Integral Design of E-commerce Systems: Aligning the Business with Software Architecture through Scenarios

Jaap Gordijn
Vrije Universiteit
Deloitte & Touche/Bakkenist
Management Consultants
e-mail: gordijn@cs.vu.nl

Hans de Bruin
Vrije Universiteit
e-mail: hansdb@cs.vu.nl

Hans Akkermans
Vrije Universiteit
AKMC Knowledge
hansakkermans@cs.vu.nl

Keywords
Business model, electronic commerce, value chain, value constellation, architecture, software architecture, Use Case Maps, business process, role-based modelling.

Abstract

Advancements in information and communication technology pave the way for a new class of business systems: e-commerce systems. These systems differ from traditional business systems in that they almost constitute the business rather than that they merely support the business of an organization. As a consequence, business and technology issues are intertwined in such a way that it is not sufficient anymore to consider them in isolation. For this reason, we argue that an integrated approach to e-commerce system development is required with which we can assess the impact of a business model on the information system and vice versa. In our approach, which we call *e³-VALUE*, an e-commerce system is considered from three architectural areas: business value, business process, and software architecture area. These three architectural areas cater for the needs of the various stakeholders involved in the development process at such an abstraction level that qualitative assessments can be made without getting buried by details. A scenario-based technique, represented by Use Case Maps (UCM), is used to relate the different architectural levels. The *e³-VALUE* approach is illuminated by means of an elaborated case study. Although it is too early to draw definite conclusions from this and other case studies that we have conducted, we did learn some important lessons. The first important lesson is that the case studies suggest that e-commerce systems can indeed be assessed qualitatively at a high level of abstraction as provided by the three architectural areas. The second important lesson is that an integrated approach can reveal organisational consequences that are not obvious from a business model alone.

1 Introduction

Today, we are facing a new and challenging class of information systems: e-commerce information systems. An important characteristic of these systems is that they are an integral part of the way of doing business and should reflect the business well. E-commerce systems are not a derivative from business processes or a-like, but mostly reflect new ways of doing business, enabled by new technological possibilities. Development of an e-commerce information system is therefore a continuous process of aligning technical possibilities and business opportunities. Therefore, we argue that the development of e-commerce business opportunities and their supporting information systems should be *integrated* processes, rather than separately or sequentially performed processes. Moreover, this design should be initially at a global level to allow for early assessment of design alternatives and communication with all stakeholders.

We address the development of e-commerce systems by a structured approach, called *e³-VALUE*, which offers a way of developing new ways of doing business and supporting information systems in an integrated way. Our approach focuses on a description of three architectural areas representing the interest of various stakeholders: the business value area, the business process area and the business software architecture area. Scenarios, represented by Use Case Maps (UCM) [Buhr (1998)], are used to integrate these architectural areas.

In this paper, we show our *e³-VALUE* approach by a case study. By developing our three architectural areas and integrating scenarios we are able to identify design trade-offs between several architectures for an e-commerce system in an early stage. It is interesting to notice, while we were aiming at

assessing the economical and technical feasibility of the e-commerce system, the stakeholders used the architectures to consider their position in the

This paper is structured as follows. In Sec. 2, we discuss the *e³-VALUE* framework for e-commerce applications. Sec. 3 presents the various identified architectural areas and the use of scenarios in more detail by working out an e-commerce case study. In Sec. 4 we present conclusions and lessons learned.

2 The *e³-VALUE* framework for e-commerce applications

2.1 Developing e-commerce systems

A commonly used definition for a system is that a system is any actual or possible part of reality that, if it exists, can be observed [Wieringa (1996)]. An e-commerce system consists of two subsystems: the e-business system and the e-information system. The e-business system, is comprised of, amongst others, contracts, legal rules, and organisational structures. These matters are not implemented in a software or hardware. This in contrast with the e-information system, which consists of the interconnected hard- and software components.
E-commerce information systems differ from other business information systems in the way they relate to the business process of a company [Rayport and Sviokla (1995)]. E-commerce information systems perform most business processes *themselves*, especially when goods and services are intangibles. However, most information systems for traditional ways of doing business only *support* the business process of company. Because e-commerce information systems are an important *part* of the way of doing business, the development of e-commerce information systems requires a tight integration of the e-business system and the e-commerce system

To address the development of e-commerce systems, we propose a number of architectural areas which are of relevance for e-commerce systems. To ensure that these areas are integrated we use scenarios as a glue to interconnect these areas. The areas are not developed in a sequential way but require mutual adjustment and are developed in iterations and sometimes in parallel [Marco Iansiti and MacCormack (1997), Gordijn and van Vliet (1999)].

Architectural areas are developed by an architect. He is responsible for the architectural design of a system, which include (1) designing artefacts, in particular the design of a way of doing business, business processes and the information systems, (2) communicating an architectural design to the stakeholders, and (3) supervising the development process.

Using an architectural approach it possible (1) to gain insight at an early stage in the qualities of an existing system or a system to be, (2) to use an architectural design as a guide for planning and controlling the subsequent development stages, and (3) to inform the stakeholders about what is built, the way it is built, and the implications on the current situation.

2.2 Architectural areas for e-commerce systems

In our approach, *e³-VALUE*, we distinguish the e-business value area, the e-business process area and the e-business software architecture area, respectively. The business value area shows the way of doing business and captures business decisions. It consists of actors, activities performed by these and the exchange of objects of value. Moreover, it imposes requirements to the business process and the software architecture. The business process area shows which activities are performed by actors and which information is exchanged between actors as well as usage of resources. The software architecture, finally, should prescribe requirements for an e-information system implementing important parts of such a business process.

