The Role of Contract and Component Semantics in Dynamic E-Contract Enactment Configuration

Heiko Ludwig and Yigal Hoffner

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract: Dynamic establishment of electronic service contracts necessitates the dynamic configuration of a contract enactment system to either perform or consume the service. The more complex a service is, the more likely it is to be enacted by several components (or subsystems). Flexible configuration of a contract enactment system from components requires a blueprint defining which component type plays which role in the service's enactment and how components are parameterised with contract elements. For this specification we need to understand the semantics of the contract and of the available components as well as rules for a correct composition of components. We investigate how components can be configured and dynamically assembled and propose an extensible service enactment model that provides the basis for component assembly. On the basis of this model the Internal Enactment Specification language is introduced for describing component compositions and parameterisations that can be processed by an automatic configuration mechanism.

Key words: Electronic contract, enactment configuration, deployment, provisioning, component semantics, contract semantics.

1. INTRODUCTION

1.1 Motivation

The increasing availability of computer networks enables companies to offer services that can be bought, consumed, and managed online by service consumers. Examples are network and application hosting services or traditional business processes such as warehousing and parcel delivery. In many

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35658-7_21](https://doi.org/10.1007/978-0-387-35658-7_21)

R. Meersman et al. (eds.), *Semantic Issues in E-Commerce Systems*

© IFIP International Federation for Information Processing 2003

cases, customers expect immediate service availability after completing an order form online. A possible approach for delivering this is to configure the **service enactment systems**, which implement the service delivery and consumption, in advance. This entails knowing all the different likely configurations beforehand. Worse still, this necessitates the pre-allocation of resources that may never be used. The requirement to flexibly configure a service enactment system in a dynamic way points to the need to allocate resources fast just as the need arises. This requires service providers and consumers to be able to automate the configuration of their service enactment.

Dynamic contracting and service enactment configuration have been addressed by the CrossFlow project (www.crossflow.org, Grefen et al., 2000, Hoffner et al., 2000), which sets the context of the work described here. CrossFlow deals with dynamic business process outsourcing on a case-by-case basis. Providers advertise their services on a service marketplace. Service consumers, wanting to commission a particular business process instance to another organisation, search the marketplace for suitable offers. In the course of the ensuing negotiation, the details of the service such as start date, QoS parameters and price, are fixed in a formal service contract. After the contract is agreed upon, both provider and consumer organisations prepare their internal systems for service provision and consumption, respectively. Outsourced business processes are not regarded as atomic entities and service consumers want to be notified about the progress of the process and intervene if necessary, adding to the enactment configuration complexity.

To illustrate the issues of this paper and our solution approach we use the example of a logistics company that offers warehousing and parcel delivery for customers. The company warehouses contain goods stored on behalf of customers. Customers can ask goods to be delivered to particular destinations, and are allowed to monitor and control the process. For our purposes, we ignore the route planning and other aspects and focus on the setup of the system to provide monitoring and control for international transports. This example draws on the logistics scenario of the CrossFlow project (Damen et al., 2000) but is largely simplified to allow focusing on the issues relevant to this paper.

1.2 Flexible and Dynamic Enactment Configuration

There are many factors that make the automation of the configuration of **Contract Enactment Systems (CESs)**, service enactment systems for contractually agreed services, for a dynamically established contract a non-trivial task: *Contracts* are complex. They describe a business relationship and must therefore cover a wide range of issues such as service descriptions, QoS guarantees, liability issues and arbitration. *CESs* are also likely to be

complex. They contain the functionality that an organisation uses to provide or consume the **core service** and also the functionality to administer the business relationship between the contracting parties, e.g. accounting and billing systems. Organisations have to flexibly assemble and configure their CES according to each specific contract.

The key problem is to provide an approach that allows organisations to specify a blueprint for the configuration of a CES that deals with the contingencies of different contracts in each specific case. The CrossFlow project approached this problem by the use of templates for both contracts and Internal Enactment Specifications—blueprints for the configuration of a CES. Figure 1 illustrates the process.

