# Requirements Engineering Techniques for e-Services

Jaap Gordijn[1], Pascal van Eck[2], Roel Wieringa[3]

24 January 2007

**Abstract**

E-services are deeds, processes and performances that are exchanged commercially and provisioned via the Internet. Such e-services rely on web-service technology as an implementation platform. One of the key problems in developing innovative e-services is to create a shared understanding of the e-service at hand, and to analyze whether the e-service is commercially viable and technically feasible. Complicating factors are that, because of innovativeness of the service at hand, the service is at most vaguely articulated at first, and that many different stakeholders representing different enterprises and different interests are involved, thus creating misunderstanding and confusion. In contrast to normal service development, an e-service is highly Information Technology (IT)-intensive by definition, so an e-service from a commercial point of view should be properly aligned with its web-service enabled implementation. In this chapter, we present an approach, based on requirements engineering techniques, to first understand and analyze the e-service, and second to develop a blue-print for a web-service based implementation. We take a multi-perspective approach that includes a commercial value perspective, a process perspective, and an information systems perspective. This chapter is based on two earlier conference papers [17, 26].

## 1 Introduction

In the past decade, several standardization bodies as well as software vendors have attempted to facilitate e-commerce and value chain automation by proposing standards for cross-organizational application integration, such as standards for description, coordination and composition of *web service*s. These efforts are still in full swing. Example proposals include BPSS [10], BPEL4WS [3], WSCI [5], and WS-Coordination [7], to name only a few. The aim of these activities is to set a standard for a new generation of software components that are able to deal with the specific problems of cross-organizational application integration [1]. In this chapter, we consider the problem of how to actually use this technology. How does one identify requirements for cross-organizational integration, and how are these requirements transformed into a web-service implementation?

[1] Free University, Computer Science, De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands. gordijn@cs.vu.nl

[2] University of Twente, Department of Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands. p.vaneck@utwente.nl

[3] University of Twente, Department of Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands. r.j.wieringa@utwente.nl

In a cross-organizational setting, web-service technology is used to implement *e-services*. E-services are similar to normal commercial services, but now the Internet is used as a user interface or a channel to interact with customers [33, 21, 27]. The notion of *services* is well studied in Economics; they are deeds, processes and performances [32]; additionally they are activities of a more or less intangible nature that normally take place in interactions between the customer and service employees and/or physical resources or goods and/or systems of the service provider, which are provided as solutions to customer problems [18].

e-Services are different from normal services in the sense that e-services rely heavily on web-services and related technologies as an implementation platform. So, developing e-services always results in a serious software engineering effort. It is then important to be able to clearly articulate functional and non-functional requirements, to be satisfied by a web-service implementation. To do so, first the e-service *itself* should be understood well. This understanding is not only about the inter-organizational business processes (the *how*), but also focuses on the comprehension of the business *value* created for participants involved, related to the e-service at hand. In this chapter, we propose requirements engineering techniques that contribute to such understanding of the e-service at hand. Additionally, we show that the requirements specification for an e-service serves as a bridge between business considerations and web-services. Once an e-service is better understood and specified, it is relatively straightforward to design and develop web-services supporting the e-service.

It is important to understand that our requirements engineering approach is intended to be used during the very first phases of e-service development. During these phases, the e-service idea itself is not clear, including which enterprises are going to participate, the way these enterprises create profit with the e-service, as well as the required inter-organizational business processes. Additionally, e-service exploration should be done within a reasonable time frame, say a few weeks, to satisfy time-to-market requirements. Consequently, the techniques we propose are lightweight and easy to use.

Another issue that a requirements engineering approach for e-services has to deal with is the inherent multi-disciplinary nature of the e-service development process. We see e-service development ranging from business value proposition development to software implementation; so many different stakeholders (e.g. CEO, CIO, marketers) are involved in a first exploration of the e-service. Therefore, our approach contains a series of viewpoints to be developed and facilitates in making transitions between viewpoints smoothly.

We illustrate the use of requirements engineering for e-services with an industrial strength case study outlined in Section 2. One of the contributions of requirements engineering is the idea of taking multiple requirements viewpoints. In Section 3, we present three of such viewpoints for e-services. Then we elaborate on these viewpoints. The first viewpoint is the business value viewpoint, and is dealt with in Section 4. Thereafter we introduce the process viewpoint in Section 5. Our last viewpoint, the information system viewpoint in presented in Section 6.

## 2 Running Case Study: The Amsterdam Times

We illustrate our approach using a case study concerning the *Amsterdam Times*, which publishes a newspaper. The *Amsterdam Times* has an already existing subscriber base. The proposition of the *Amsterdam Times* is to offer its subscribers several e-services, such as accessing news articles on-line, surfing on the Internet, and email. In this chapter, we focus on the idea to offer subscribers an on-line news article archive only. This case study is taken from the consultancy practice of one of the authors (Gordijn).

For the correct understanding of this case study, it is important to mention that this study was carried out a few years ago. At that time, still many people accessed the Internet using the Plain Old Telephone System (POTS), by dialing their Internet Service Provider (ISP). During exploration of this idea, it became apparent that the commercial basis of the idea is to use a *termination* fee to finance the on-line article service. In this context, *termination* is picking up the phone when you are called. When a caller calls a callee, the telecommunication network sets up a connection path from caller to callee. When the callee picks up the phone, the termination point of this connection is realized. If an actor (here the *Amsterdam Times*) is willing to cause termination of a large quantity of telephone calls (e.g. because people want to access interesting content made available by this actor), most telecommunication operators are willing to pay the actor for that. This price as paid by the telecommunication operator per realized termination is called the *termination fee*. Because the *Amsterdam Times* has a large subscriber base, it is capable of generating a large number of terminations. The stakeholders involved in exploring the e-service were not capable of articulating the idea this way initially. The elaboration process presented below helped them in doing this.

## 3 Requirement Viewpoints for Developing e-Services

The task of the e-service requirements engineer is to match business processes of a set of business actors to consumer needs in a market, in such a way that the result is a sufficiently detailed specification to design a web service-based specification. (A consumer can be a business, i.e. a legal person, or it can be a natural person.) To make this task manageable, we structure it according to the following viewpoints.

- Taking the **value viewpoint,** we produce three descriptions of the e-service.
    - The *value hierarchy* identifies the top-level consumer need and allocates this to services of economic value to be produced by the business actors.
    - The *value exchange graph* refines this by identifying the activities in which these services are created or exchanged by the business actors. This graph can be seen as a shared discussion object, and can be used to generate profitability sheets stating whether enterprises involved are expected to generate profit with the execution of the e-service idea.
    - *Profitability sheets* quantify the value exchanges for each business actor.

- Taking the **process viewpoint**, we describe inter-organizational business processes and intra-business tasks.
    - A *process decomposition hierarchy* decomposes the interactions between businesses into processes.
    - A *task hierarchy* decomposes each process into tasks to be performed by business actors.
    - A *business process* shows the interacting tasks that are assigned to individual enterprises. These interacting tasks show interactions *between* enterprises as well as *interaction* within enterprises.
- Taking the **information systems viewpoint**, we describe a workflow-like specification to be used by some enactment engine and the web-services required during execution of the workflow.
    - A *WSDL* specification of the web-services.
    - A *BPEL-WS* specification of the coordination of web-service invocations.

We explicitly distinguish these different viewpoints because e-service projects are characterized by many different stakeholders, such as business strategy oriented stakeholders, business process developers and IT-biased stakeholders. These stakeholders have different concerns that should be addressed relatively independent from each other. Otherwise, the requirements engineering process tends to become unfocused and not very productive.

So, in such cases, requirements engineering theory proposes to develop various viewpoints on the service, which are grounded in differences in skills, responsibilities, knowledge and expertise of stakeholders [12]. Such viewpoints deal with the aforementioned multi-perspective problem by decomposing complicated requirement issues into self-contained viewpoints, which can be addressed and decided on relatively independent from each other. With respect to e-service development, the value viewpoint relates to CxO-type stakeholders and marketers, whereas the business process viewpoint is the domain of process (re-)designers. Finally, the information system viewpoint is closely related to the IT staff of participating enterprises.