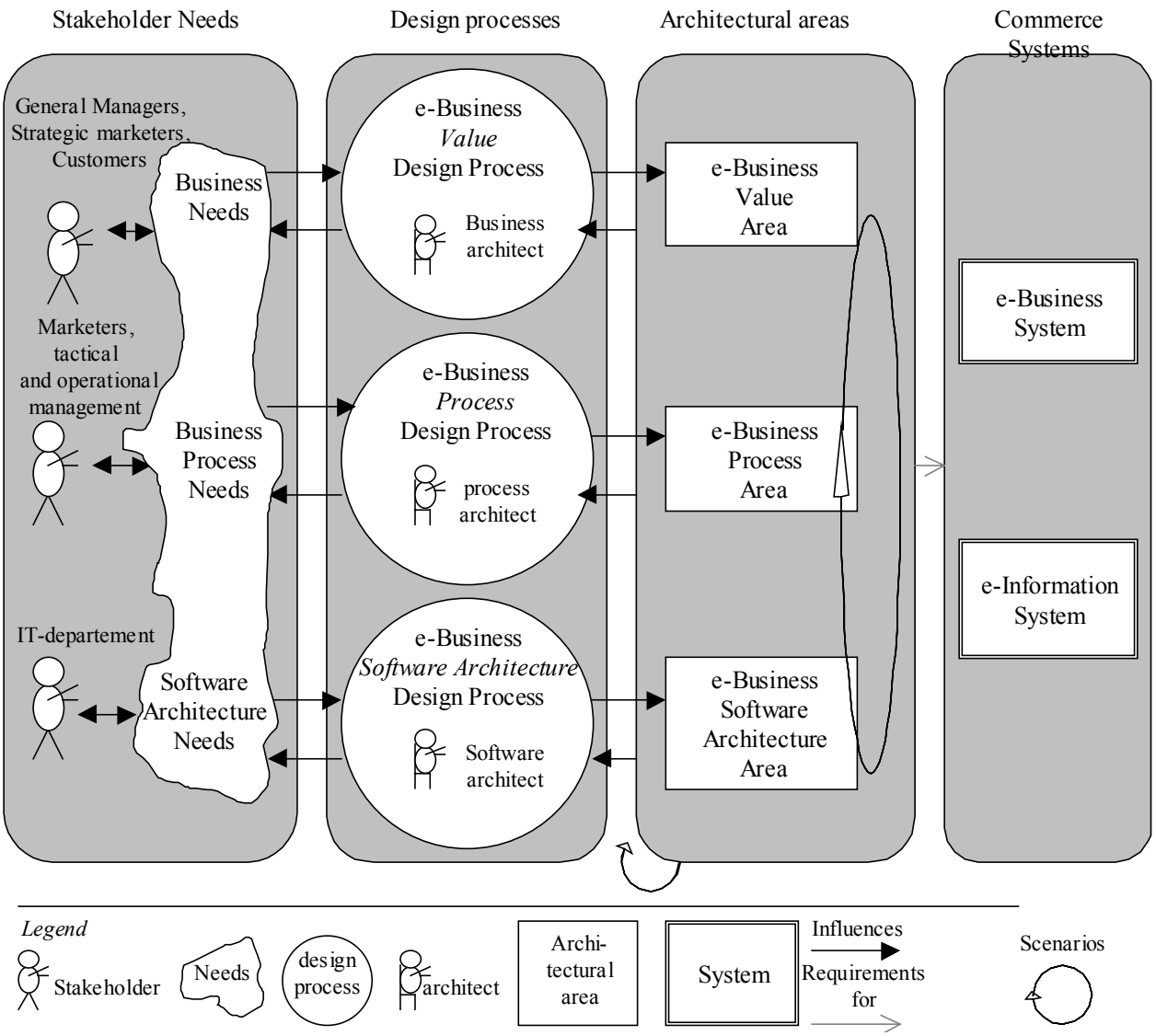

Figure 1 E-business design processes produce system requirements in different areas, based on various stakeholder needs.

2.2.1 E-Business Value Area

The top-level area of interest for our electronic commerce framework concerns the electronic-commerce business value area. The value area describes the way of doing business between actors, and so sparks off requirements for the business process and supporting software architecture. It identities actors being companies, consortia or persons, activities performed by actors and exchange of objects of value between activities. Stakeholders are general managers of companies participating in the execution of the business model, marketers and customers. Business developers are the primary designers of this area.

For the design of such value models is hardly any scientific consensus or sound method available. In this paper, we present a way to represent such a business value area by identifying core concepts present in such an area. A key idea of our approach is that structured *value* analysis is a crucial activity in business model design. In modelling value, we suggest that a good starting point is found in business administration literature, in particular work on value creation in micro-economic pricing theory, the value-chain concept [Porter and Millar (1985)] or, better, the value-constellation notion [Normann (1994)]. However, a more formal representation of business value areas is needed, and we propose such a representation in Sec. 3.

2.2.2 E-Business Process Area

The e-business process area, the middle level in Figure 1, shows on how activities should be performed and by whom. Initially, it shows roles to be performed by various actors. These roles can be detailed in to activities to show the process flow. On the level of roles, we show messages exchanged between roles performed by actors. Stakeholders are managers on tactical and operational level because they are responsible for carrying out most processes, and marketers regarding detailed buy and sell flows. Business process engineers are the most important designers. To represent a business process view a number of techniques are suitable, for instance UML collaboration diagrams, supplemented with swimming lanes to represent actors[Fowler and Scott (1997)], or role-based process-modelling techniques [Ould (1995)].

2.2.3 E-Software Architecture Area

The software architecture area shows how the business model captured in the business value and process architecture can actually be realized in a software system. A software architecture is comprised of components that models an e-commerce system at such a level of abstraction that we can evaluate the consequences of design decisions in terms of quality attributes that are of prime importance in this particular application area. The stakeholders that are involved in the design of a software architecture are software developers which include software architectects, software designers and implementers, but also man-machine interface experts. Business process engineers can be involved to ensure that business processes are properly implemented in the software architecture.

For our purposes, a software architecture will be primarily developed to assess the feasibility of a business model. That is, to check whether a business model can be realized in the first place and to check whether the business goals can be reached. Once feasibility has been demonstrated, both business and software architecture wise, the software architecture can be elaborated further into an implementation. The quality attributes that are important include performance, availability, maintainability, and security. In order to assess an e-commerce application with respect to these attributes we must develop a view or perhaps multiple views on the software architecture to identify the relevant components and structures. For e-commerce applications, the components that are of interest typically consist of databases, WEB-servers, and networks that are structured following proven architectural styles (also referred to as patterns) such as the 3-tier architectural style.