Figure 1. Automatic Contract Enactment System configuration.

An organisation offers services of a particular type using a **Contract Template (CT)** (Hoffner 2000). A CT is a formal specification of a business relationship that contains some open fields that are filled in the course of negotiations to yield a contract. Each contract is complemented by an **Internal Enactment Specification (IES)** that describes how the contract enactment systems is to be configured on the basis of a set of available components or sub-systems (e.g. for freight management or billing). Similarly to a CT, an **IES Template (IEST)** is an IES with some open fields whose value adapts the enactment policy to the needs of a particular contract. A contract and an IES are input to an **Enactment Configuration Manager (ECM)** that selects a set of components and configures the CES as specified.

The specification of an IES and an IEST and their automated processing require understanding the semantics of the contract elements (or CT) as well as those of components that can be deployed in a CES configuration.

1.3 Outline

The objective of the paper is to demonstrate how the IES builds on the semantics of contract elements and components to facilitate an automated configuration process by the ECM. For this purpose we discuss the CES configuration process in more detail next, and derive requirements on the IES language. In the 3rd section we propose a model of dealing with contractual events that provides the basic component semantics while being adaptable to particular service and implementation specifics. In the 4th section we introduce the IES concepts and sketch the IES language. Finally, we compare this approach with related work and draw some conclusions.

2. CONTRACT ENACTMENT CONFIGURATION

2.1 Contract enactment system configuration schemes

CES configuration schemes differ in the use of contract elements and the granularity of available CES components. We introduce the basic configuration schemes and show how they can be combined, building on our logistics example:

1. **Parameterisation:** The CES instance created by the ECM can receive parameters that determine its behaviour according to the specifics of a contract (Figure 2(1)). The parameters can either originate from the contract (Destination, in the example) or from within the organisation, e.g. from backend system (OrderID).
2. **Parameter Translation:** Contract elements may require translation before they can be used as CES parameters (Figure 2(2)). In the example the contract states Paris as destination. However, the delivery component requires a particular airport as parameter. The ECM translates the contract parameter destination into the airport *Paris Orly*.
3. **Selection:** one or more elements of the contract determine which type of CES is to be selected (Figure 3). In the example the service level prime leads to the choice of the *Flight* component rather than *Rail*.

Figure 2. Parameterisation of CES components and parameter translation.

Figure 3. Parameterisation and selection of CES components.

4. **Composition** consists of the selection and parameterisation of multiple components of a smaller granularity than those used in the above two schemes (Figure 4). An important aspect of component-based applications is that components must work together in accordance to a component model that defines valid compositions of components and interaction patterns among them. Thus, composition also includes the assignment of roles to components in the composition and the establishment of communication relationships. In this example the CES consists of three components: the flight delivery component controlling the air transport of the parcel, a billing component, and a communications component that facilitates RMI access by the service customer. The components have been picked from a larger pool and parameterised.

Both choice and parameterisation provide some flexibility to implement contracts. Additional component composition leads to a very flexible mechanism for CES creation.

Figure 4. Configuring a CES from components by composition.

2.2 IES – Role, Expressiveness and Prerequisites

The ECM configures the components to a working CES that matches the requirements of the contract. For each particular contract we need a plan defining which components will be used and how they should be parameterised such that they implement the organisation’s role in performing the contract as intended. Rather than coding this plan into the ECM, we have chosen an explicit representation in a formal language, the IES, to provide flexibility by using templates as discussed in the introduction. Together with the contract, the IES is an additional input to the ECM (*Figure 5*).

The **IES language** must contain constructs to express the following aspects of contract enactment:

- A **definition of the components** to be deployed, their role in the contract enactment and their technical implementation.
- The **parameterisation of the components**. Component parameters are values defined in the IES, references to and translations of contract elements, and internal information from other sources such as our example’s order ID. Component parameters can address different aspects, e.g. component implementation, technical parameters where to instantiate it, and also service-level information such as account numbers.

Figure 5. The role of the Internal Enactment Specification (IES).