We present no fixed sequence of writing the different descriptions listed above. However, there are two orientations associated with these viewpoints.

- *Divergence.* Taking the value viewpoint, we explore possible business models underpinning the economic sustainability of an e-service. The focus is on creating new possibilities for value creation, and consequently (initially) a series of possibilities has to be explored. The outcome should be a few e-services that seem to be commercially viable.
- *Convergence.* Taking the business process and information system viewpoints, we identify realistic means to realize the e-service identified from the value viewpoint. The focus is on assessing feasibility of the processes and information systems, both from an economical and technical perspective. By doing so, we obtain a *global*, multi-viewpoint, blueprint of the e-service to be developed, which then can be detailed in a sub-sequential requirements engineering process. So, our approach aims at finding convergence in many potentially

interesting e-services ideas, to arrive at an economic sustainable and focused e-service. Detailing such an e-service is not part of this chapter.

## 4 Value viewpoint

### 4.1 Value hierarchy

The value viewpoint is described using the *$e^3$-value* methodology [13]. This methodology has been successfully applied by the authors and others in consultancy projects done for the music, ISP, news, and energy industries.

One of the lessons learned from the *Amsterdam Times* project is that easily understandable description techniques are needed for the exploration of an e-service. Persons are involved with no background knowledge in conceptual modeling techniques at all and with no time nor inclination to learn these techniques. Such persons need at least to be able to *read* models constructed using our notations. *Making* the conceptual models themselves is done by trained business analysts. So, to allow for an easy understanding, all our notations are simple.

We start elaborating an e-service idea with the elicitation of a **value hierarchy.** Figure 1 shows a value hierarchy for the on-line article idea. The numbers are used to be able to refer to parts of the hierarchy later on. They are not part of the notation.

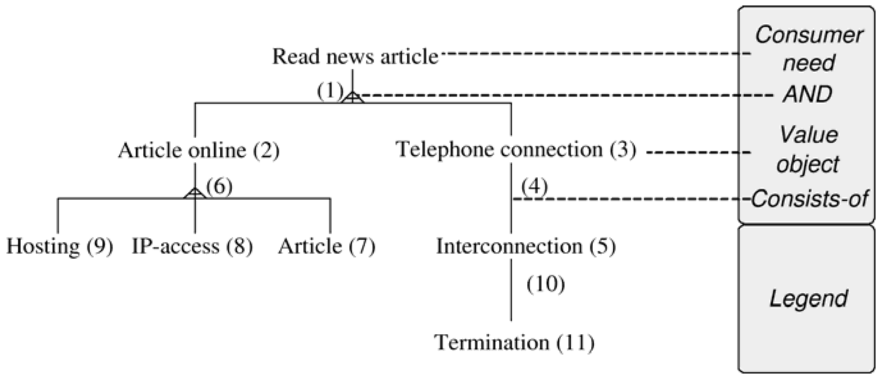

Figure 1: Value hierarchy showing the objects satisfying a consumer need.

The value hierarchy says that to satisfy the need to read a news article, we need an online article and a telephone connection. The on-line article can be provided if there is an article, a hosting service that stores the article, and an IP access service to the stored article. A telephone connection can be provided if we have a number of interconnections and a termination at the consumer. Note that the further we elaborate the hierarchy, the more design choices we make. Such design choices occurred during the execution of the online article project. For instance, the hierarchy proposed in this chapter supposes that a *read news article* need can be decomposed into an *article online* object and a *telephone connection* (decomposed in *interconnection* and *termination*), each to be delivered by a separate actor. Telecommunication companies refer to this as call termination. Another possible hierarchy supposes that the *telephone connection* is deeper into the hierarchy, as a sub-part of the *article online*. This is referred to as call origination (see [15]). Then delivery of an *article online* by a specific actor consists of the article itself, *and* the telephone connection needed to deliver the article. Whether or not to see *telephone connection* as a part of an *article online* is a business design choice; if the connection is a part of the article, the fee to be paid for the connection (just regular telephone ticks) is included in the price for the article, whereas in the other case a user of the service will be charged separately for the connection.

In general, a value hierarchy is a rooted a-cyclic directed graph whose root represents a consumer need. Starting with a consumer need increases the chance that a product is really wanted by a consumer [25]. The other nodes of the graph represent value objects used to satisfy this need. A **value object** is a good or service of economic value to some actor. In this chapter, we focus on the ’service’ interpretation of value object.

The edges of a value hierarchy represent the *contributes-to* relationship. The reverse relationship is called *consists-of.* An AND-node represents the fact that all children are needed for the higher-level one and an OR-node represents that fact that only one of the children is needed.

The leaves of the value hierarchy are the boundary of our value descriptions. We know that one or more actors can produce these leaf objects against known expenses, so in order to elaborate the e-commerce idea, we do not need to decompose these leaf objects further. The value hierarchy is an important tool to relate the satisfaction of a consumer need (the intended benefit) to activities already performed by the business actors (expenses made to create the benefit).

Value hierarchies are similar to goal hierarchies known from requirements engineering (RE) [4, 9, 31]. Both are means-end hierarchies. The difference is that the nodes in a value hierarchy represent value objects to be produced or exchanged between business actors, whereas the nodes in a goal hierarchy represent goals to be achieved. Often, goal hierarchies are developed for single business actors, whereas value hierarchies are always developed for multiple business actors. Finally, a value hierarchy always starts with a consumer need, whereas a goal hierarchy typically starts with a business mission.

## 4.2 Value exchange graph

A value exchange graph shows which actors are involved in the creation and exchange of the value objects shown in the value hierarchy. The constructs which make up such a graph and their semantics have been defined extensively in [13].