2.3 Scenarios: Use Case Maps

The three architectural areas need to provide an integrated, consistent, view on the commerce system rather than views on their own. In our approach we represent the relations between these areas through scenarios. For each architectural area, we show the same set of scenarios. Scenarios are represented by Use Case Maps (UCM) [Buhr (1998)]. A UCM is a visual notation to be used by humans to understand the behavior of a system at a high level of abstraction. It is a scenario-based approach showing cause-effects by traveling over paths through a system. The semantics of UCMs are not defined clearly and have to be tailored to the architectural area they are being applied to. If applied in combination with business models, UCMs model causal exchanges of values between actors. For a particular scenario, a UCM shows which values are exchanged between which actors. In the case of business processes, UCMs show the causal exchange of messages between activities. In the realm of software systems, they bridges the gap between global requirement analysis models (e.g., use cases and class diagrams) and very detailed design models (e.g., interaction diagrams such as collaboration and message sequence diagrams) by showing coarse-grained behavior of interacting software components. An important feature of a UCM is that it can show multiple scenarios in one diagram and the interactions amongst them. This makes them well suited to depict architectural designs focusing on the behavioral aspects of a system.

The basic UCM notation is very simple. It is comprised of three basic elements: responsibilities, paths and components. The term component should be interpreted in the broadest sense. It may be a software component, but it can also represent a human actor or a hardware system. A simple UCM exemplifying the basic elements is shown in Figure 2. A path is executed as a result of the receipt of an external stimulus. Imagine that an execution pointer is now placed on the start position. Next, the pointer is moved along the path thereby entering and leaving components, and touching responsibility points. A responsibility point represents a place where the state of a system is affected or interrogated. The effect of touching a responsibility point is not defined since the concept of state is not part of UCM. Typically, the effects are described in natural language. Finally, the end position is reached and the

pointer is removed from the diagram. A UCM is concurrency neutral, that is, a UCM does not prescribe the number of threads associated with a path. By the same token, nothing is said about the transfer of control or data when a pointer leaves one component and (re-)enters another one. The only thing that is guaranteed is the causal ordering of executing responsibility points along a path. However, this is not necessarily a temporal ordering, the execution of a responsibility point may overlap with the execution of subsequent responsibility points.

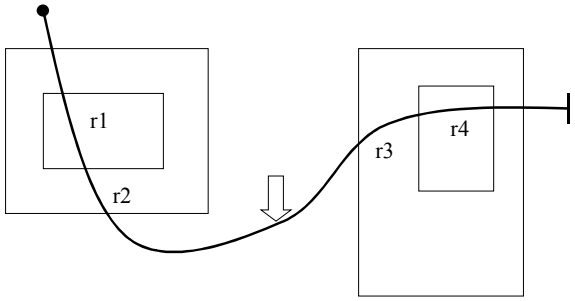

Figure 2 A basic UCM

A more realistic example is shown in Figure 3 depicting a distributed client-server system. Because the client communicates with the server over a network that can fail occasionally, a proxy server is included to provide transparent access to the real server. The proxy server is modeled as a stub for which two implementations are given: a transparent proxy server which passes the requests to and the replies from the server unaltered thereby denying the possibility of network failures, and a proxy server with a timeout facility with which failures are detected. The notation used in the figure is supposed to be self-explanatory.

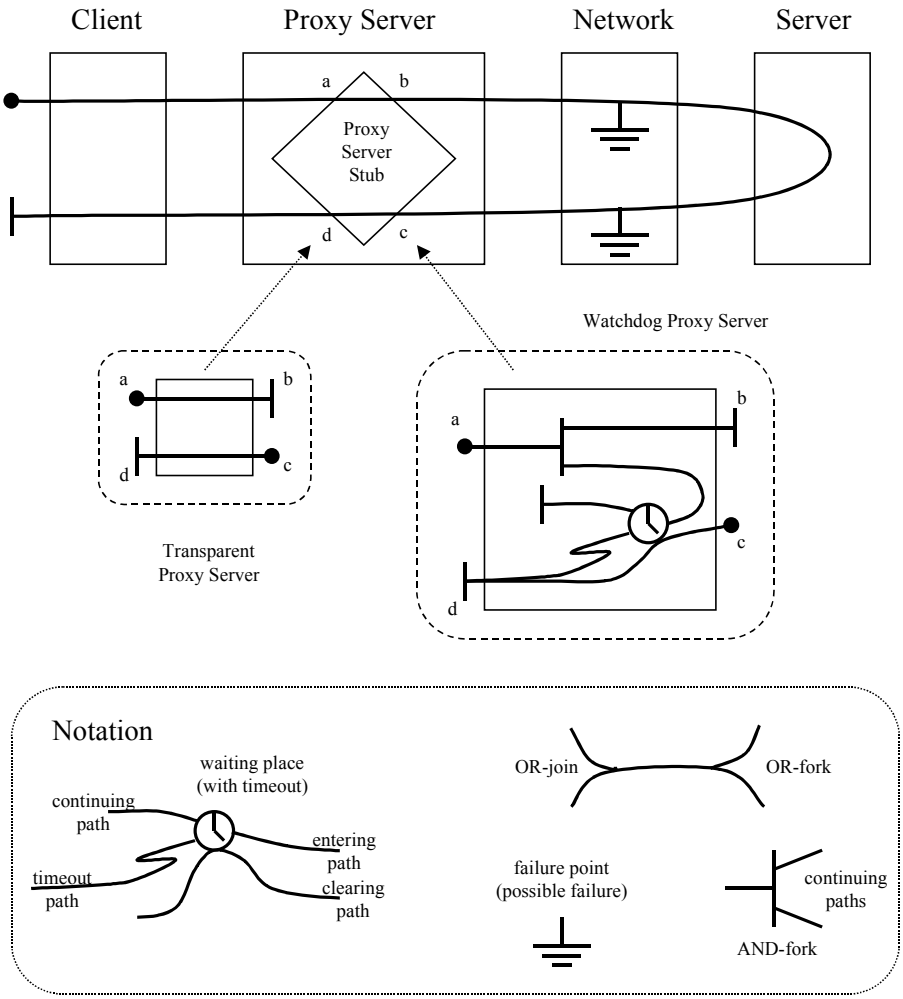

Figure 3 A UCM depicting a client-server system

It is interesting to see that many things are unspecified in UCMs, but the intended meaning is suggested strongly. For instance, distribution aspects (e.g., connection mechanism and the amount of concurrency in a component) are not dealt with. However, the client, the server and the proxy server are distinct components that are connected by a network, which is also modeled as a component. By using these names, it is natural to assume that the components are distributed over a number of computer systems. But again, it is not specified, it is all in the eye of the beholder.

