RZ 3081 (# 93127) 12/07/98
Computer Science/Mathematics 15 pages

Research Report

Modeling Customer Service Control using the CoCS Approach

Heiko Ludwig

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

IBM Research Division
Almaden · Austin · Beijing · Haifa · T.J. Watson · Tokyo · Zurich

Modeling Customer Service Control using the CoCS Approach

Heiko Ludwig

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

Electronic commerce systems currently focus on providing sales and payment. This contribution adds another dimension: the customer's electronic control of a purchased service. Today's business process modeling approaches do not provide adequate abstractions of the concept of "service". This article proposes the CoCS framework and modeling approach to represent and analyse how a customer can control the performance of a complex service by an organisation on his or her behalf. The CoCS approach encompasses a model of a service that includes the intervention of customers during the service performance, a means of description for the structure, and the effects of a control operation of a customer. With this approach, automation of service delivery can be planned and its efficiency enhanced.

1 Introduction

Providing customers with an electronic means to control a service that is performed on their behalf in an electronic commerce environment requires analysis and description of the service to be supported. At the moment, performance of services bought electronically is usually not well supported by electronic means. Today's systems for electronic commerce focus on sales and payment activities related to the electronic trade of goods and services. Organisations can choose from a variety of online shop-building tools such as IBM's Net.Commerce or Microsoft's Commerce Server and can rely on electronic payment standards such as SET [1]. However, only a few services beyond order and payment exist that offer a means of monitoring service performance. Example of services that do provide this are Federal Express (http://www.fedex.com/, as of 13 March 1998) and UPS (http://www.ups.com/, as of 13 March 1998), which allow customers to track the delivery of parcels over the Web. However, they offer no control of the ongoing service performance. That is, customers cannot stop the delivery or change the delivery address.

One of the reasons for the currently poor support of customer control during service performance is that there is no explicit notion of customer intervention in ongoing service processes provided by business modeling approaches such as ARIS [2] or LOVEM [3] and object-oriented approaches such as UML's use cases [4]. These approaches do not provide adequate means to express the conditions and effects of customer intervention. This paper takes a first step in that direction. The objective here is to investigate ways in which customer control during service performance has an impact on the ongoing performance and to provide a means to describe the circumstances and effects of customer control. This paper is organized as follows:

- In the next section the notions of service and service performance will be investigated and discussed.
- Subsequently a general framework of service performance will be introduced that shows how customer intervention during service performance fits into the picture.
- The general framework is the basis for introducing the CoCS (Control of Customer Service) approach that provides means to represent the structure and the effects of customer intervention during service performance.
- Finally it will provide an outlook on how the results of the CoCS approach can support the design of a system that allows customers to control the services they have bought. It concludes with outstanding issues.

2 Complex Services – A Customer's Point of View

The notion of service is used in a rather broad sense in this article: A service encompasses any activity done by one party (service provider) for the benefit of another party (service consumer) from the latter's point of view. This means that a service can include obvious service-like activities such as a haircut or the execution of a financial transaction as well as the act of delivering real goods such as books. In terms of granularity, we consider a service to be a separate entity if it is perceived as such by the service consumer and it is (or at least could be) paid for separately.

The life cycle of a service relationship between a service provider and a service consumer can be divided into three phases of interaction (see Fig. 1) as has been done in other approaches [5, 6]:

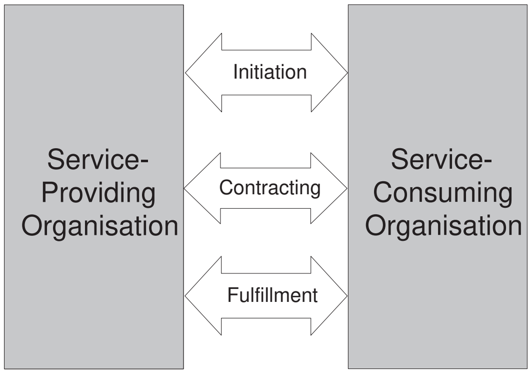

Figure 1: Phases of interaction between service and service consumer.

1. In the *initiating phase* the partners exchange information about themselves and the service offerings of the provider.

2. The *contracting phase*, if successful, results in the agreement of both parties that a service of a specific quality will be performed and at a given price.

3. In the *fulfillment phase* of the interaction, both organisations fulfill their obligations as defined in the contract. In most cases this means that the service provider performs the service, and the consumer pays the price.

Services to be performed according to a contract in the fulfillment phase can vary substantially in terms of size and complexity. There are small, simple services, such as faxing stock price information to customers on request, and there are “larger”, complex services, such as providing insurance coverage or a telecommunications infrastructure for an international company. The size and complexity of a service can also vary from the different points of view of the service consumer and the service provider. From a service provider’s point of view, a service is complex if it is performed in many interdependent steps or activities that “produce” the performance and must be co-ordinated. We call this the *internal complexity* of a service. From a service consumer’s point of view, however, the complexity of a service depends on his or her perception of the internal complexity. If the service consumer can see the various different service-performing activities—maybe because they contribute substantially to the overall service from his point of view—and can participate in the co-ordination of these activities we attribute high *external complexity* to this service. Participation in co-ordination by the service consumer means in this context that a service consumer is given means to influence the way the service is performed during its “runtime”.

Considering these examples, the stock-price faxing service is not very complex, neither from the provider’s nor from the consumer’s point of view. Once service fulfillment begins, the information is being faxed, there is not much co-ordination to be done, at least not on a business level. Considering the example of a parcel service, the fulfillment of the service requires that a parcel be transported along the entire route from sender to receiver. If this is made transparent to the customer—as is the case with “tracking” Web sites of major parcel-delivery services—the service is internally and externally complex, even if the possibility of control by customers is still missing. There is no means for a customer to influence the parcel delivery once it is on its way. In contrast, traditional postal services might have the same internal complexity, but this is not visible to the customer. The service is externally simple.