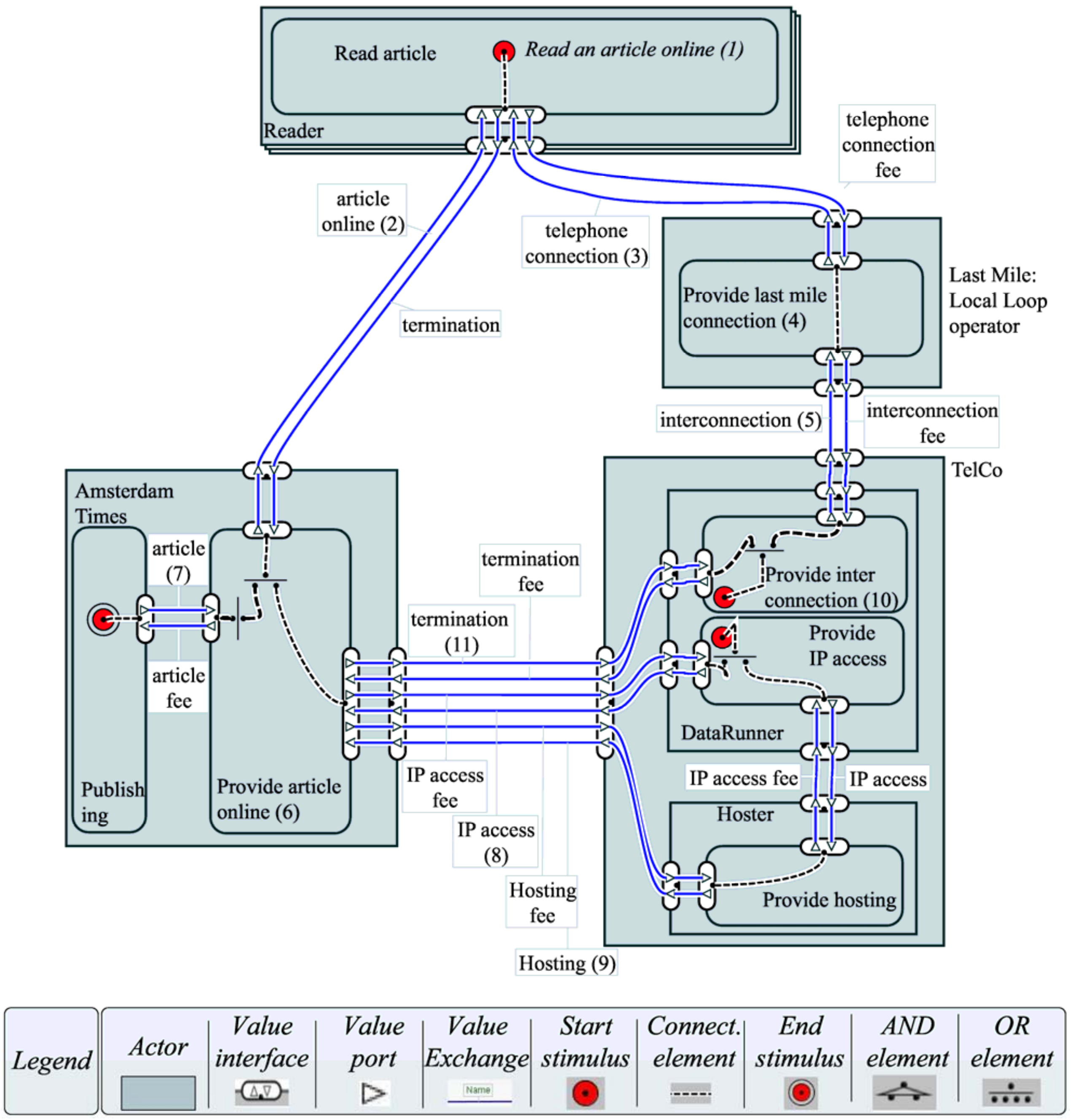

Figure 2: The on-line article service offered by *Amsterdam Times* is funded by termination fees to be paid by the *Telecommunication consortium*.

Figure 2 shows a value exchange graph for our running example. Like a value hierarchy, the value exchange graph represents a number of design decisions. The graph of Figure 2 shows that there is a set of consumers called *Readers*, and that these exchange value objects with the *Amsterdam Times* and with an actor called the *Last Mile*. These in turn exchange value objects with a complex actor, a *Telecommunication consortium*, which consists of two actors, *Data Runner* and *Hoster*. Inside each actor node, one of more value activities are shown as rounded rectangles. Some of the exchanged value objects are numbered. This is not part of the notation. The numbers correspond with the numbers in the value hierarchy. We now explain the value exchange graph in more detail.

An **actor** is an entity perceived by itself and by its environment as an independent economic (and often also legal) entity. An actor makes a profit or increases its utility by performing activities. In a sound, sustainable, value exchange model *each* actor should be capable of making a profit. Actors are represented by rectangles with sharp corners. Sets of actors with similar properties, called markets, are represented by stacked rectangles.

To satisfy a consumer need, or to produce a value object for others, an actor should perform a value activity, for which it may be necessary to exchange value objects with other actors. A **value activity** is an operation that can be performed in an economically profitable way by at least one actor. It can be seen as the act of service provisioning and is depicted by a rounded rectangle. An important design decision represented by a value exchange graph is the decision whether a value object is to be obtained from other actors by means of a value exchange, or to be produced by means of a value activity by the actor itself. This reflects e.g. decisions on outsourcing, and decisions about the optimal size of an enterprise [30].

A **value exchange**, depicted by an arrow, shows that actors are willing to exchange objects of value with each other. Value exchanges are between actors, or between value activities performed by actors. So in the example, a telephone connection is exchanged between the *Last Mile*, and a *reader*.

Each value exchange graph expresses economic reciprocity. We assume that our actors are rational economic entities that are only willing to offer a value object if they acquire another value object in return that is of reciprocal value. Such a reciprocal value objects need not necessarily to be obtained from the same actor who delivers/obtains the other object. Also note that reciprocal exchanges say nothing about the time frame they should occur; all we say is that such exchanges must all happen or none at all.

Reciprocity is shown by value interfaces and value ports. A **value port** is a willingness of an economically rational actor to acquire or provide a value object. A **value interface** is a collection of value ports of an actor that is atomic. By this we mean that an actor is willing to acquire or provide a value object through one port of a value interface if and only if it is willing to acquire or provide values through all ports of the interface. This models economic reciprocity. For example, Figure 2 shows that a *reader* is willing to offer a telephone connection fee and a termination to its environment, but wants in return for that an on-line article and a telephone connection to deliver the article.

Note that the requirement of reciprocity causes us to introduce value objects not mentioned in the value hierarchy. The reason that these reciprocal objects are not mentioned in the value hierarchy is that their introduction is a design choice. Different elaborations of the value hierarchy contain different choices.

In most cases, value interfaces of actors are identical to value interfaces of activities performed by the actors: they exchange the same objects. For these cases, we only show the value interfaces of the activities and not of the actors. However, sometimes, we explicitly need to express an actor's value interface. For example, Figure 2 shows that the *Telecommunication consortium* has a complex value interface that is built up from simpler value interfaces offered by activities in the *Data Runner* and *Hoster*. This is called **bundling.** The *Telecommunication consortium* offers IP access, hosting and termination to the *Amsterdam Times* as one bundle for specific pricing conditions. So, it is only possible to obtain these objects *in combination* in return for the fees mentioned in the diagram as part of the bundle. This is because *Data Runner* and *Hoster* co-locate equipment at the same physical site and therefore can offer hosting and IP-access for a lower tariff compared to the alternative that equipment is located at different sites.

A value exchange graph shows which businesses are involved in a value proposition and what they exchange of value in order to satisfy the consumer need [14]. It does not tell us which processes are performed to realize each value exchange, and it does not tell us in which order the exchanges take place. In general, each value exchange can be put into operation in different ways. These choices are to be made in the process and information systems viewpoints.

A value hierarchy and one or more corresponding value exchange graphs are usually developed iteratively, starting with the value hierarchy. In order to use a value hierarchy for the design of a value exchange graph, note the following.

- The value objects in the hierarchy are the input or output of a value activity.
- The *consists-of* relationships between value objects in the hierarchy indicate a value activity in the graph. This activity produces a value object by using other value objects. An AND-node in the value hierarchy indicates that several value objects are needed to produce the output value object, and so the corresponding value activity at least aggregates and possibly transforms value objects into the desired output object.

For instance, the AND construct labeled (1) in Figure 1 results in a value activity called Read article labeled (1) in Figure 2, to be performed by a *reader*. The graph shows that the *reader* needs to aggregate an article online and a telephone connection to be able to satisfy his need. The graph also shows that the *Amsterdam Times* produces the article itself by the value activity *Publishing*, and obtains *IP access* and *Hosting* from others.

The development of a value hierarchy and a related value exchange graph is a process of step-wise refinement. It is common to start with a more course-grained hierarchy, which results in a value exchange graph with a few actors and value activities. To find more fine-grained value hierarchies and value exchange graphs, we have proposed a deconstruction process which breaks down a hierarchy and graph into smaller parts [16].

## 4.3 Profitability sheets

To estimate the profitability of the value activities and exchanges, we have to estimate the number of actual value exchanges in a time period (e.g. a month). For each actor, the results are summarized on a **profitability sheet**, which shows our best estimate whether the e-service could be profitable (see Table 1 for an example).

To create a profitability sheet, we first express the occurrence of a consumer need by a black dot in the consumer activity (Figure 2). Each such occurrence will lead to an exchange across the consumer value interface, as indicated by the dependency path connecting the dot with the interface symbol.

A **dependency path**, depicted by gray lines, shows via which value interfaces an actor should exchange objects when triggered. Each dependency path connects via one or more *connection elements* two or more *dependency elements*, being an interface, an AND node (represented by a bar) or an OR-node (represented by a split). Dependency paths are used to assemble the data on profitability sheets by just counting how many objects are exchanged via value interfaces that are part of a path that is executed a given number of times.

The value exchange graph in Figure 2 shows that an activation of the consumer value interface leads to an exchange with the *Amsterdam Times* with the *Data Runner*, and this in turn leads to the activation of the value interface of these actors. The paths show the following information.

- The *Last Mile* exchanges a telephone connection for a telephone connection fee if and only if it exchanges an interconnection for an interconnection fee.
- The *Amsterdam Times* needs to obtain a termination fee, IP access and hosting service from the *Telecommunication consortium* to offer an article online. Also, it shows that a number ($N$) of *online* articles can be produced by using only *one* article written by a journalist. This shows that the marginal expense of generating a copy of a similar article is zero.