The UCM notation is quite rich and supports also the creation and the deployment of dynamically created components (see Figure 4). This is referred to as structural dynamics. Components can be stored in pools for later use, but only if they are moved in slots. Once they have performed their duty, they can be moved out of their slots to be destroyed or to be restored in a pool. An interesting application area of pools is to model a limited set of resources. A scenario in progress can only remove a resource form a pool if one or more resources are in it, otherwise the scenario will wait until a resource has been restored into the pool by another scenario.

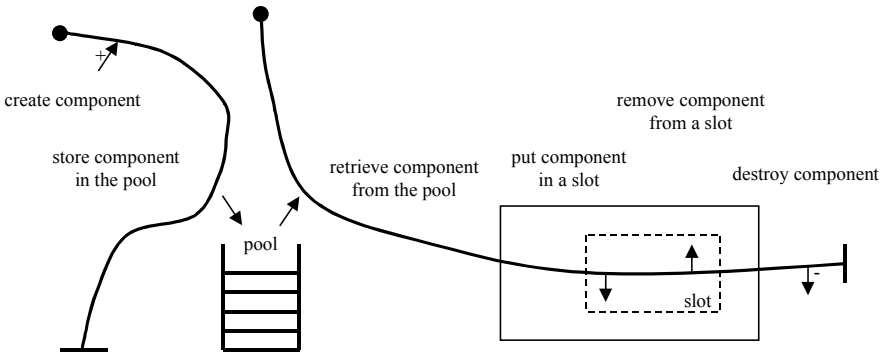

Figure 4 The usage of pools in UCM

3 The *e³-VALUE* framework illustrated by an e-commerce case study

3.1 Case outline

The Ad Association is a company which co-ordinates more than 150 local free ad papers called FAPs. FAPs produce (non-electronic) papers with ads. These FAPs are located world-wide. They are independent, often privately owned organisations. A FAP serves a geographical region, for instance a large city or a county. The handling of ads is as follows. A customer submits an ad to a FAP. The FAP checks the ad (e.g. for absence of dirty language and for style) and places the ad in its next issue. It is possible to place an international ad. In this case, the FAP to which the ad was submitted distributes the ad to other FAPs (serving different geographical regions). These other papers publish the ad as soon as possible. Placement of an ad is for free. However, a person who wants to read an ad has to pay a FAP by buying its paper. The exchange of international ads between FAPs is nearly for free. FAPs are only charged for the use of a common infrastructure which is offered by the Ad Association. The Ad Association carefully analysed the international ads. They concluded that international ads are mostly contact ads. In a contact ad, someone is searching for another person. The Ad Association is considering an Internet-based service for international contact ads. A number of business objectives are important. First, FAPs wants to protect the current market share of world-wide (paper-based) contact ads. FAPs are afraid of new parties entering the arena of international contact ads. They are especially afraid of competitors which are capable of setting up a world-wide Internet-based contact service. Ad papers want to exploit their local trusted brand names now to establish a trustworthy internet based contact ad service before someone else does. Thus, the development of a contact service has rather defensive objectives. Second, FAPs want to enlarge the market share of ads by exploiting yet another communication channel. Third, FAP wants to attract customers to their existing ad papers by offering a full service spectrum, amongst others, placement of an ad on the Internet.

The next sections show for this case the architectural areas in more detail. We work out one business value model and business process model. Other options are considered in [Gordijn *et al.* (1999)]. Subsequently, we show three software architectures that all realize the given business value and process model. The goal here is to show feasibility, from an economical as well as a technical perspective.

3.2 e-business value area

We propose that the central concept in any representation of the way of doing business is that of a value activity. A value activity is performed by actors and aims at producing material or immaterial objects that are of value to others. This notion of value activity is recognised in, e.g., [Porter and Millar (1985), Normann (1994), Kaplan and Norton (1996)]. Value activities as specified in [Porter and Millar (1985)] can be connected to form a value chain. At the macro-level, we can use these concepts to specify a business value model. However, from micro-economics theory, interesting concepts can be borrowed in the field of pricing theory [Choi *et al.* (1997), Hagel III and Armstrong (1997), Shapiro and Varian (1999)]. In particular, these authors consider extracting the maximum price a customer is willing to pay as one of the challenges of electronic commerce applications. They propose to do this by offering each customer a specific tailored version of a product to each customer. We use these macro- and micro-concepts, as well as our consulting experience in designing electronic commerce applications, to derive a small set of below discussed core concepts needed to represent a semi-formal business value model.

Actor. An actor is an independent entity such as a company or a person. Actors perform one or more value activities. In our case, the following actors participate: (1) contact searcher, (2) FAPs and (3), Ad Association.

Value activity. A value activity represents a process which adds value. Actors perform these value activities. An actor can perform multiple value activities, but a particular value activity is performed by one actor only. When developing business value models, we are primarily interested in finding chunks of activities that add value and in studying the various possible assignments of these activities to different actors. These reflect important business decisions. Value activities for a specific case are specialisations of the value-activity concept. The granularity of defining value activities should be such that they can be performed technologically and economically independently from other value activities [Porter and Millar (1985)], *and* that they cannot be further decomposed into smaller activities that can be assigned to different actors. Instances of specialised value activities are mapped onto the set of actors. Constructing these value activities and mapping of its occurrences onto actors is an important part of the electronic commerce design problem.