There is a particular class of services that relies heavily on the ability to provide consumers with the means to intervene during the performance of the service. These are insurance-like services that provide abstract "coverage", which is only useful to a customer if he or she is provided with the means to make a claim—to intervene in the service performance—if the insured event occurs in a defined period. As long as the customer makes no claim, no activity is performed in the service-providing organisation to "deliver" the coverage. However, part of this service agreement between an insurance company and the insured is also what happens if the insured event does occur. Usually the insured will report the event to the insurance company which, in turn, settles the claim. The settlement of the claim is again an activity that is beneficial to the insured—a service in itself, according to the initial definition.

As we have seen, the potential of consumers to intervene in a service gives rise to uncertainty concerning the execution plan of service-performer activities. It might not even be known at the start of the service performance which activities will have to be carried out to implement the performance of the service as specified in the service contract.

3 General Service Performance Framework

To represent the structure and impact of customer intervention in service performance we first need a more general view of a service on which we can build our modeling constructs. The central subject of investigation of this model is the service to be performed. To whom, or what, i.e. human or application, the activity is assigned for performance is irrelevant in the analysis. This becomes relevant, of course, once we proceed to the design of an actual system. Initiating and contracting interactions will not be represented, only the service that has to be fulfilled according to the prior contracting phase.

3.1 Outside View: Interaction between Service and Service Consumer

Four kinds of interactions can occur between a service s and its service consumer (see Fig. 2):

- The service delivers its *performance* to the service consumer. We have already seen that this performance can be rather abstract, such as insurance coverage, or very concrete, such as parcel delivery. The service performance of a servicing activity is Perf_s.

- The service consumer's part of the fulfillment is the *compensation* Comp_s. In most cases the compensation will consist of a payment.

- The service-consuming activity can submit a *control operation*. This encompasses operations to start the service's performance, stop it, or set a parameter of the service. The set of control operation types of a service s is CO_s.

- Finally a service can notify its consumer, e.g. about its state or progress, which in turn can trigger control operations by the consumer. The set of *notification types* is Not_s.

Consider the example of the parcel service: A control operation is "Change delivery address", which sets the parameter "address" that is used for the execution of the service, i.e. the delivery of a parcel to a different destination. For the insurance service an address change of the insured or a claim are examples of control operations. Customers might be notified, for example, about results of claims or expiration dates of their insurance policies.

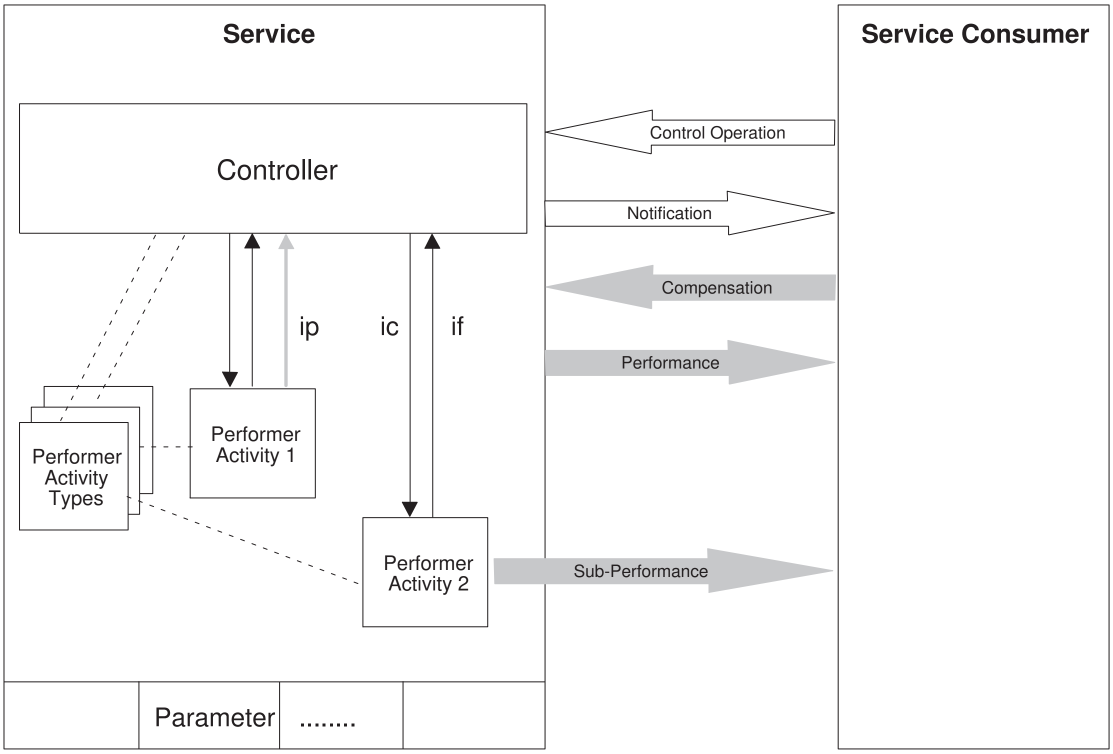

Figure 2: View of service performance.

In our model a service "delivers" its performance and is compensated once (gray arrows in Fig. 2). During one runtime there might be several types of control operations available, each of which might be executed several times. If we consider the insurance example it might happen that the insured makes several claims while his insurance policy is in force. The same applies to notifications.

3.2 Inside View: Elements of a Service and Their Interaction

As pointed out in Section 2, the distinction between the control of service performance and the performance itself leads to the structure of a complex service. In this respect, a service (s) consists of (see Fig. 2):

- a controller (c),
- a set of performer activity types (PAT) that can be used to deliver the service,
- the actual set of performer activities (PA), and
- a set of parameters (PARAM) of the service that are agreed upon during the contracting phase and which the controller uses as execution parameters.