Dependency paths have been inspired by Buhr's use case maps [6] but differ from them because they do not carry any scenario information. They do not represent business processes but are instructions to assemble the profitability sheets; their main purpose is to facilitate counting.

To construct the profitability sheet, we start at the consumer need and follow the paths and value exchanges, until we have reached all end stimuli. Each time value objects are exchanged between actors, we update the profitability sheets for these actors.

Subsequently, the economic value of objects in terms of a monetary unit (e.g. Euros) is calculated. How to do so depends on the kind of actor. *End consumer* actors want to maximize their consumer value, defined by [34] as the receipts experienced by consuming the object divided by the sacrifices to obtain the object. Holbrook's consumer value framework can be used to elicit factors which determine the valuation by consumers [19]. Table 1 presents a profitability sheet for an *enterprise actor*. Such an actor wants to maximize its profit and net cash flow, or at least wants to play break even. According to enterprise investment theory [20], cash flows are considered only for an investment evaluation. Consequently, Table 1 shows objects representing goods, services or intangibles (in short, objects other than fees) in parentheses, because we do not consider these objects for profitability analysis. Then for fees, the sheet shows how these fees are calculated and an estimate on the profitability for actors (not given here due to confidential project data). The profitability sheet for each actor gives the information for business managers to decide whether it makes business sense to go ahead to the next stage of elaborating this e-service, which is the definition of business processes required to produce the desired value objects.

Table 1: Profitability sheet for the *Amsterdam Times*

| *Actor* | Amsterdam Times | |
|---|---|---|
| *Consumer need* | Read news article online | |
| | *Value Object In* | *Value Object Out* |
| Exchanges with readers: | (*termination*) | (*online article*) |
| Exchanges with telco: | *termination fee=telephone connection fee×revenue sharing factor* | (*termination*) |
| | *IP* (*access*) | *IP access fee=fee per second×duration* |
| | (*hosting*) | *hosting fee=fee per pageview×page views* |

To design value hierarchies as well as to calculate profitability sheets automatically, free tool support can be obtained from *http://www.e3value.com/*.

## 4.4 Lessons learned

The value exchange graph as depicted in Figure 2 is the final result of a modeling process. This process helped the stakeholders to understand which enterprises are required to realize the business idea, and their exchanges of value, to be able to offer an article online service. For instance, the TelCo used the model to explain the mechanism of call termination to the stakeholders. They did not succeed in explaining this complex construct otherwise. Additionally, developing profitability sheets provided actors insight whether the idea itself is economically sustainable. We learned that the actual numbers on profitability sheets are not so much of interest, but playing with these number and their assumptions, i.e. doing a sensitivity analysis, is interesting. By doing so, we discovered that the newspaper must be capable of attracting a substantial number of readers (order of magnitude hundreds of thousands) in order to be able to exploit the article online service economically sustainable. Additional lessons learned are explained in [13].

## 5 Process viewpoint

The value exchange viewpoint elaborates the e-service for the strategic manager. It does not represent processes but the *willingness* of an economically rational actor to create and exchange value. It represents a steady state that exists when as yet to be identified technology and people do their work. Taking the process viewpoint, we describe:

- which inter-organizational processes must exist to be able to satisfy the consumer need, and
- which tasks each actor must perform to realize these processes.

This elaborates the value viewpoint for the operational manager to execute the e-service in terms of processes. It shows which activities have to be performed by whom or what, and in which order, to produce which result. We discuss each of these hierarchies in turn.

### 5.1 Business process hierarchy

To find the required inter-organizational business process, we ask which processes must be performed to create the steady-state situation as represented by the value exchange graph.

Inter-organizational business processes are on-going activities that involve at least two actors. To identify the required processes, we use the following three types of processes, which are well known from economics [22] and business process / requirements engineering [23, 28] literature:

- *Primary processes,* which directly contribute to the satisfaction of consumer needs. This includes processes performed in the steady state, as well as ex-ante processes such as supplier selection and service subscription, and ex-post processes such as dispute resolution or service un-subscription.
- *Support processes,* which enable execution of primary processes and provide a suitable working environment.
- *Management process,* which organize, staff, direct, and monitor primary and support processes.

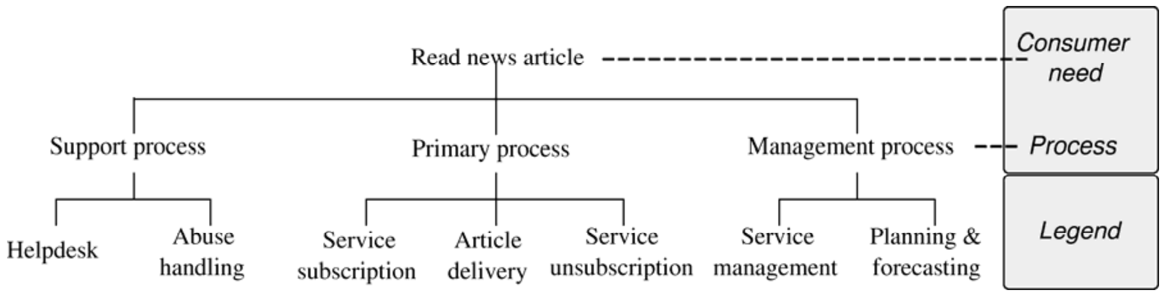

Figure 3: Process decomposition hierarchy for on-line news article delivery.

Figure 3 shows inter-organizational processes needed for satisfying the consumer need to read an online article. The leaves represent processes; the other nodes aggregate processes into compound processes that can be achieved by the lower-level processes. The processes have been elicited by using brainstorm sessions with the stakeholders involved.

Primary processes contribute directly to consumer satisfaction. The primary process consists of *article delivery*, which is the steady state of value activities and exchanges represented by the value exchange graph, and the *subscription* and *un-subscription* processes, which represent the entry and exit of a consumer to and from this steady state. Each of these processes involves several actors and is therefore cross-organizational. This requires careful integration of the information systems and business processes of the participating actors.

Support processes contribute indirectly to consumer need satisfaction. Here we identify two of such processes. The *help desk* process handles complaints and solves problems of end-customers. Because in this specific case service provisioning is partitioned over a number of actors, it is an inter-organizational process too. The same holds for *abuse management*: *Readers* for instance can use the offered IP-access for unintended and sometimes illegal purposes. This may result in blocking reader’s access to the service.

Cross-organizational management processes organize, staff, direct and monitor the other processes. An important management process in this case is *planning and forecasting*. To make the article online business case successful it is important that *precisely* sufficient resources (e.g. modem ports to dial in, web server capacity) are available to serve the *Readers*. A shortage in resources (e.g. modem ports) immediately results in a decrease in revenues because revenues are based on the total duration of telephone connections. On the other hand, unused resources, representing a substantial investment, result in a loss, especially for *Data Runner* and *Hoster* since these parties have invested in theses resources. Another process is *service management* itself. Service management consists of managing the quality of service (e.g. measuring the percentage of *Readers* who get a service denial, for instance caused by shortage in modem resources), developing the service from a content point of view (broader selection of articles, search for related articles, etc), and negotiating between parties about service delivery (e.g. between *Amsterdam Times* and the *Telecommunication consortium*).

## 5.2 Task hierarchy

Whereas business processes are on-going inter-organizational activities, tasks are terminating activities with input and output, assigned to a specific actor and therefore intra-organizational. For each process identified, we decompose the process into tasks which can be assigned to value activities of an actor. We try to reuse as much as possible of existing processes and information technology.