In the Ad Association case, we distinguish the following value activities. The (1) *place ad* and (2*) read ad* value activities represent activities typically performed by contact searchers. Value activity (3), *ad intake*, executes placement of an international ad. Value activity (4), *check ad*, checks an ad for correct use of language. Value activity (5), *publish ad*, offers a reading service of ads to contact searchers. Value activity (6), *redistribute ad*, receives an ad from a FAP and redistributes this ad to other FAPs.

Many assignments of value activities to actors are possible. We choose for an assignment that closely resembles the current practice of the Ad Association (Table 1).

Value activity	*Actor*
Place ad	Contact searchers
Read ad	Contact searchers
Ad intake	FAPs
Check ad	FAPs
Publish ad	FAPs
Redistribute ad	Ad Association

Table 1 Assignment of value activities to actors

Value object and value object type. A value object is what is produced or consumed by a value activity. Value objects are the things that are exchanged between value activities. A value object type denotes the type of asset which is created or used by a value activity. A value object type refers to a type of (digital) good, a service type, or type of money [Choi *et al.* (1997)], for instance token-based or notational money [Camp (1996)]. A value object has one value object type.

Value port. We further need a formal way to indicate how value activities can be connected to each other in a component-based and (re)configurable manner. Here, we introduce the concept of ports, a notion known from general and technical systems theory (as a helpful analogy, think of a wall outlet for electricity; it has two ports). A value port, then, denotes a connection point of a value activity that defines how it may be connected to the external world of other value activities. On a value port, value objects are exchanged. A value port has exactly one value object type. Value objects can flow into a value activity or away from a value activity via a port. This direction is modelled as a property of the value port. A value port can have various properties such as a price or price range for the value object. Note that a property such as a price is seen as a property of the port and not of the value object, because other actors may offer the same value object for a different price.

Value interface. Value ports are grouped into value interfaces. A value interface represents a commerce service offered to or requested from a value activity. It consists of at least one value port. A value interface having only one value port can be used to model a value activity which produces value objects for free. In other cases, we have two ports; one value port for the outgoing good or service to be sold and one value port for the incoming payment (not necessarily money, for instance in some cases one can pay with privacy information). Finally, one can think of more than two ports in an interface, to model the business concept of bundling [Choi *et al.* (1997), Shapiro and Varian (1999)]. Bundling refers to the situation that a customer buys a number of products or services (the bundle) as a whole

and pays for this bundle as a whole. The opposite situation also can occur: in these case multiple payment instruments are used. A value activity may have multiple value interfaces. Two motivations for having multiple interfaces exist. Firstly, a value activity typically requests (buys) value objects from actors and uses these objects to create and sell other value objects, mostly to other actors. The value activity has in this case two faces to its environment: one as a buyer and one as a seller. For each, a value interface is available defining the commerce service requested or offered. Secondly, multiple versions of equally typed value objects can be sold against different terms and in different bundles to address price and product differentiation [Shapiro and Varian (1999), Hagel III and Armstrong (1997), Choi *et al.* (1997)]. Versioning, bundling and different terms are ways to implement value-based pricing. With value-based pricing, a seller tries to extract as much value from the buyer as possible, by making an offer that is targeted to the specific customer. We employ different value interfaces to model the situation that a value object is offered in different versions, bundles and with different terms since they are different commerce services. A value interface also prescribes the value ports of value activities which can be interconnected. A connection between two ports of different value interfaces can only be made if these value interfaces match. Interfaces match if for each value in-port in an interface, a corresponding value out-port in the other value interface can be found and vice-versa, and, for each set of connected value ports, the value ports have the same type. On a value interface a number of rules and constraints can be defined. For example, consider a time-ordering rule stating that a customer has to pay on a value port first and subsequently receives the good (pre-payment) or vice versa (post-payment) via another value port.

For the Ad Association case, value interfaces, value ports and value object types are concisely presented in Table 2.

	Value activities	*Value interfaces consisting of value ports with value object type*
1	Place ad	In port: Placed ad of type ad Out port: Submitted ad of type ad
2	Read ad	In port: Read ad of type ad Out port: Payment for reading ad of type money
3	Ad intake	In port: Submitted ad of type ad Out port: Placed ad of type ad
		In port: Checked ad of type ad Out port: Payment for checking ad of type money
		In port: Payment for sending ad of type money Out port: Sent ad of type ad
4	Check ad	In port: Payment for checking ad of type money Out port: Checked ad of type ad
5	Publish ad	In port: Received ad of type ad Out port: Payment for receiving ad of type money
		In port: Payment for reading ad of type money Out port: Read ad of type ad
6	Redistribute ad	In port: Received ad of type ad Out port: Payment for received ad of type money
		In port: Payment for sending ad of type money Out port: Sent ad of type ad

Table 2 Specific value activities, value interfaces, value ports and value object types for the FAP centred business value model.

We have developed a technique to graphically represent the business value model. The Ad Assocation business value model is illustrated in Figure 5. Note that this figure also shows scenarios as illustrated below.

Value scenario. A value scenario shows the causal sequence of value exchanges for typical real-life cases. A scenario consists of a path to come from a start point to an end point. It is possible to have more than one path from a start point to an end point, resulting in more than one scenario. Such a situation exists if the path has an OR-fork. A scenario can split in multiple sub-scenarios using an AND fork. In such a case, a scenario has multiple end-points.

For the Ad Association case, we distinguish the following main value-scenarios: (1) place ad, (2) distribute ad, and (3) read ad.