The controller is the component that steers the foreseen progress of the service performance and receives the control operations $co \in \mathrm{CO}_s$ from the service consumer. A service-control algorithm emerges that also decides which measure has to be taken, given that a particular control operation has been invoked by the service consumer.

A performer activity contributes to the performance of the service Perf_s to the service consumer. As we saw in the previous section this might be perceivable for the service consumer

(Performer Activity 2 in Fig. 2) or just an internal step (Performer Activity 1). In many cases several different performer activities might be necessary to deliver Perf_s. The controller activity creates new performer activities according to its control algorithm. The controller and a performer activity interact by exchanging internal control and feedback operations (ic, if in Fig. 2. A performer activity can deliver a given some sub-performance (e.g. the transport leg) directly to the service consumer or/and it can produce an internal performance ip for the controller (e.g. accounting, checking, etc.).

The way a performer activity behaves is defined by a performer-activity type. Every performer activity $p \in \text{PA}$ is built according to a performer-activity type $pat \in \text{PAT}$. A performer-activity type describes the kind of performance it delivers, the set of internal control operations IC it accepts, and the set of internal feedback information IF it can send back to its controller activity.

The circumstances of the service are defined in the set of service parameters PARAM. For example, in the case of parcel delivery this could be the sender's and receiver's address; in the insurance example this could be the expiration date.

3.3 Controllers

A controller receives two kinds of input: control operations as specified in the set of control operations CO from the service consumer and internal feedback from its performer activities. The controller has three options to react to this input:

1. It can create a new performer activity according to an activity type description to perform another part of the service.
2. It can perform an internal control operation ic on a performer activity in order to change its behaviour.
3. It can generate a notification to its service consumer.

Which ever option the controller activity chooses depends on the current state of the service and on the control algorithm. To control services in the real world, a large variety of implicitly or explicitly formulated algorithms is in use, but we find two basic patterns of service controllers:

- The *event-based controller*: The controller activity waits a (possibly pre-defined) period of time. If a specified event occurs during that period, a performer activity is created that delivers part of the service to the service consumer. If no event occurs during the entire period, the service has been delivered perfectly.

  A typical example of this service control pattern is insurance coverage. Even if the insured event does not occur, the coverage was "delivered" to the service consumer at all times. A claim by the insured is an event that can trigger the delivery of another part of the service, e.g. a payment of damage.

- The *process controller*: A process controller works according to a—pre-defined—order of steps to be performed. Each step represents the execution of a performer activity that contributes to the service. There might be parallel execution of performer activities or alternative branches in the process definition that underlies the control algorithm.

  There are also many examples of services that are controlled according to this pattern: when one books a voyage, the first service one receives might be a flight, then a hotel room, and then a return flight.

The difference between these patterns is that the event-based controller is driven mainly by control operations of the service consumer, whereas the process controller is triggered mainly by the progress of the process, i.e. by feedback information of the performer activities. These two patterns of service controllers relate to two different kinds of service: The process controller steers a service in a narrow sense—activities that are performed for the benefit of a service consumer—whereas an event-based controller implements an option on the performance of an activity that will be triggered by a service consumer.

In many cases, a controller of a complex service encompasses both elements. Therefore we represent a controller as consisting of an event-based controller and a process controller. Figure 3 illustrates their relationship.

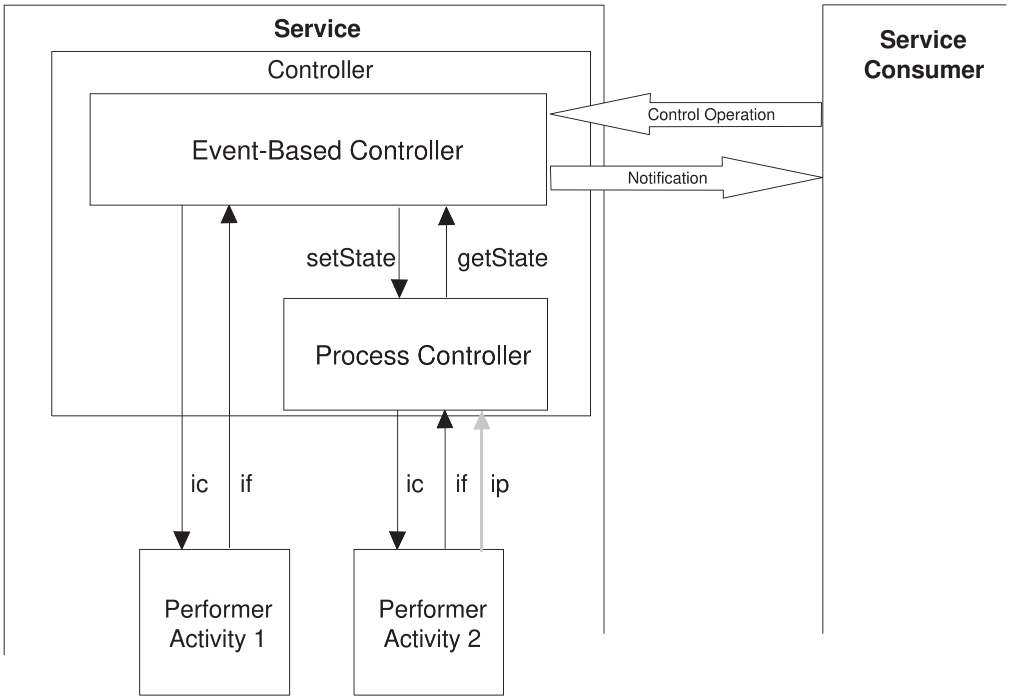

Figure 3: Controller consisting of an event-based and a process controller.