- **Involvement of components in contractual events.** Components do not enact the contract autonomously but must work together dealing with contractual events. The most important category of contractual events is interaction among the business partners. For example in the case of a customer sending a message to redirect a parcel to a different address, the transport and the billing components may need to be involved.

The definition of a CES by a composition of components requires the understanding of two underlying models: (1) A model of what can be expressed in a contract, and (2) a component model defining the responsibility of different component types in fulfilling the contract, interactions between components, and valid component compositions.

The expressiveness of CrossFlow's contract model, a service model-based kind of electronic contract, and its corresponding XML-based contract language have been discussed in (Koetsier et al., 2000). The constituent elements of a contract are:

- **Partner description:** A description of the service provider and consumer, including their technical properties such as the addresses of their communication protocols.
- **Service description:** The core of the contract is the description of the service to be performed.
- **Contractual interaction description:** This part describes the possible interactions between the parties while the service is being performed.
- **Description of contractual guarantees:** This describes guaranteed properties of the service.

3. CONTRACT ENACTMENT MODEL

3.1 Structure of the contract enactment model

Components cannot simply be plugged together. They need a structure that defines how they work together to achieve the intended functionality of a component composition. The purpose of the contract enactment model is to define such a structure upon which the IES language is built. The model must be independent of a particular component implementation platform, and independent of service specifics while still representing how components contribute to a contract's enactment.

Formally, a component model needs to address three main issues:

- 1. It defines the available **component types** and their specific properties.
- 2. It specifies the **interaction relationships** between components.
- 3. It defines constraints on **valid compositions** of components to a contract enactment environment.

We propose a three-level component model for contract enactment (Figure 6). The foundation of the model is the **technical level**, which describes the implementation issues of the components of a CES and depends on a particular implementation platform. The **contractual event level** captures the involvement of components in dealing with contractual events. Contractual events are the central concept that underlies the contract enactment model. In our example the main contractual events are management operations by the consumer and status update information originating from the service provider. Components may have a service-specific role defined on the **service-specific level**.

Figure 6. Levels of the contract enactment model.

Each level of the model has its own object types, interaction relationships and constraints on the composition of objects leading to a valid CES configuration. While technical and service-specific levels may vary in different cases—having their particular types, relationships and rules of valid compositions—the contractual event level is a case-independent concept that is the foundation of the CrossFlow IES. The following subsection introduces the object types, relationships and constraints of the contractual event level.

3.2 Dealing with contractual events

On the contractual event level we divide the set of components into communication components, service implementation components within an organisation, and components that link the two classes:

- **Proxy-Gateways (PGs)** implement a communication relationship with the business partner, e.g. for a particular protocol such as RMI.
- **Functional Extensions (FXs)** implement (on a provider's side) and use (on a consumer's side) the service. They can receive messages from business partners and respond to them. FXs can implement different roles, providing different interfaces. An **Actor** implements the main effect of a contractual operation, e.g. in the case of a stop operation on a business process, it actually performs this step. It is also in charge of generating a return value, if appropriate. A **Supervisor** can answer the question whether an operation is admitted at the current state of the service—with respect to the aspect of the service the Supervisor represents. A business process Supervisor could make its decision based on process progress, an accounting Supervisor could decide on the business partner's credit limit, if a contractual event incurs a fee. The third component type is a **Listener**, monitoring interaction and causing secondary effects. A component in this role could measure condition parameters of guarantees or simply perform administrative tasks such as accounting. An FX can implement multiple roles.
- The **Coordinator** is a switch. It routes messages between PGs and FXs.

Each CES consists of a Coordinator, a (non-empty) set of FXs and at least one PG. One of the FXs must be an Actor.

