to identify the class of instances. With respect to a given business activity, the behaviour governing corresponding management concerns is also modelled using ECA-like rules albeit extended with new management primitives that capture the aspects discussed in the previous section. The general form of a management law is as follows:

```
management  law < law-name>
partners <variables typed with
           management interfaces>
rule <rule-name >
  when <trigger>
  who <conditions on partner preferences and operation histo-
  ries>
```

```
at-time <conditions on time issues>
manage <set of operations>
after <set of operations
```

Under the clauses *when*, *who* and *at-time* we specify the event triggering the concerned business activity, the related management constraints and conditions expressing partners preferences, and the degree of trust using the history of previous operations if necessary. Under the clause *manage* we specify all management actions to be undertaken. Finally, when the triggering event happens but some constraints under *who* and/or *at-time* do not hold, the actions identified under *after* are performed.

With respect to the request business activity, two interfaces are again required, one for the customer and the other for the provider.

```
management interface MaCuReMI
partner type CUSTOMER
import type  Item, Request
events request(i:Item,Qt:nat)
operations
  TmRep(): Time
  RqHrs(): Time
  RqDay(): Day
  PreferListPrv():(ProviderNames)
  HistPrv(): List(Operations)
```

This management interface identifies an event – *request(i,Qt)* – that will be used as a trigger in a law (see below). The customer has also to opt for a list of preferred provider names, captured through the operation *PreferListPrv()*. Additional requirements include the response, *TmRep()*, which the customer imposes on the provider within which he is expected to reply. The days and hours of the request order are also of interest, and captured respectively through *RqDay()* and *OrdHrs()*. Finally, because the history of different operations may also be relevant to the customer this is considered through the operation *HistPrv()*.

```
management interface MaPrReMI
partner type PROVIDER
import type  Item, Request
operations
  NamePrv():Name
  TmRepLm():Time
  DeliverTm():Time
  WrkDays():List(Days)
  WrkHrs():interval(Time)
  ManageRequest()
```

This interface requires a number of operations from the provider: name – *NamePrv()*, and current availability in terms of time for reply – *TmRepLm()*. Further operations are required as to inform the customer of possible time for delivering the goods, denoted by *DeliverTm(),* and of the working days and hours, denoted respectively by WrkDays() and WrkHrs(). Finally, there should be an operation for telling the provider that the management for the activity can proceed – ManageRequest().

A business activity, as expressed in terms of management laws, should initially adhere to the informal meaning of corresponding business rules, and to the management characteristics we identified.

```
management law RequestML
partners MgCt:MaCuReMI; MgPr:MaPrReMI
attribute Tflexple:Time
```

```
when MgCt.Request(i,Qt)
  who MgPr.NamePrv()in
           MgCt.PreferListPrv()
  at-time MgCt.TmRsp()+MgCt.Tflexple()
                      ≤MgPr.TmRepLm()
       and MgCt.OrdHrs()in MgPr.WrkHrs()
       and MgCt.RqDay() in MgPr.WrkDays()
  manage MgPr.ManageRequest()
  after  MgCt.Add(HistPrv,Cancel)
```

As for the coordination law, the management process is triggered by a request from the customer as specified by the *when* clause. The condition part, as already mentioned, is split into two parts:

- **The time part**, under the clause *at-time*, specifies any conditions reporting on timing issues. The request activity allows us to check that the time for reply requested by the customer is not beyond the ability of the provider. Nevertheless, in order to provide flexibility, there must be an allowance for assessing any specific time lapses denoted by *Tflexple*. The day and hour of the customer request should also come within the normal working days and hours of the provider as stated in this management law.
- **The preference and history part** is specified under the clause *who*. As the name suggests, this conditional part concerns any condition regarding the preferences of the customer or the provider, as well as giving a history of operations in the business activity concerned. For the request activity, it is necessary only to check that the name of the provider is among the providers in the customer-preferred list.

The management actions to perform are declared under the clause *manage*. In the case of the request activity, the only action that has to be performed consists of informing the customer and provider "partners" that the request activity has fulfilled all the requirements from the management perspective.
Finally, the clause *after* specifies the actions that need to be undertaken when there is a failure to fulfil all the requirements (whether this is to do with timing, or preference, or history). In the example, it adds the cancellation to the customer's history.

7 Integration of Concerns

Being able to model these two concerns separately, with all emphasized benefits, does not mean that they are independent. The way a business activity is performed within a process emerges from the coordination and management laws that jointly apply to that activity. For instance, at runtime, the way a request is processed is captured not by independent coordination and management partners: both coordination and management interfaces are instantiated by the same business entities. That is, the request is executed by a (run-time) customer service that is an instance of both *CustReqCI* and *MaCuReMI*. In other words, both coordination and management interfaces are instantiated by the same run-time services or components; instantiating coordination and management laws means binding the coordination and management interfaces to services that are running on the current system configuration.