The process controller supervises the planned progress of the service performance and starts and interacts with the performer activities that deliver the standard service (Performer Activity 2 in the figure) —without customer intervention. The event-based controller monitors the process controller and receives control operations from the consumer service. Depending on the state of the process controller, the service parameters, and the control operation, it decides what to do: Create a new performer activity (Performer Activity 1), notify the service consumer, and/or modify the state of the process controller.

Let us consider how changes of delivery addresses in the parcel delivery scenario would fit into that model. The event-based controller could work according to the rule: If the parcel did not yet reach its final destination (state of the process controller), set the service parameter delivery address to the new value and ask the process controller to consider the new parameter. Otherwise notify the service consumer that the parcel has already reached its destination. In the case of an option-like service, such as insurance, the focus is on event-based control.

3.4 Complex Performer Activities

From the description of some of the examples in the previous sections it has already become clear that performer activities can be internally complex. For example an insurance claim, i.e. a performer activity of an insurance service, is usually divided into various subtasks and involves several employees of an insurance company. Performer activities of a service can, recursively, consist of controller activities, performer activity types, and performer activities (see Fig. 4). If this complexity can be perceived by the service consumer—who has been provided with his/her own control operations and has received pertinent notification—this internally complex performer activity is also an externally complex sub-service to the service consumer.

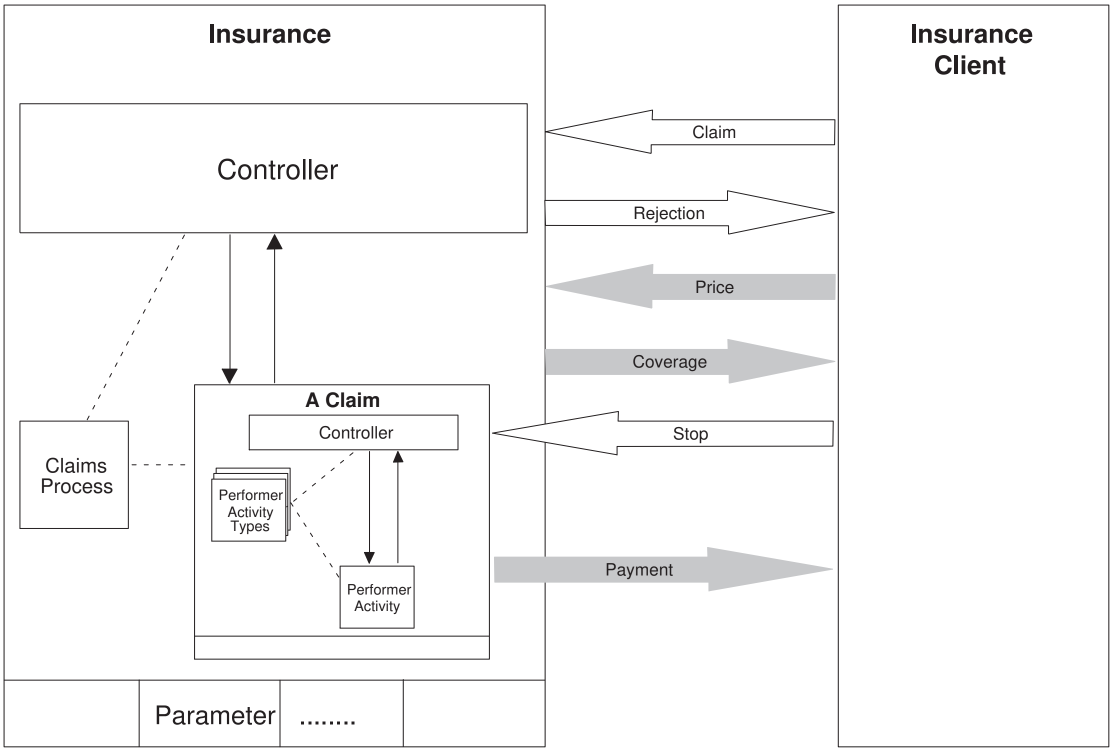

Figure 4: Performer activity as a complex sub-service.

In the example outlined in Fig. 4 the claim is a externally complex sub-service of the service insurance. It provides not only its own sub-performance (payment) that implements the abstract service "coverage" of the parent service "insurance", it also offers the control operation stop to the service consumer. From the service consumer's point of view this operation is only available as long as the sub-service is performing. When the sub-service is finished it is no longer possible to stop it, of course. The performer activity type of a sub-service can be considered its service type.

In general, services can be nested to an arbitrary depth. It is an issue of designing the service to what degree the service consumer should be given insight into the ongoing service performance and what means of control should be provided. The more insight is provided, the more control operations can be offered.

3.5 Requirements for Modeling Client Intervention

The general framework of service performance as described in this section gives us a structure of the elements of service provision. In particular it shows that customer control of service performance can be seen as separate from the control of the "routinely" ongoing service performance.

Faced with the task of designing a service, we already have a choice of means to describe a process controller. Process modeling approaches such as ARIS [2], SOM [6], or LOVEM [3] address this issue. The results of such process designs are implemented in today's operational systems such as logistics applications. In some cases the process controller is implemented using a workflow management system, which can enact process models that are specified in a very detailed way [7].

We wish to provide a means to analyse and design the way service consumers can interact with a service. To do so we have to describe two issues in particular:

1. What is the structure of a service and its sub-services regarding the interaction with the service consumer?

2. What effect do control operations have on sub-services and on the main service?

In addition the means of description should take into account that it should *complement* existing business process modeling approaches rather than attempt to provide a replacement.

4 CoCS Modeling Approach

The CoCS modeling approach provides a means to describe a service structure and an event-based controller's behaviour in a diagrammatic way.

4.1 CoCS Service Structure Diagram

The purpose of the CoCS Structure Diagram is to provide an overview of the interaction of a service type with its service consumer. Figure 5 shows the elements of this type of diagram. Each of the symbols of the diagram denotes an element of the framework as described in the previous section.