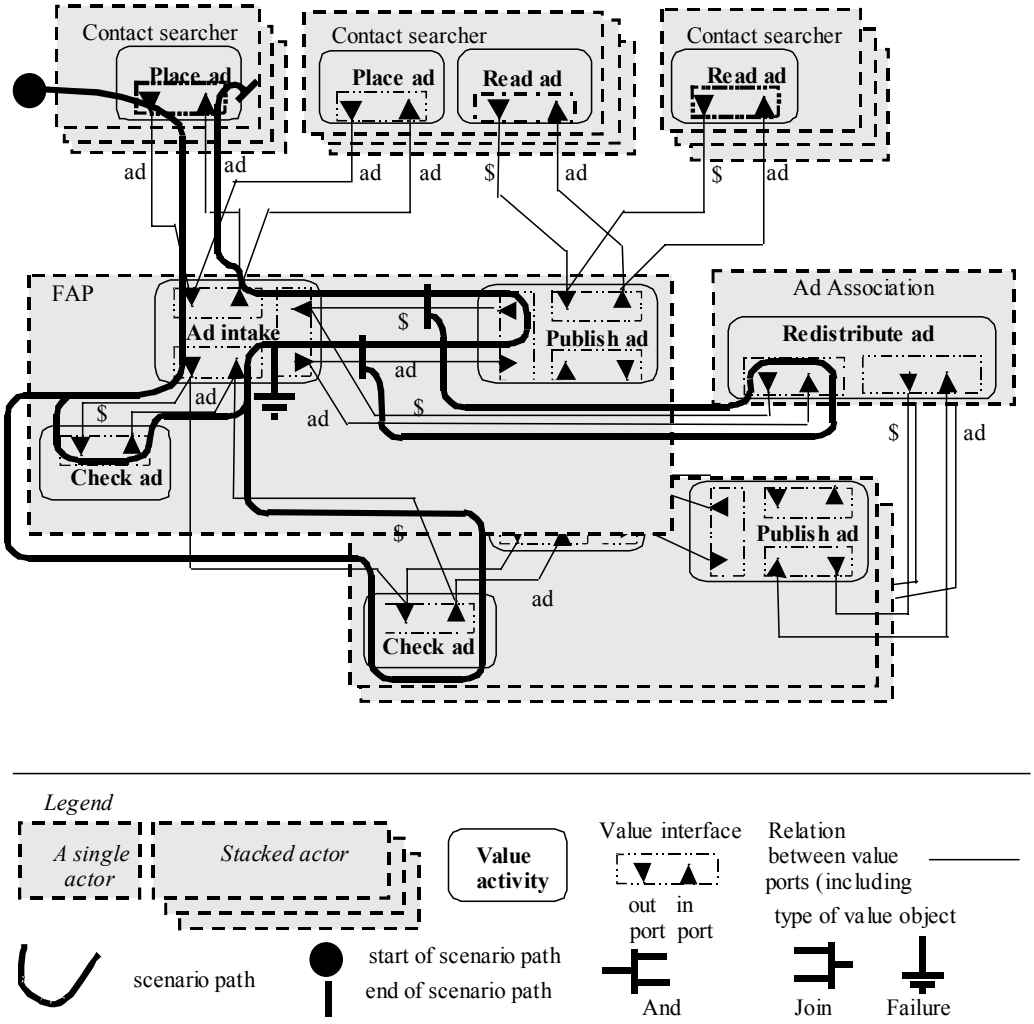

Figure 5 Bussiness value area of the Ad Association – place ad scenario

The value scenario (1), *placement of ad*, is presented in Figure 5 and starts at the upperleft corner. The contact searcher places an ad which is the first value exchange between contact searcher and FAP. Next, the scenario chooses one of two paths to continue for checking an ad for correct use of language. If the first route is chosen, the ad is checked by the same FAP who performed the *ad intake* value activity; following the other route, another FAP checks the ad. In both cases, two value exchanges occur: the FAP who performs the ad intake, pays the FAP who checks the ad. Now, the ad has been approved and can be published. The scenario continues with an AND fork, and therefore splits in two sub-scenarios. One sub-scenario models that the ad is published by the FAP and the other sub-scenarios shows that the ad is redistributed to other FAPs. Both sub-scenarios result in the same type of value exchanges: the FAP who performs the take in, receives money for each ad he delivers. Hereafter, the last value exchange occurs: the FAP who takes in the ad, delivers value the contact searcher because his ad is placed successfully.

In some cases, a submitted ad an be rejected for reasons of bad language. This is modeled by the earth symbol (taken from electrical wiring diagrams) which in UCM represents a failure point. In such a case, the scenario terminates. Note that a submitted ad is exchanged between contact searcher and a FAP but that the ad is of no value to the FAP. However, this only can be determined after checking the ad. To put it differently, this is a loss situation for both the contact searcher (his ad is not placed) and the FAP (an ad has been checked that will not generate money).