In this example, the event that triggers the request business activity is *request(i,Qt)* in both the coordination and management interfaces instantiated by the customer service. The joint execution of the corresponding coordination and man-

agement rules reflects the *conjunction* of the conditions and the parallel composition of the actions:

```
when CS.Request(i,Qt) do
if PS.AvailQt(i)≥Qt
  and PS.TypePrv(i)
  and CS.TmRsp()+CS.Tflexple ≤PS.TmRepLm()
  and CS.OrdHrs()inPS.WrkHrs()
  and CS.RqDay()inPS.WrkDays()
  and  PS.NamePrv()in
       CS.PreferListPrv.list()
then PS.PendingRequest(i,Qt)
           and PS.ManageRequest
else PS.CancelRequest(i,Qt)
       and CS.Add(HistPrv,Cancel())
```

That is to say, the request activity is performed according to both coordination and management rules, which share the same trigger – *CS.request(i,Qt)* where *CS* is the run-time customer service.

8 Concluding Remarks

With the growing scale and complexity of service-driven applications and business processes, a methodology is required that allows the concerns mentioned above to be handled separately and at an early stage when considering business requirements. Moreover, rigorous approaches are needed for modelling such concerns at higher levels of abstraction.

Towards this end, we proposed an approach based on architectural techniques and separation of concerns that builds on previous work on Software Architecture [6, 8]. In this approach, interaction and management concerns are separately modelled, validated and evolved as architectural connectors. To reflect the semantics of business activities, both concerns are then integrated.

Our immediate plans are to enhance and consolidate the approach with more case studies. The modelling of the whole business processes will also be tackled. For this purpose, different ways of adopting and extending UML-activity diagrams will be investigated. Finally, we are developing an explicit mapping of our business-level architectural approach to web-service management over service-oriented architectures [2, 3].

Acknowledgments

We would like to thank our colleagues Luís Andrade and Georgios Koutsoukos from ATX Software for many useful remarks. This work was partially supported through the IST-2005-16004 Integrated Project *SENSORIA*: *Software Engineering for Service-Oriented Overlay Computers*.

References

1. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2004) "Web Services Agreement Specification (WS-Agreement)", Draft, Global Grid Forum.
2. Casati F., Shan E., Dayal U. and Shan M. C. (2003) "Business-Oriented Management of Web Services", *Communications ACM,* 46, pp. 55-61.
3. Benatallah B., Casati F., Toumani F. and Hamadi R. (2003) "Conceptual Modeling of Web Service Conversations", *LNCS,* 2681, pp. 449-467.
4. Skene J., Lamanna, D. and Emmerich W. (2004) "Precise Service Level Agreements", in *International Conference on Software Engineering, 26, IEEE Computer Society: ICSE 2004,* pp. 179-188.
5. Castellanos M., Casati F., Dayal U. and Shan M. C. (2003) "Intelligent Management of SLAs for Composite Web Services", *LNCS,* 2822, pp. 158-171.
6. Andrade L. F. and Fiadeiro J. L. (2003) "Service-Oriented Business and System Specification: beyond object-orientation", in Kilov H. and Baclwaski K.(eds), *Practical Foundations of Business and System Specifications*, Kluwer Academic Publishers, pp. 1-23.
7. Sahai A., Machiraju V., Sayal M., Moorsel A. v. and Casati, F. (2002) "Automated SLA Monitoring for Web Services", *LNCS,* 2506**,** pp. 28-41.
8. Andrade L. F. and Fiadeiro J. L. (2001) "Coordination Technologies for Managing Information System Evolution", in *Advanced Information Systems Engineering, LNCS*, 2068, pp. 374-387.
9. Kardasis P. and Loucopoulos P. (2004) "Expressing and Organising Business Rules", *Information and Software Technology,* 46, pp. 701-718.
10. Orriens B., Yang J. and Papazoglou M. P. (2003) "A Framework for Business Rule Driven Service Composition", *LNCS,* 2819, pp. 14-27.
11. Charfi, A. and Mezini, M. (2004) "Hybrid web service composition: business processes meet business rules", in *International Conference on Service Oriented Computing ICSOC '04, ACM Press*, pp. 30-38
12. Keller A. and Ludwig H. (2002) "Defining and Monitoring Service Level Agreements for Dynamic e-Business", in *Systems Administration Conference 16,* pp. 189-204.