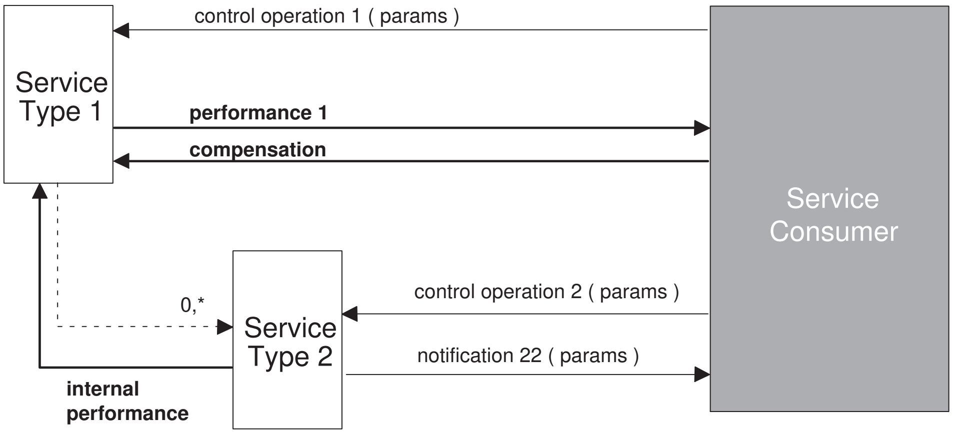

Figure 5: CoCS service structure diagram.

In Fig. 5, *service types* are represented as white boxes. The term service type is used as defined in the framework description in in the previous section. The *containment relationship* is represented by the dotted arrow from Service Type 1 to Service Type 2. Next to the arrow, the cardinality of the containment relationship is given in min,max-notation similar to ER diagrams or object-modeling approaches such as UML's class diagrams [4]. For instance, the cardinality of (0,*) means that an instance of Service Type 1 can contain zero or any number of instances of Service Type 2 during its performance.

Service consumers are represented as gray boxes. If necessary, several service consumers can interact with the same service.

Between service types and service consumers there can be four kinds of interactions as elaborated in Section 3:

Performance: Between each service type and a service consumer there can be at most one performance relationship. The top-level service type has to have a performance relationship. If a sub-service has no performance relationship with a service consumer, it has one to its super-service provider. This sub-service is represented separately because it provides another interaction to the service consumer that is valid only during its existence.

Compensation: Compensation is the inverse interaction to the performance. At least one compensation relationship should exist between a service consumer and a service type, but not more than one per service type. If only the top-level service has a compensation relationship, this is equivalent to a "flat rate" payment, independent of the usage of sub-services. If only sub-service types have compensation relationships, the cable channels' pay-per-view scheme might serve as our example.

Control Operation: Every service type can receive an arbitrary number of control-operation relationships from service consumers. A control operation is specified by its name and parameters.

Notification: As the inverse relationship to control operations, an arbitrary number of notification relationships can be defined from a service type to a service customer. A notification relationship is also identified by its name, its parameters, and the type of information it carries.

If the interaction between service types and service consumers is represented in this form of diagram, the interaction structure is captured. What remains is the description of the dynamics of the interaction: Which control operation has an impact on which sub-service and therefore modifies the currently usable interactions?

4.2 CoCS Service Behaviour Diagram

The purpose of CoCS service behaviour diagrams is to capture the effect of control operations. A diagram expresses in which situation a control operation yields which impact on the rest of the system. We use the notation described in Fig. 6.

The main part of the diagram, i.e. the box that describes the behaviour of an operation *within* the service, is divided horizontally into two parts: The upper half of the diagram describes the preconditions to be met for a control operation to be applied. A conjunction of such conditions is called a *situation*. The lower part of the diagram shows the effects of a control operation in the given situation.

There are four sources of conditions that determine a situation:

Process Controller: The current state of the ongoing regular process, e.g. "not yet finished". Such a condition refers to a description of the process controller expressed in terms of the language that is used for it.

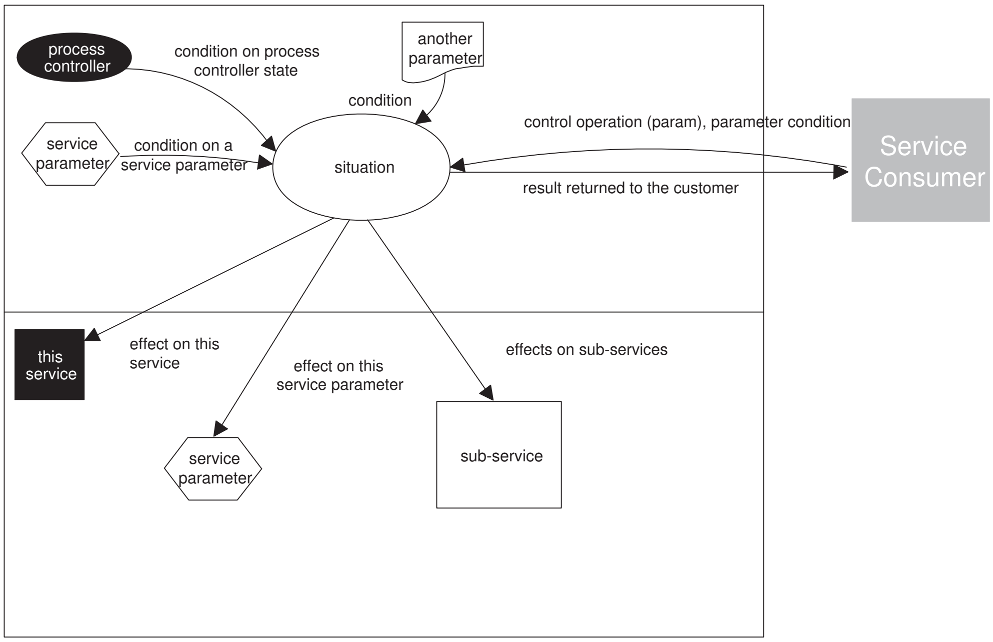

Figure 6: CoCS service behaviour diagram.