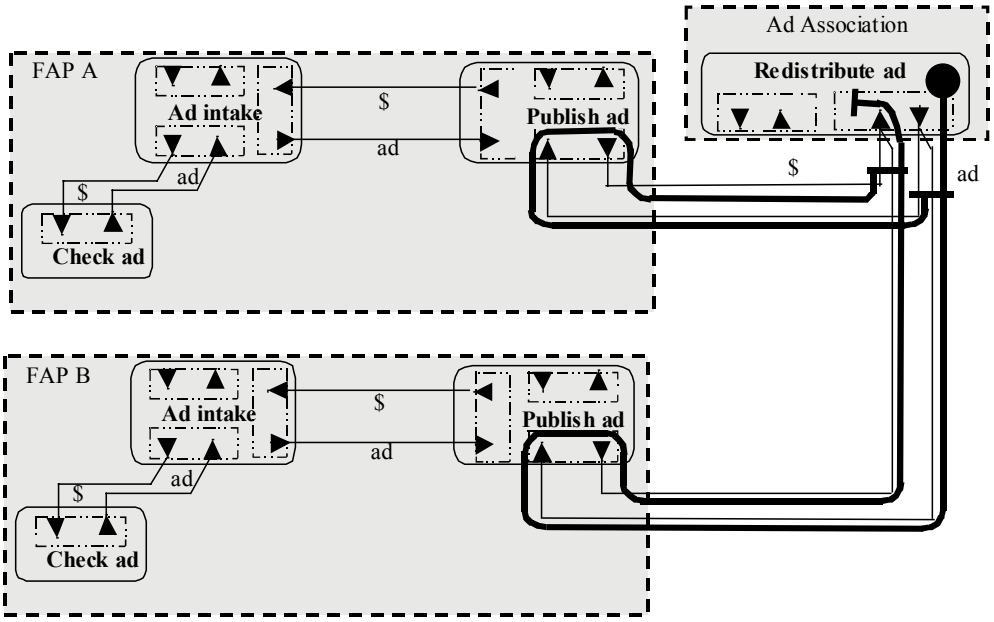

Figure 6 Business value area of the Ad Association – distribute ad scenario

Scenario (2) in Figure 6, *distribute ad*, shows value exchanges which occur between the actor responsible for redistribution of the ad, and FAPs. The Ad Association distributes the same value object (the ad) multiple times to different FAP. For distributed ad to a FAP, two value exchanges occur: (1) the Ad Assocation receives an amount of money in return for (2) the ad.

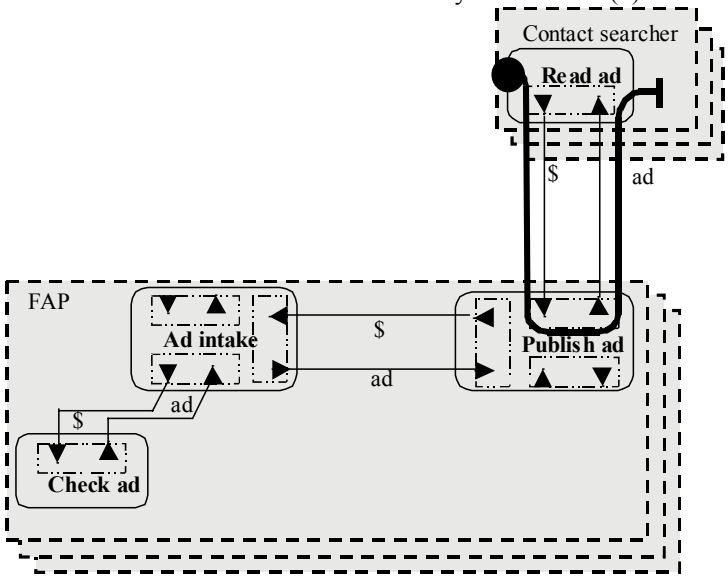

Figure 7 Bussiness value area of the Ad Association – read ad scenario

Scenario (3), Figure 7, *read ad*, has two value exchanges: (1) the contact searcher receives an ad from a FAP, and (2) pays for it.

Scenarios can be used to evaluate properties of the business value model, for instance the profitability of carrying out the scenario. On the business value level, we can get only get a first sight on this. We detail the evaluation during business process and software architecture design. Table 3 and Table 4 present sales revenues and costs for the various actors per scenario.

Sales↓ Actor→	FAP A	Other FAP	Ad Association	Contact searcher
Place ad	Redistribute ad $fee_{\text{Ad Assocation}}$	Check ad $fee_{\text{FAP A}}$	-	-

Redistribute ad	-		ΣRedistribution $fee_{other\ FAPs}$	-
Read ad	Read $fee_{contact\ searcher}$		-	-

Table 3 Sales revenues for each actor per scenario

Subscripts denote the direct source of the sales revenue. Most table entries speak for themselves. Note that in the *place ad* scenario, it possible that a different FAP from the FAP who handles the take in of an ad, earns money by checking ads.

Costs↓ Actor→	FAP A	Other FAP	Ad Association	Contact searcher
Place ad	Check ad $fee_{self\ or\ other\ FAP}$	-	Redistribution $fee_{FAP\ A}$	-
Redistribute ad	-	Redistribute ad $fee_{Ad\ Assocation}$		-
Read ad	-	-	-	Read ad fee_{FAP}

Table 4 Costs for each actor per scenario

Note that costs involved in checking an ad by the FAP who performs the take-in of an ad can differ from costs for checking an ad by another FAP. For the *redistribute ad* scenario, there is a redistribution fee to be paid to the ad association by each FAP *receiving* the ad and a redistribution fee to be paid by the Ad Assocation to the FAP *sending* the ad. The first fee is higher than the second fee.

3.3 e-business process model

The e-business process model illustrates processes to be carried out by actors, and messages interchanged between those actors, on a conceptual level. If important, usage of (human) resources are shown. A number of techniques have been developed to model business process adequately, amongst others UML activity diagrams with swimming lanes to represent actors [Fowler and Scott (1997)], or role-based process-modelling techniques [Ould (1995)]. In this paper, we choose for the latter. In Ould (1995)], a *role* is defined as a set of activities that are carried out by an actor in an organisation. An *activity* is what actors do in their roles. Between activities and therefore between roles *interactions* can occur. An interaction is an coordination between activities which has no implied direction. An interaction with no direction between two activities for instance might model to agree on something. However, interactions also can model exchange of something between activities, for instance goods or messages.

Figure 8 shows a process model which corresponds to the business value model. We show roles performed by actors and their interactions. Ould offers also facilities to do detailed process modeling. Working out a detailed process model can be done in a subsequent stage, but for now, the actors, roles and actors provide sufficient detail. The process model is illustrated by using scenarios.

As a starting point, we use the scenarios identified in the business value model: *place ad*, *redistribute ad* and *read ad*. Value activities are mapped on roles. Value exchanges are candidates for interactions between roles. However, value exchanges are not equal to interactions. Value exchanges denote things of value to (other) actors which do not always result in interactions between actors directly. On the other hand, interactions may be introduced which do not have their counterpart in value exchanges. In this process model, value exchanges regarding payments between value activities which are performed by the same FAP do not have counterparts in interactions. We assume a FAP is a administrative unit, so payments within a FAP are not handled. Some interactions are new, for instance the query asked by a contact searcher to a FAP, and the ad to be checked. The same scenarios as in the business value model are shown, however the paths now show a sequence of interactions between roles.