Contractual events are processed by the Coordinator in the following way (*Figure 6*): When an interaction arrives from a business partner connection (in.1), it asks the appropriate Supervisors whether this interaction is admitted at the current state of the service (in.2). If they all agree (conjunction), a list of Listeners are notified (in.3) prior to the forwarding of the interaction to an Actor (in.4). When the Actor performed the according primary effect, another set of Listeners is informed (in.5). Then, a return message is sent back, if necessary. Outgoing interaction (out.1) is dealt with a similar way. After

authorisation of an outgoing interaction by the Supervisors (out.2), Listeners are notified before (out.3) and after (out.5) the interaction is forwarded to a business partner (out.4). This pattern is applied to contractual events in both consumer and provider CESs. The contract enactment model allows us to capture the semantics of components in fulfilling a contract while being able to adapt to particular service types and technical implementations.

3.3 Contract enactment framework

To deploy the contract enactment model for creating CES using the ECM, components need to be built to assume roles in this model. The CrossFlow project developed the CRAFT framework (Ludwig et al., 2001), which is a component framework that covers the contractual event level of the contract enactment model. Technically, components are implemented as Java classes providing role-specific interfaces.

4. INTERNAL ENACTMENT SPECIFICATION

The IES serves the purpose of describing how a particular contract is implemented in using a set of components according to the contract enactment model. The IES consists of four main elements:

- A description of the context of the IES, i.e. to which contract it relates and the role of the IES-authoring organisation in the contract.
- A specification of the components that need to be instantiated to enact the contract. This includes a definition of the components' roles according to all relevant levels of the contract enactment model.
- The components' parameterisation.
- The implementation of contractual events.

Next we describe each of these elements and give examples of their specification in the XML-based IES language. The complete document type definition of the IES language can be found in (Ludwig, 2000).

4.1 Context information

The context of an IES is simple and consists of two elements:

- A reference to the contract which it implements. The syntax of the IES language is illustrated in the following XML tag.

```
<ContractName ContractID="contract1" />
```

- The contractual role that the organisation assumes in the contract, provider or consumer.

```
<Role RoleType="consumer" />
```

4.2 Component specification

We need a definition of the contract enactment environment that is relevant for the enactment of the specified contract, i.e. a specification of all components to be deployed. This comprises each component's potential role in dealing with contractual events, its service-specific roles, if applicable, and its technical implementation information, needed by the ECM to instantiate the component. This specification is needed because it is generally not easily extractable from a component framework and the component implementations. We illustrate component definitions using the component type FX, which has the richest set of parameters:

```
<Components>
  <FX FXID="deliveryflight"
    Actor="yes"
    Supervisor="Yes"
    Listener="no">
    <Technical>
      <ClassName>org.crossflow.fx.wfstatefx
    </ClassName>
    ...
  </Technical>
  <ServiceSpecific>
    <Name>DeliveryFlight</Name>
    ...
  </ServiceSpecific>
</FX>
...
</Components>
```

This example defines an FX on the various levels of the contract enactment model. The tag FX defines its contractual event roles using the attributes Actor, Supervisor and Listener. There is a Technical and a ServiceSpecific section that are used for level-specific role information. The Technical section defines the implementation of the component as a Java class. The component name DeliveryFlight represents the service-specific role of the component. Any other role information regarding the service-specific level can be supplied in the ServiceSpecific tag. PG and Coordinator are defined similarly in the components section.

4.3 Component parameterisation

Each component may need additional configuration information in the form of parameters, as discussed in the 2nd section. Parameters can be contract elements, translations thereof, or any additional configuration parameter from a source within an organisation. Parameters can correspond to the technical or the service-specific level. The specification of component

parameters is part of the component specification of the IES. The following example illustrates the parameter description:

```
<Components>
  <PG PGID="rmipg">
    <Technical>
      <classname>org.crossflow.rmipg</classname>
      <myURL>contract.providerURLID</myURL>
      <partnerURL>contract.consumerURLID</partnerURL>
    </Technical>
  </PG>

  <FX FXID="billing"
    Listener="yes" Supervisor="no" Actor="no"
  <Technical>
    ...
  </Technical>
  <ServiceSpecific>
    <Name>Billing</Name>
    <BillingMode>contract.BModeID</BillingMode>
    <AccountNo>12345</AccountNo>
  </ServiceSpecific>
</FX>
...
</Components>
```

The example illustrates the specification of the RMI communication component, a PG, and the billing component of our scenario, an FX. In addition to its class name, the PG has two technical parameters: `myURL`, the URL of the service, and `partnerURL`, the communication address of the contract partner. Both parameters are taken from the contract. The billing FX has two service-specific parameters, `BillingMode`, taken from the contract, and `AccountNo`, directly specified within the IES (12345).