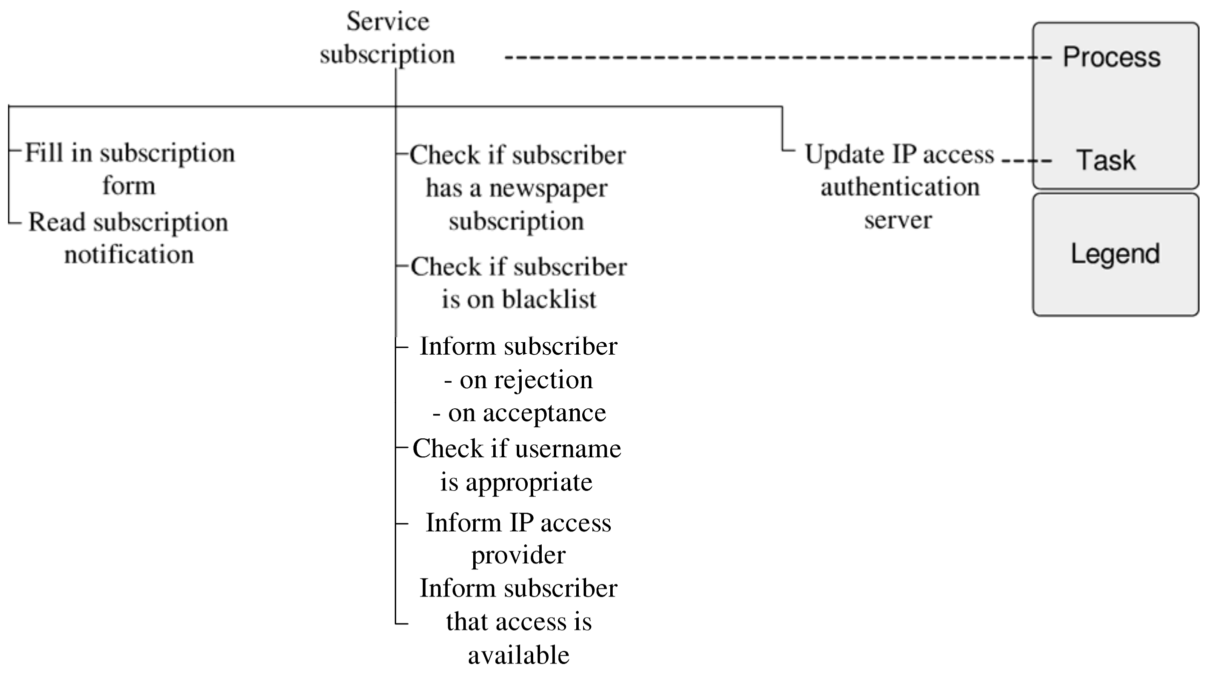
Service subscription
Fill in subscription form
Read subscription notification
Check if subscriber has a newspaper subscription
Check if subscriber is on blacklist
Inform subscriber - on rejection - on acceptance
Check if username is appropriate
Inform IP access provider
Inform subscriber that access is available
Update IP access authentication server
Process
Task
Legend

Figure 4: Tasks for the subscription process

Figure 4 shows the tasks for the subscription process as a task hierarchy. As with business processes, tasks are found by doing workshops with the participants involved in the e-service. The task hierarchy provides sufficient detail for a description of an e-service and does not require stakeholders more than two minutes to understand. A task hierarchy is used for the following purposes.

- To understand the tasks needed to fulfill consumer needs;
- To understand which tasks must be performed by an actor to realize a value activity;
- To identify whether tasks are new for an actor, or whether tasks performed anyway can be used or changed;
- To gain insight in operational expenses of processes and tasks, which can then be compared to the estimations as expressed by the profitability sheets;
- To identify information systems that can support the tasks.

To this end, we annotate the task hierarchy with a **task expenses estimation table** (see Table 2). To assess the profitability of an e-service, it is important to discover substantial labor resulting in high expenses which may inhibit the business idea. These expenses can by added to the profitability sheet (see Table 1) as value object *out* flows. As such, they will decrease the profitability number. Additionally, it is important to recognize whether tasks should be newly developed or existing tasks can be used for building the process.

Table 2: Task expenses estimation table.

| *Task name* | *Estimated labor* | *Existing/new* | *Implements value activity* |
|---|---|---|---|
| Fill in subscription form | 10 minutes / form | New | Read article |
| Check if a subscriber has a newspaper subscription | 0 (automated) /subscription | New | Provide article online |
| Update IP access authentication server | 0 (automated) | Existing | Provide IP access |
| … | … | … | … |

## 5.3 Inter-organizational business process

Tasks can be organized in an inter-organizational business process. To do so, we constructed a UML activity diagram of the service subscription process, with a swim lane for each actor (see Figure 5). Constructing the business process goes hand in hand with the creation of the task hierarchy.

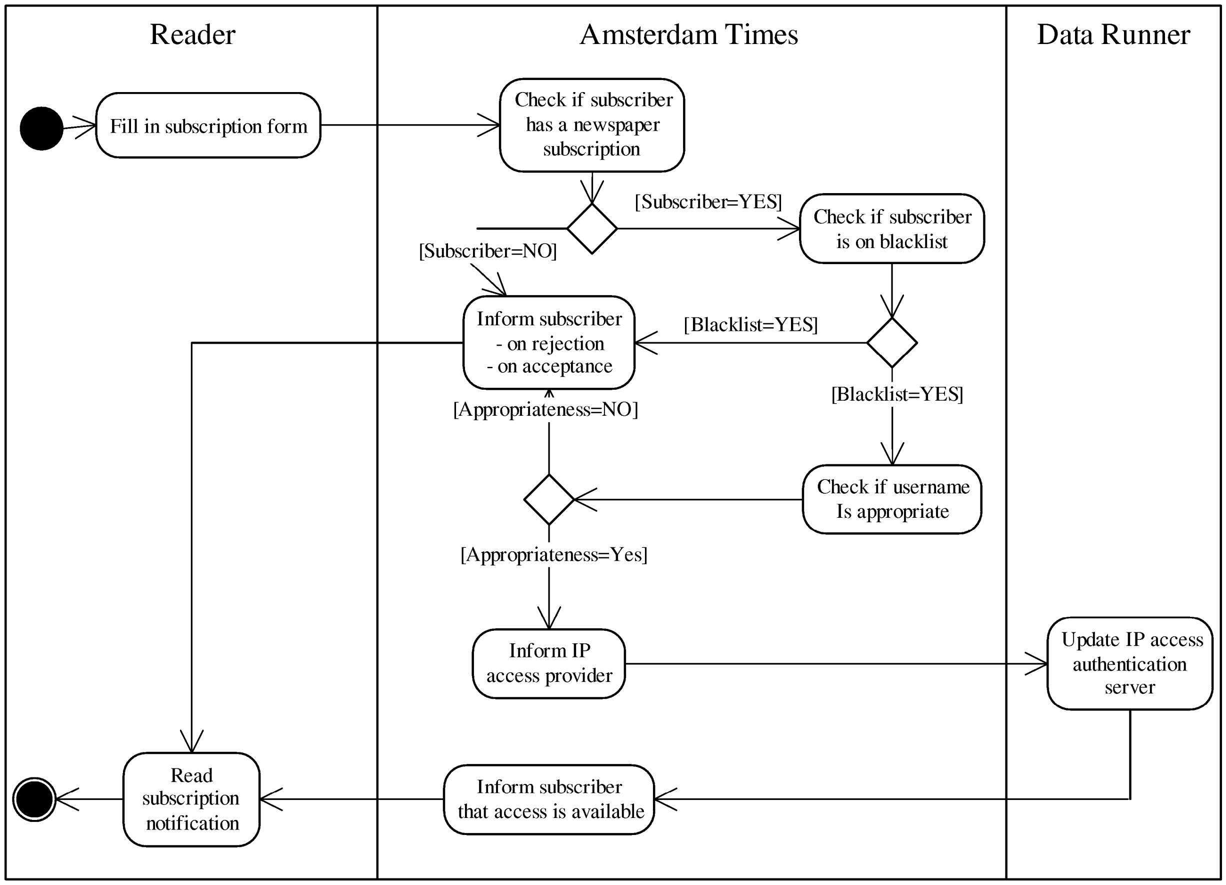
Reader
Amsterdam Times
Data Runner
Fill in subscription form
Check if subscriber has a newspaper subscription
[Subscriber=YES]
Check if subscriber is on blacklist
[Subscriber=NO]
Inform subscriber - on rejection - on acceptance
[Blacklist=YES]
[Blacklist=YES]
[Appropriateness=NO]
Check if username Is appropriate
[Appropriateness=Yes]
Inform IP access provider
Update IP access authentication server
Read subscription notification
Inform subscriber that access is available

Figure 5: UML activity diagram for the subscription process

Understanding the inter-organizational business process is important to arrive at a web-service implementation (see Section 6). In this specific case, the process can be largely supported by web-services and workflow technology, because most tasks (except filling in the subscription form and reading the notification) can be done automatically.