Service Parameter: The parameters that have been agreed upon during the contracting phase, e.g. the expiration date of an insurance policy or the delivery address of a parcel service.

Parameters of Control Operation: Control operations can carry parameters. The effect of the control operation might also depend on their value.

Other: Any other entity beyond the scope of the service can also be used as a parameter of a condition. The entity "day of the week" is needed as input, for example, if one wishes to express that a control operation is not available on Sunday.

The various shapes represent the different types of source for a condition. A control operation might be applied in several situations. Each situation leads to a different effect. For pragmatic reasons we assume that there is always a situation "otherwise" that applies if the other conditions fail. This is only the case if the situations are mutually exclusive. Conditions are defined in a textual way at the edges that lead from the source of the conditions to the situation. The service consumer is represented outside of the main box. It is not part of the service.

An operation can cause effects on several other model entities:

Sub-service: A very important effect is on the sub-services of a service. A control operation might entail the creation of a new instance of a sub-service or stop its performance. The effect of a control operation is always on particular instances of a service, not on a service type. In the diagram it has to be specified which particular instances are affected.

Service parameter: Agreed service parameters could be modified—recall the delivery address example.

Process controller: A control operation can affect the performance of the service itself, e.g. it can stop it.

The effects are described along the edges that lead to the affected objects. Using this graphical notation we can express arbitrary pre-conditions of situations and their effects in this situation. In addition, its graphical language provides an expressive means of communication.

4.3 Example

Let us apply the CoCS approach to a simplified example of a well-known service: Home insurance usually covers the loss of personal belongings such as furniture and clothes in case of fire, burglary, or other events. The service that the insurance customer receives is the coverage of a risk as defined in the insurance policy. As long as the policy is in force the policy holder can "control" the service performance by using a number of operations, three of which are:

1. A claim can be submitted in the case of an insured event, which can entail compensation for the lost property if justified.
2. The insured amount can be modified to reflect changes of the value of personal belongings in the insured home.
3. The policy can be cancelled.

A claim is carried out in the insurance company as the claims process. It consists of several steps. It delivers a perceivable performance to the insurance client, the settlement of the claim, i.e. the payment of the damage (or its refusal). During the execution of the process, insurance customers also want the opportunity to take part in its control. In particular, customers want to be able to stop the claims process, e.g. if stolen goods are recovered by the police, and to get information about its progress. As service providers want to be able to control this performer activity, it is again a complex service according to this model.

One step of the claims process is the on-site assessment of the damage. Although this step does not provide a perceivable performance to the client (assuming that the mere assessment is of no use to him or her), it requires interaction between service and service consumer. The insurance client is notified about a planned assessment date and is given the means to reschedule the visit. Figure 7 shows a service structure diagram of the example.

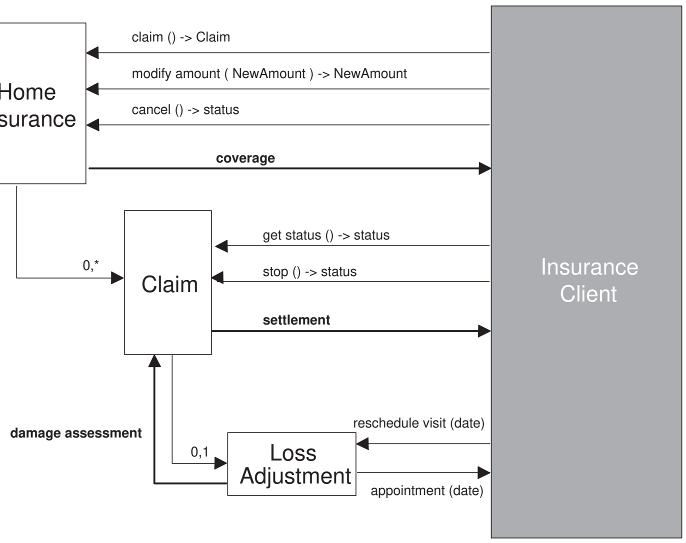

Figure 7: Service structure diagram of the home insurance.

The following sequence of diagrams shows the specification of the control operation in the service structure diagram as service behaviour diagrams.

- Figure 8 specifies the cancel operation. This is applicable if the insurance contract has not yet expired and not on Sunday (just to give an example of an external parameter).
- The pre-condition and effects of the claims operation are specified in Fig. 9. This is applicable if the policy has not yet expired.

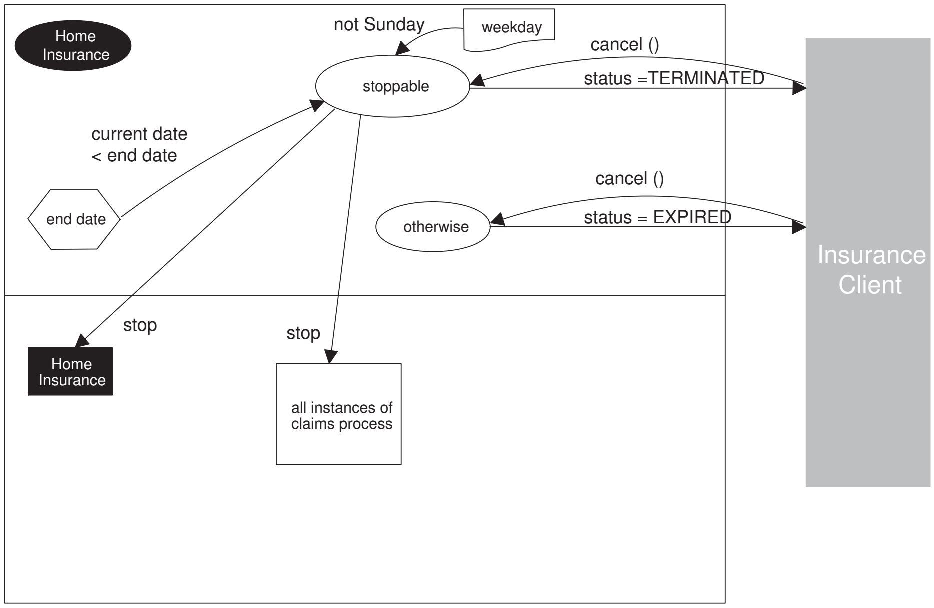

Figure 8: Service behaviour diagram of home insurance: cancel operation.