Parameters that are taken from the contract XML document are referred to using ID attributes. XML elements can have an attribute associated with them that must be unique ID in a valid XML document. The referral mechanism of the IES language makes use of this. References to contract elements are described by prefixing an ID of the contract document with the string "contract" as shown in the example. The reference is resolved by the ECM at configuration time. At configuration time all parameters are passed to a component as an XML string using the tags that are specified in the IES. This requires that the author of an IES understand the semantics of the contract's elements as well as of the component parameters.

4.4 Contractual events

The specification of the management of each contractual event (specified in the contract) defines which Supervisors, Listeners and Actor, or PG, to

involve. Two main event types exist: inbound and outbound interaction. The following example explains the event specification:

```
<InboundOperationImpl name="changedeliveryaddress">
  <Actor FXRef="deliveryflight">
    <Supervisor FXRef="locfx">
      </ InboundOperationImpl>

  <OutboundOperationImpl name="statenotification">
    <PGDef PGRef="rmipg">
      </ OutboundOperationImpl>
```

In the case of the inbound operation, we define an actor, the Delivery-Flight FX, which implements the operation, and a single supervisor, in this case the CrossFlow Level of Control FX, that verifies whether the operation is permitted at the current stage. The outbound operation refers to a notification of state updates that has been defined in the contract. In this simple interaction only the PG is involved to send the message to the contract partner.

4.5 Interpretation by the ECM

The ECM interprets the contract and the IES and configures the CES accordingly. This involves a number of steps:

1. Parsing the contract and the IES.
2. Checking and resolving the IES's references to contract elements.
3. Verification whether the component selection and typing comply with the component model on each level of the contract enactment model.
4. Instantiation of the components according to their technical specification.
5. Supplying all technical and service-specific parameters to all components.
6. Configuring the coordinator component with the contractual interaction specification.

This requires that a particular ECM be aware of the contract enactment model *on all levels*, i.e. that it identifies invalid component compositions on service-specific, contractual event and technical levels.

5. CONCLUSION AND RELATED WORK

This paper discussed the role of contract and component semantics in the automated configuration of a contract enactment system (CES) as needed for dynamic e-business systems. CrossFlow supports the automatic configuration of CESs based on an Internal Enactment Specification (IES). In the IES, a system designer specifies the composition of a CES from components. The concept of the IES provides an additional level of indirection that results in increased flexibility of organisations. Using IESs, organisations can offer a

single internal service type through multiple different contracts. Service consumers can purchase services on the basis of different contracts and relate them to their internal system. Automatic configuration of CESs requires understanding the semantics of contract elements and of the components of the CES, in particular for the choice of components and their parameterisation. We introduced a three-level contract enactment model that separates service-specific issues of a contract and the technical implementation from the contractual events. This model is extensible and can be applied to various services and implementation platforms. Based on the contract enactment model, the IES language provides constructs to describe which components to deploy, how to parameterise them—including referral to contract elements—and how to involve them in contractual events. Contract and IES are processed by the enactment configuration manager that creates a CES. A number of components according to the contract enactment model and an ECM for the process management setting of CrossFlow's scenarios have been implemented in the project.

If service providers want to deploy this technology, they must implement components according to the contract enactment model (or wrapper existing systems), create or re-use a contract template and then use the IES language to describe which components to instantiate and how to parameterise them to implement the CES that corresponds to the contract.