## 6 Information system viewpoint

The final step is to develop the information systems viewpoint based on the activity diagram presented in the previous section. By developing the information system viewpoint, a first view of the IT-consequences of the e-service at hand can be obtained. This is important to judge technical feasibility. Additionally, IT may require substantial investments in software and technical infrastructure. We now concentrate on a high-level software architecture in terms of workflows and web-services.

So, we specify the information system viewpoint using the web services standard specification formats WSDL and BPEL. We do so by first modeling the perspective using UML and then converting the resulting model automatically to WSDL [8] and BPEL [3] specifications.

We use a UML extension called the UML Profile for Automated Business Processes developed by IBM [2]. This profile defines UML extensions in the form of stereotypes that closely follow the concepts that comprise WSDL and BPEL. A UML model that uses this profile can systematically and automatically be converted into WSDL and BPEL documents. In fact, tool support for this has already been made available by IBM[4] .

The information system viewpoint consists of a static structure specification in the form of UML class diagrams and a dynamic structure specification in the form of UML activity diagrams. Figure 6 presents the static structure specification. This diagram defines a number of «role»s that are 'played' by business partners. A role can provide and/or use operations (the constituent parts of a web service). Services are defined by UML «interface»s, which are graphically denoted by classes with a circle icon in the upper right corner. A «realize» abstraction between a «role» and an interface denotes that the «role» provides this operation. A dependency between a «role» and an «interface» denotes that the «role» uses the operation.

The central class is *ATProcess*, stereotyped «process». Objects of this class represent instances of the subscription process presented in the previous section. The rest of the diagram is most easily explained as follows: «process» *ATProcess* plays a «role» called *AmsterdamTimes*, which «realize»s an «interface» *SubscriptionService*, which is used by «role» *Subscriber*. «process» *ATProcess* also plays a «role» called *AuthRequester*, which uses an «interface» *AuthentificationService*, which is «realize»d by «role» *AuthProvider*. To make the example more interesting, we assume that this is actually an asynchronous service implemented via a callback mechanism. Therefore, «role» *AuthRequester* also «realize»s an «interface» *AuthentificationCallback*, which is used by «role» *AuthProvider*.

The UML profile developed by IBM closely follows the structure of WSDL and BPEL. Consequently, the structure of Figure 6 is more or less a given; WSDL and BPEL do not leave design freedom. For reasons of space, operations and attributes have been kept as simple as possible. In reality, especially the data interchanged in the parameter lists needs to be designed more carefully. Also note that error handling is completely lacking. BPEL does provide facilities for this.

[4]See `http://www-106.ibm.com/developerworks/webservices/library/ws-uml2bpel/`.

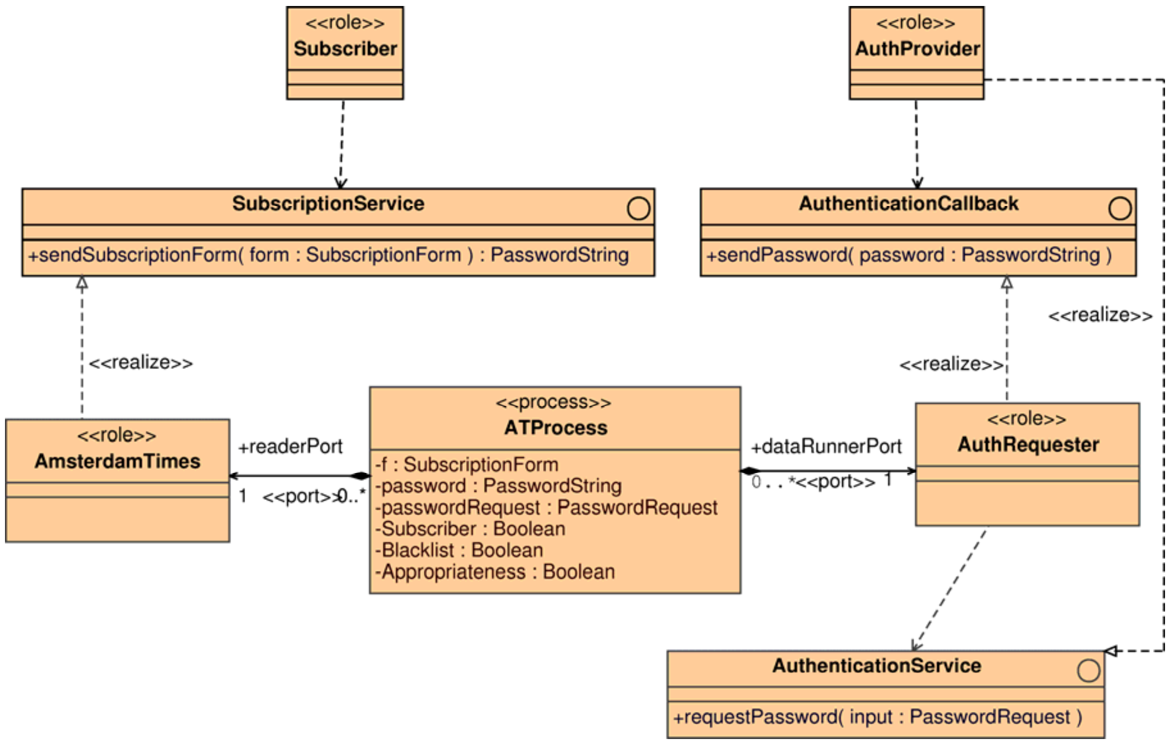

Figure 6: Static structure specification of the subscription process

Business partners actually exchange information via «port»s, which are communication endpoints. Associated with «process» *ATProcess* are two ports, one to contact readers and one to contact the *Data Runner*. Via its «port» *readerPort*, *ATProcess* plays its *AmsterdamTimes* «role», and therefore the operations specified by «interface» *SubscriptionService* are offered through this port. Via its «port» *dataRunnerPort*, *ATProcess*, *ATProcess* plays its *AuthRequester* «role».

Figure 7 shows the dynamic part of the information system viewpoint. The diagram in Figure 7 shows the process from the perspective of *ATProcess*. While in the activity diagram presented in the previous section (Figure 5) each swim lane corresponds with a business partner, in Figure 7 each swim lane corresponds with a port of *ATProcess* through which it connects with business partners. Consequently, the activities in the reader and *Data Runner* swim lanes of Figure 5 are not present in Figure 7. (This is a requirement of the IBM profile.) Therefore, the 'entry actions' specified for the activities in the *readerPort* and *dataRunnerPort* swim lanes are executed by *ATProcess*, just like the actions in swim lane 'self'. For the processes of the other business partners, separate activity diagrams have to be created. The relation between these activity diagrams is not covered by the UML profile developed by IBM. One would expect that operations offered in one diagram would be «invoke»ed in others.

The diagram can be explained as follows. Activity *Receive subscription form* is a BPEL receive activity, one of the types of basic activities provided by BPEL. This activity indicates that

*ATProcess* is waiting for a business partner to invoke the *sendSubscriptionForm* operation through the *readerPort* port. (This operation is defined in interface *SubscriptionService* specified in Figure 6). Parameter *f* is a local state variable of *ATProcess* that is used to store the incoming subscription form. The next activity, *Check subscriber status*, is a BPEL assign activity, which is similar to an assignment statement in a procedural programming language. We assume that a local procedure is available called *checkSubscriberStatus* to check whether we are dealing with a current subscriber. If this procedure were itself a web service, we would have to use a BPEL invoke activity instead of an assign activity. The argument given to *checkSubscriberStatus* is an XPath expression that is able to extract the relevant data from the form that has been received in the previous activity (this form is an XSD-defined XML data structure). After some more checks that are modeled by BPEL assign activities, *Data Runner* is contacted. As noted before, *Data Runner* uses asynchronous service invocation. Therefore, both a BPEL invoke activity and an explicit receive activity are needed. For synchronous invocation, a single BPEL invoke activity would have been sufficient.