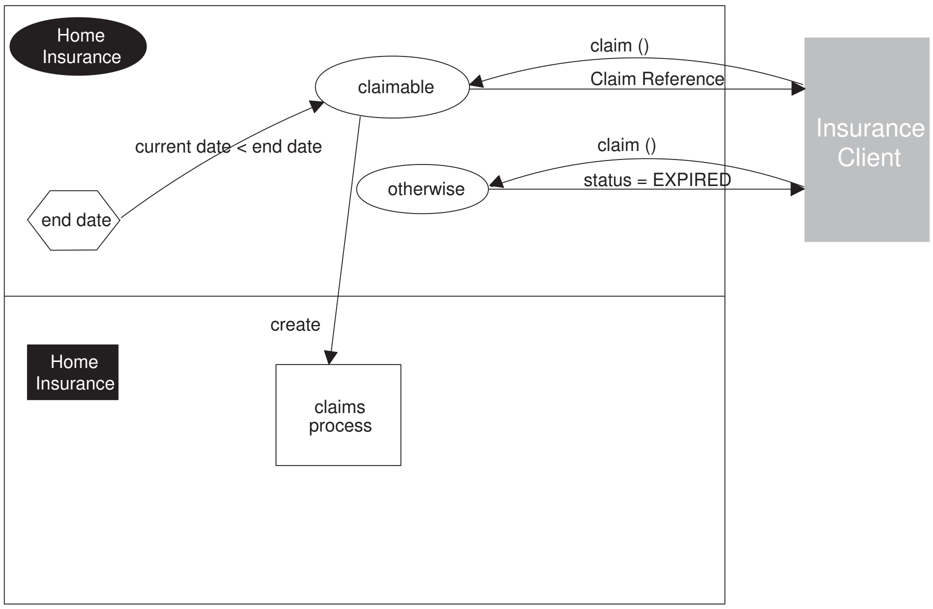

Figure 9: Service behaviour diagram of home insurance: claims operation.

- Figure 10 specifies the modify-amount operation. It is applicable if the insurance contract has not yet expired and the new amount is less than 100,000.
- Figure 11 represents the behaviour of the control operations of the claims sub-process. In addition to the other examples, the situations "stoppable" and "non-stoppable" refer to a state of the process controller. It is assumed in this example that one of the steps of the process is "assess damage", the completion of which determines whether the claims process can still be stopped by the insurance client.

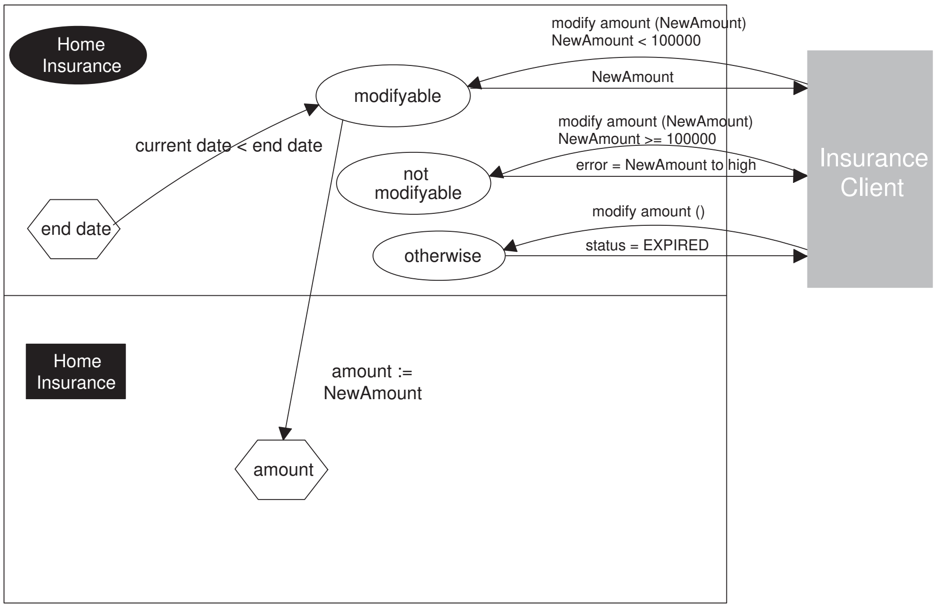

Figure 10: Service behaviour diagram of home insurance: modify amount.