The contribution described in this paper needs to be put in the context of related work both in industry and the research community. Recently, a number of products and service offerings have been developed that allow organisations to integrate business processes across organisational boundaries, among them Microsoft's BizTalk infrastructure (www.biztalk.org) and Extricity's Alliance product (www.extricity.com). Also, a number of standard bodies work on message definitions to facilitate cross-organisational interaction, e.g. RosettaNet (www.rosettanet.org) and the eCo framework (www.commerce.net). The basic approach is to standardise a set of messages for a particular purpose on an industry-wide basis. Software supporting the standard can be implemented and integrated with existing systems. However, this approach is not sufficient in a flexible environment, where service relationships (contracts) are negotiated case by case.

There are some proposals for formal contract languages intended for direct interpretation by a CES, e.g. (Caminda, 2000) and (Grosz et al., 1999). This approach has some restrictions: It is impossible to define a general contract language that captures all relevant information, and to build the corresponding enactment system. The tight coupling of contract and enactment system limits the possibility to define a contract enactment strategy. Other approaches, such as tpaML (Dan et al., 2000), ebXML CPA (www.ebxml.org), and the Web Service Description Language (www.wSDL.org)

allow the flexible specification of the communication aspect of a service relationship but do not support a flexible specification approach for a CES.

Based on the contract enactment model and the IES language, further work needs to be done in dealing with IES templates. The current work focuses on a particular IES. IES templates are supposed to be processed by the contract establishment system. An IES template language could include constructs expressing how to instantiate an IES depending on values of a particular contract and thereby provide a mechanism to automate the filling of IES templates in the contract negotiation process.

REFERENCES

- Caminada, M.: Towards a formal model for contract execution. In *Proceedings of the 5th Intl. Workshop on the Language-Action Perspective on Communication Modelling (LAP 2000)*. Aachen, Germany, 2000.
- Damen, Z.; Derks, W.; Duitshof, M.; Ensing, H.: Business-to-business e-commerce in a logistics domain. In *Proceedings of the ISDO'00: CAiSE Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing*. Stockholm, 2000.
- Dan, A.; Dias, D.; Kearney, R.; Lau, T.; Nguyen, T.; Parr, F.; Sachs, M.; Shaikh, H.: *Business to business integration with TpaML and B2B protocol framework (BPF)*. IBM Research Report, RC 21863, Yorktown Heights, 2000.
- Grefen, P.; Aberer, K.; Hoffner, Y.; Ludwig, H.: CrossFlow: cross-organizational workflow management in dynamic virtual enterprises. In *International Journal of Computer Systems Science and Engineering*. (15)5, CRL Publishing, London, 2000.
- Grosof, B.; Labrou, Y.; Chan H.: A Declarative approach to business rules in contracts: courteous logic programs in XML. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC-99)*. ACM Press, New York, 1999.
- Hoffner, Y.: Supporting contract match-making. In *Proceedings of the 9th Int'l Workshop on Research Issues in Data Engineering – Information Technology for Virtual Enterprises (RIDE-VE'99)*. Sydney, Australia, 1999.
- Hoffner, Y.; Ludwig, H.; Gülcü, C.; Grefen, P.: Architecture for cross-organisational business processes. In *Proceedings of the Second International Workshop on Advanced issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*. IEEE Computer Society, Los Alamitos, 2000.
- Koetsier, M.; Grefen, P.; Vonk, J.: Contracts for cross-organizational workflow management. In *Electronic Commerce and Web Technologies. Proceedings of the First International Conference (EC-Web 2000)*, Springer-Verlag, Berlin, 2000.
- Ludwig, H.: *Internal Contract Representation. CrossFlow deliverable D6c*. La Gaude, 2000.
- Ludwig, H.; Hoffner, Y.: CRAFT: A framework for integration facilitation in cross-organisational distributed systems. In: *Proceedings of the 9th EUROMICRO Workshop on Parallel and Distributed Processing (PDP'01)*. IEEE Computer Society, Los Alamitos, 2001.
- Schuster, H.; Georgakopoulos, D.; Cichocki, A.; Baker, D.: Modeling and composing service-based and reference process-based multi-enterprise processes. In *Proceedings of CAiSE 2000, Lecture Notes in Computer Science 1789*, Springer-Verlag, Berlin, 2000.