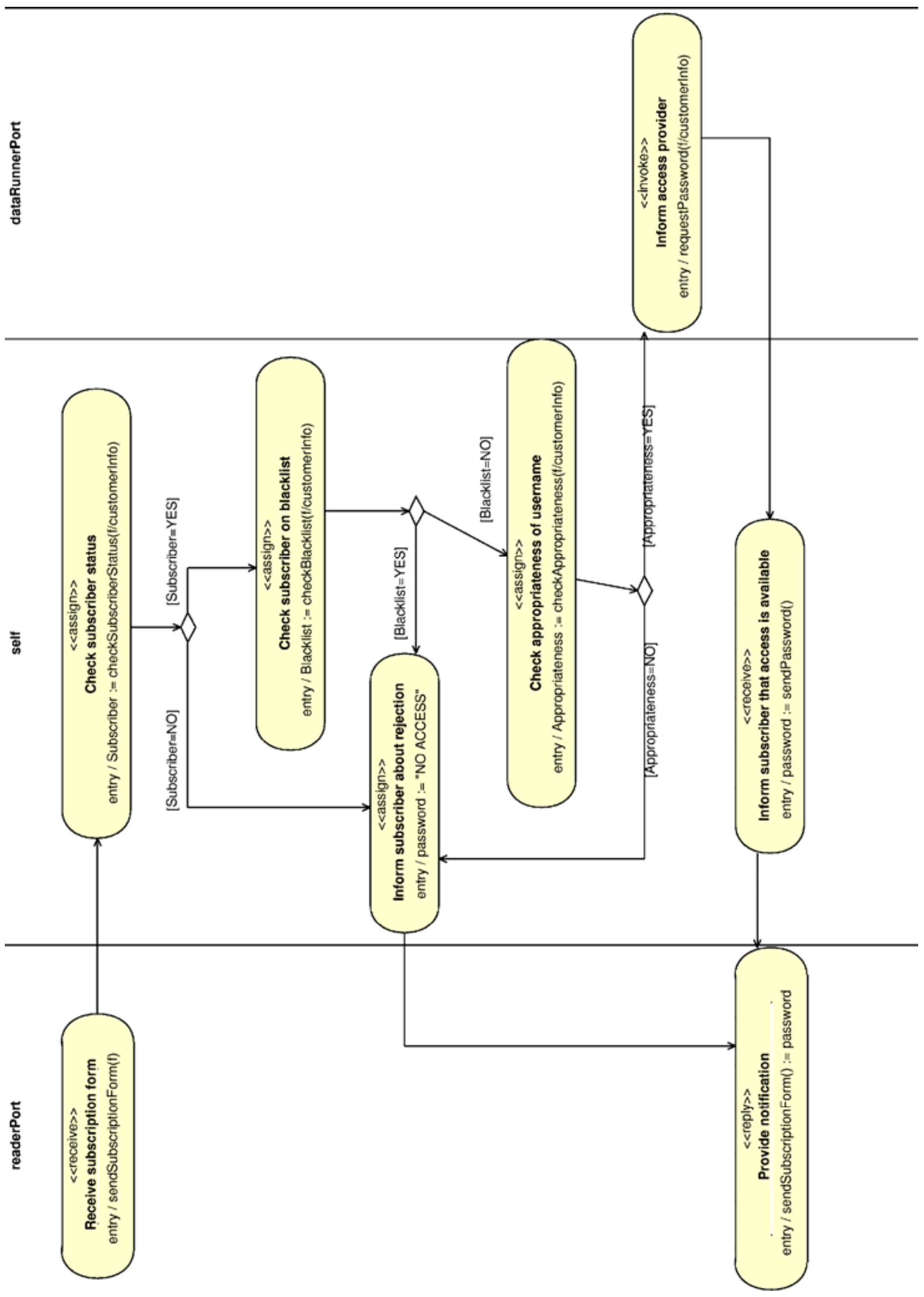
dataRunnerPort
self
readerPort
<<receive>>
Receive subscription form
entry / sendSubscriptionForm(f)
<<assign>>
Check subscriber status
entry / Subscriber := checkSubscriberStatus(f/customerInfo)
[Subscriber=YES]
[Subscriber=NO]
<<assign>>
Check subscriber on blacklist
entry / Blacklist := checkBlacklist(f/customerInfo)
[Blacklist=YES]
[Blacklist=NO]
<<assign>>
Inform subscriber about rejection
entry / password := "NO ACCESS"
<<assign>>
Check appropriateness of username
entry / Appropriateness := checkAppropriateness(f/customerInfo)
[Appropriateness=YES]
[Appropriateness=NO]
<<invoke>>
Inform access provider
entry / requestPassword(f/customerInfo)
<<receive>>
Inform subscriber that access is available
entry / password := sendPassword()
<<reply>>
Provide notification
entry / sendSubscriptionForm() := password

Figure 7: Dynamic specification

## 6.1 WSDL Specification

The static structure specification presented in Figure 6 can be converted systematically and automatically as described in the mapping developed by IBM [2]. The tool converts UML data types of operation parameters to XSD data types definitions, which are in turn used to define WSDL message types. These message types are used in the definition of portTypes, which are collections of operations provided via a port. The mapping developed by IBM converts UML interface to portTypes. For instance, interface *SubscriptionService* is converted to the following (fragment from) a WSDL document (we have deleted all XML name space references).

```
<wsdl:message name="SubscriptionForm">
     <wsdl:part name="customerInfo" type="someType"/>
</wsdl:message>
<wsdl:message name="PasswordString">
     <wsdl:part name="password" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="SubscriptionService">
     <wsdl:operation name="sendSubscriptionForm">
     <wsdl:input message="SubscriptionForm"/>
     <wsdl:output message="PasswordString"/>
     </wsdl:operation>
</wsdl:portType>
```

## 6.2 BPEL Specification

WSDL provides concepts for defining portTypes and for binding portTypes to actual ports and communication mechanisms (SOAP, HTTP), but not for sequences of invocations. The WSDL part of the mapping therefore only results in specifications of sets of operations that can be provided by someone to someone else. The actual structure of a business network as well as the order of messages exchanged can be specified using BPEL. In fact, the associations in Figure 6 are mapped to BPEL partner declarations. Class *ATProcess* is converted to a BPEL process declaration, and the associated activity diagram (Figure 7) is converted to a BPEL flow and its elements such as invocations and decision constructs (switches).

## 7 Summary

In this chapter, we have introduced an approach for specifying e-services. We explicitly distinguish e-services from web-services. e-Services are deeds, processes and performances which are of a more or less intangible nature that use Internet technology, and have a commercial, business orientation. Web-services and related technology are an implementation platform for such e-services.

Developing e-services is inherently a multi-disciplinary task. One way to deal with multiple disciplines is to use multiple viewpoints, according to requirements engineering. We distinguish three of such viewpoints.

The first viewpoint is the value viewpoint. This viewpoint represents the commercial perspective and explains why an e-service potentially can be successful. Industry has clearly shown that it is important to explore this viewpoint; many e-commerce implementations went bankrupt due to an insufficient understanding of revenue streams and value creation, distribution, and consumption [24]. As with the other viewpoints, requirements on the value viewpoint are expressed by a conceptualization, in this case a conceptualization of the actor network and the economic values that are created, distributed and exchanged.

The process viewpoint explains how the value viewpoint can be put into operation in terms of inter-organizational business processes. Consequently there is a strong relation between the value viewpoint and process viewpoint: We use process viewpoint to show *how* objects of economic value are created, distributed and consumed. Similarly to the value viewpoint, we distinguish more than one representation: we use an easy to understand process hierarchy, which can be decomposed in tasks. These tasks in turn can be organized in a UML-like inter-organizational specification that provides the starting point for a specification of the inter-organizational information system.