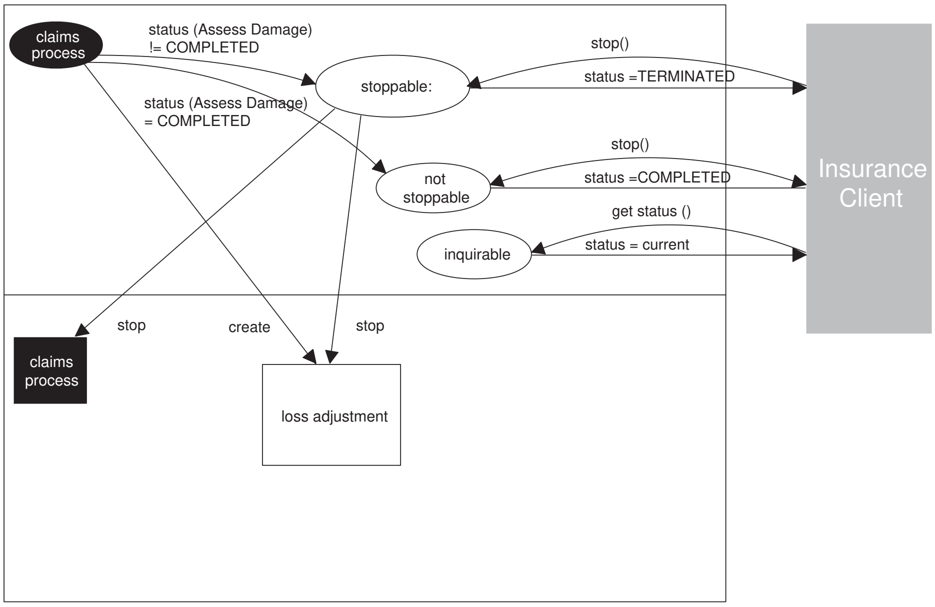

Figure 11: Service behaviour diagram of claims: all control operations.

The control operations of the sub-service "loss adjustment" can be specified in a similar way. In this example the top-level process did not have a "routine" process that was ongoing because its performance is insurance coverage. Therefore, the state of the process controller is irrelevant for control operations. The claims sub-service, however, is performed as a process such that the effects of control operations of the service consumer depend on the state of its process controller, as shown in Fig. 11.

5 Results of Modeling and Paths to Implementation

What is achieved by having represented customer interaction using the CoCS approach? A CoCS model captures the *service structure*—which type of service is a sub-service of another—and with that implicitly structures the interface that a service presents to its consumer. The service behaviour diagrams primarily specify the *pre- and post-conditions of the customer's control operations*, though the specification is informal. In addition, the diagrams define the *interaction of the event-based controller with the process controller*, i.e. the co-ordination of the "routine" process. Moreover, we can derive which *service parameters* should be accessible to the service customer and which *service-external parameters* have to be monitored to evaluate the pre-conditions of operations.

The outcome of an analysis of offered services is a good basis for a design process that leads to the implementation of an event-based controller—supplementing a process-oriented infrastructure. This calls for a decision process to choose which control operations should be given to the customer using which interaction channels. The choice of the interaction channels depends on who, or what, carries out the activity associated with the interaction channel, which is beyond the scope of the framework introduced previously. In addition to existing traditional channels such as telephone call centres, branch offices, or mail—all suitable if activities are carried out by humans—today's network infrastructure, particularly the Internet, adds two kinds of channels that are suitable for electronic submission of control operations:

1. A channel for a human service consumer. In many cases, in particular in a consumer services setting, the service consumer is completely non-automated and carried out by one person. This person wants a graphical user interface to review status and submit control operations. Today, this is readily done by WWW-enabled applications.

2. A channel for an automated service consumer. If both parties of the service relationship are businesses one might find that the service consumer and the controller of the service are not carried out by humans, but by applications such as order management or workflow management systems [7].

Depending on the system's type, other channels such as EDI [8] or Interface 4 of the Workflow Management Coalition [9] are more appropriate. By providing new channels for control operations, service providers might be able not only to offer a better way to implement current control operations to customers, new controls might also enhance the service level for customers.

6 Conclusion

This paper describes the CoCS framework and modeling approach. CoCS provides a new view of customer control of services in addition to current business modeling approaches.

The framework introduces a structure of a service that takes into account that customers of a service want to be able to control its performance while it is ongoing. According to the framework a service consists of a controller that creates and controls performer activities that actually deliver the performance of the service to the customer. Interaction during service performance is done using control operations by the service provider or notifications that are sent by the service. The service provider consists of two parts, a process controller that coordinates the routine procedure of the service performance and an event-based controller that deals with the interaction.

On the basis of this framework, a diagrammatic notation has been introduced that can be used to specify the structure of a service with respect to its interaction with the service consumer. In addition it can express the pre- and post-conditions of control operations, defining the behaviour of an event-based controller and its interaction with the process controller.

However, the CoCS approach is not intended to replace other modeling languages, but to complement means to describe the control of processes that are already well established, such as ARIS [2], SOM [6], and LOVEM [3] as mentioned above.

Many aspects of the CoCS approach are still unclear. How is the exact relation between an event-based controller and a process controller? How can the CoCS descriptions be transferred into a more general-purpose object modeling approach as UML to enable a more seamless transition to designing an actual system? These issues are left to future work.

References

[1] MasterCard and Visa: SET secure electronic transactions protocol, version 1.0 edn. (May 1997): Book one: Business specifications; Book two: Technical specification; Book three: Formal protocol definition; available from http://www.mastercard.com/set/.

[2] Scheer A.-W. Architecture of integrated information systems. Springer. Berlin. 1992.

[3] IBM Corporation: Business process modeller—user's guide. 2nd edn. Böblingen. 1997.

[4] Quatrani, T. Visual modeling with rational rose and UML. Addison–Wesley. Reading. 1998.

[5] Schmidt B. Requirements for electronic markets architecture. Electronic Commerce 1997; 7:3–6.

[6] Ferstl O, Sinz E. Modeling of business systems using the semantic object model (SOM)—A methodological framework. In: Bernus P, Mertins K, Schmidt G (eds). Handbook on architectures of information systems—international handbook on information systems. vol. I. Springer. Berlin. 1997.

[7] Workflow Management Coalition: Terminology & glossary. Document No WFMC-TC-1011. 1996.

[8] United Nations Economic Commission (UN/EC): Electronic data interchange for administration, commerce and transport—Application level syntax rules (ISO 9735). 1991.

[9] Workflow Management Coalition: Workflow standard—interoperability; abstract specification. Document No WFMC-TC-1012. 1996.