The information system perspective can be constructed from the business process viewpoint in a systematic way, and this is already supported by (prototype) tools. This is valuable, as these tools allow business network designers to abstract from the intricacies of XML and XSD. This is in the spirit of the OMG's Model Driven Architecture (MDA), which aims at fully separating a specification from its implementation platform. However, even at the level of UML diagrams, a tremendous amount of detail has to be provided to fully specify a business network. A number of these details amount to design decisions, e.g. whether a service will be designed for synchronous or asynchronous invocation. It remains a topic of further study to determine whether these design decisions are local to the information system perspective, or also have an impact on the other perspectives. The UML profile developed by IBM does not completely follow guidelines of the Model Driven Architecture, as at the level of the UML diagrams, details of implementation platform are still visible in the form of XPath queries.

Many papers on web service technology that are currently being published focus on automatic service discovery and composition. We have not used any of this. One could argue that our case study is too conservative and should have used e.g. web service discovery to dynamically find a suitable hosting service and web service composition to dynamically create the collaborations instead of statically at design time. However, it is currently much too early to employ these technologies in real-world case studies. More importantly, we think that it is essential to first fully understand the static case before moving on to dynamic composition of business collaborations.

Maintaining the consistency of descriptions across different viewpoints is a difficult problem [11]. For example, when is a cross-organizational coordination process a 'correct' implementation of a value exchange graph? How is the value proposition impacted when we add interaction mechanisms to the coordination process? How is it impacted if we change the information system requirements to allow the use of legacy systems? These questions are subject of current research [29].

**Acknowledgements**

This work is part of the Freeband A-MUSE project. Freeband (http://www.freeband.nl) is sponsored by the Dutch government under contract BSIK 03025.

## References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, 2004.

[2] J. Amsden, T. Gardner, C. Griffin, and S. Iyengar. Draft UML 1.4 profile for automated business processes with a mapping to BPEL 1.0. Technical report, IBM, 2003. `http://www-106.ibm.com/developerworks/rational/library/content/04April/3103/3103_UMLProfileForBusinessProcesses1.1.pdf`.

[3] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. K. a nd F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. T. ic, and S. Weerawanara. Business Process Execution Language for Web Services Version 1.1. Technical report, BEA Systems, IBM, Microsoft, SAP, Siebel, 5 May 2003.

[4] A. I. Antón. *Goal Identification and Refinement in the Specification of Information Systems*. PhD thesis, Georgia Institute of Technology, Raleigh, NC, 1997.

[5] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riener, S. Struble, P. Takasci-Nagy, I. Trickovic, and S. Zimek. Web Service Choreography Interface (WSCI) 1.0. Technical report, BEA Systems, Intalio, SAP, SUN Microsystems, 8 August 2002.

[6] R. J. A. Buhr. Use case maps as architectural entities for complex systems. *IEEE Transactions on Software Engineering*, 24(12):1131–1155, 1998.

[7] L. Cabrera, G. Copeland, W. Cox, M. Feingold, T. Freund, J. Johnson, C. Kaler, J. Klein, D. Langworthy, A. Nadalin, D. Orchard, I. Robinson, J. Shewchuk, and T. Storey. Web Services Coordinatioon (WS-Coordination). Technical report, BEA Systems, IBM, Microsoft, September 2003. `ftp://www6.software.ibm.com/software/developer/library/ws-coordination.pdf`.

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. Technical report, World Wide Web Consortium, 2001. `http://www.w3.org/TR/wsdl`.

[9] A. Dardenne, A. v. Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993.

[10] ebXML Business Process Specification Schema Version 1.01. `http://www.ebxml.org/specs/ebBPSS.pdf`, 11 May 2001.

[11] A. Finkelstein, D. gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handing in multi-perspectice specifications. *IEEE Transactions on Software Engineering*, 20(8):569–578, August 1994.

[12] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1):31–58, 1992.

[13] J. Gordijn and J. Akkermans. Value-based requirements engineering: Exploring innovative e-commerce ideas. *Requirements Engineering Journal*, 8(2):114–134, 2003.

[14] J. Gordijn, J. Akkermans, and J. v. Vliet. Business modelling is not process modelling. In *Conceptual Modeling for E-Business and the Web,* pages 40–51. Springer, 2000. LNCS 1921.

[15] J. Gordijn and J. M. Akkermans. Designing and evaluating e-Business models. *IEEE Intelligent Systems - Intelligent e-Business*, 16(4):11–17, 2001.

[16] J. Gordijn and J. M. Akkermans. Ontology-based operators for e-Business model de- and reconstruction. In Y. Gil, M. Musen, and J. Shavlik, editors, *Proceedings of the First International Conference on Knowledge Capture*, pages 60–67, New York, NY, oct 2001. ACM-Press. Also available from http://www.cs.vu.nl/~gordijn/.

[17] J. Gordijn and R. Wieringa. A value-oriented approach to e-business process design. In *LNCS 2681, Advanced Information Systems Engineering. Proceedings of the 15th International Conference, CAiSE 2003, Klagenfurt/Velden, Austria*, pages 390–403. Springer Verlag, 2003.

[18] C. Grönroos. *Service Management and Marketing: A Customer Relationship Management Approach, 2nd edition*. John Wiley & Sons, Chichester, UK, 2000.

[19] M. B. Holbrook. *Consumer Value: A Framework for Analysis and Research*. Routledge, New York, NY, 1999.

[20] C. T. Horngren and G. Foster. *Cost Accounting: A Managerial Emphasis, sixth edition*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[21] S. Janda, P. J. Trocchia, and K. P. Gwinner. Consumer perceptions of internet retail service quality. *International Journal of Service Industry Management,* 13(5):412–431, 2002.

[22] H. Mintzberg. *Structures in Fives: Designing Effective Organizations*. Prentice Hall, Englewood Cliffs, NJ, 1983.

[23] M. A. Ould. *Business Processes - Modelling and Analysis for Re-engineering and Improvement*. John Wiley & Sons, Chichester, UK, 1995.

[24] A. Shama. Dot-coms' coma. *The Journal of Systems and Software,* 56(1):101–104, 2001.

[25] D. Tapscott, D. Ticoll, and A. Lowy. *Digital Capital - Harnessing the Power of Business Webs*. Nicholas Brealy Publishing, London, UK, 2000.

[26] P. van Eck, J. Gordijn, and R. Wieringa. Value-based design of collaboration processes for e-commerce. In S.-T. Yuan and J. Liu, editors, *Proceedings 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE'04*, pages 349–358. IEEE Press, 2004. Taipei, Taiwan, March 28-31, 2004.

[27] A. C. R. van Riel, V. Liljander, and P. Jurriëns. Exploring consumer evaluations of e-services: a portal site. *International Journal of Service Industry Management*, 12(4):359–377, 2001.

[28] R. Wieringa. *Requirements Engineering: Frameworks for Understanding*. John Wiley & Sons, New York, NY, 1996.

[29] R. Wieringa and J. Gordijn. Value-oriented design of correct service coordination protocols. In *20th ACM Symposium on Applied Computing*, pages 1320–1327. ACM Press, March 13-17 2005.

[30] O. E. Williamson. *The Economic Institutions of Capitalism: Firms, Markets, Relational Contracting*. The Free Press, New York, NY, 1985.

[31] E. S. K. Yu and J. Mylopoulos. Why goal-oriented requirements engineering. In E. Dubois, A. L. Opdahl, and K. Pohl, editors, *Proceedings of the 4th International Workshop on Requirements Engineering: Foundation for Software Quality (RESFQ 1998)*, Namur, B, 1998. Presses Universitaires de Namur.

[32] V. Zeithaml and M. J. Bitner. *Services Marketing*. McGraw-Hill, New York, NY, 1996.

[33] V. Zeithaml, A. Parasuraman, and A. Malhotra. *A Conceptual Framework for Understanding E-Service Quality: Implications for Future Research and Managerial Practice, Report No. 00-115*. Marketing Science Institute, Cambridge, MA., 2000.

[34] V. A. Zeithaml. Consumer perceptions of price, quality, and value: A means-end model and synthesis of evidence. *Journal of Marketing*, 52:2–22, jul 1988.