We use UCM-pools to show the use of precious resources by an activity. In Figure 8, a pool is used to show that for ad checking, a person is necessary. On the scenario path, a person is retrieved from a resource pool. After checking the ad, the person is placed back in the resource pool .

Note the synchronization bar (with the N:1 indication) in the *redistribute ad* and the *publish ad* activity. This bar means in case of the *redistribute ad* activity that a number of ads are collected, before paying for them. So, only one payment has to be made for a large number of ads. This refers to a mechanism of aggregate payment [Choi *et al.* (1997)]; it is much cheaper to handle one big payment instead of a large number of small ones. The same holds for the *publishing ad* activity.

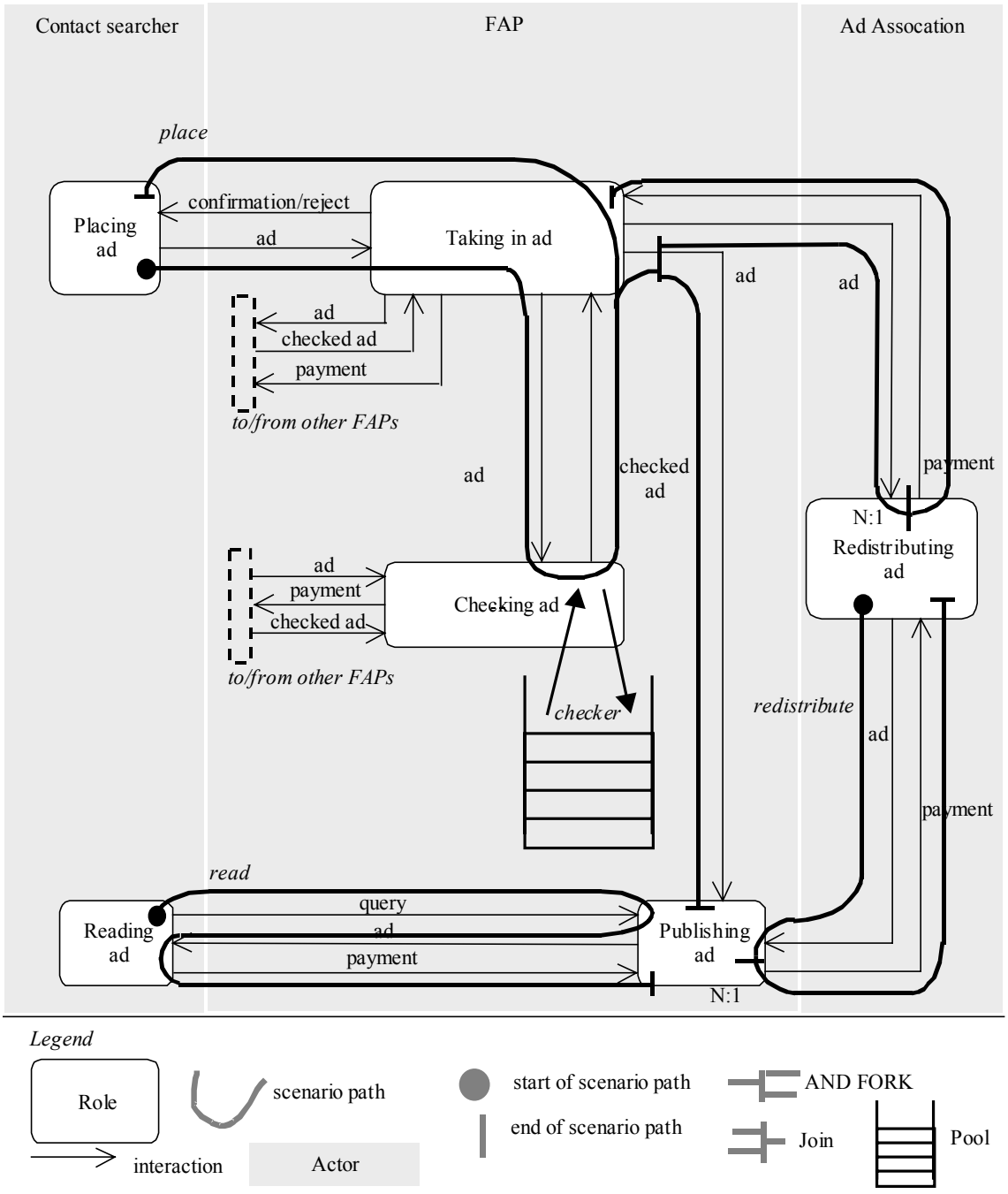

Figure 8 Business process model of the Ad Assocation

After making the business process model, we evaluate the scenarios with respect to income and expense again. The table below show extra expenses or expense refinements per scenario. The incoming remain as in Table 3.

Subscripts denote the direct source of the cost. Most table entries speak for themselves. Note that in the *place ad* scenario, it possible that a different FAP from the FAP who handles the take in of an ad, earns money by checking ads.

Cost↓ Actor→	FAP A	Other FAP	Ad Association	Contact searcher
Place ad	Check ad fee *personal cost*	-	-	-
Redistribute ad	-	Redistribute ad fee	Redistribution fee	-

		per N ads $_{\text{Ad Assocation}}$	*per N ads*$_{\text{FAP A}}$	
Read ad	-	-	-	Read ad fee$_{\text{FAP}}$

Table 5 Costs for each actor per scenario

3.4 e-software architecture

Typically, many software architectures can be devised that all realize a given business value and process architecture. The usual approach is to start with a number of candidate architectures and to perform an evaluation to pinpoint the most promising architecture, which is then elaborated further. Candidate architectures are usually based on architectures that have worked well in similar situations. Also, previous experiences of the architect may play a dominant role in selecting candidate architectures [Bass *et al.* (1997)]. This is not necessarily a bad thing. The point is that we do not have to design the best architecture (provided there is one), but rather an architecture that satisfies the preset requirements.

The candidate software architectures will be evaluated by taking the following quality attributes into account.

Performance. FAPs should respond quickly to requests for reading Ads. Note that the placement of Ads may take some time, since a client usually offers a single Ad and this Ad will be checked for bad language by a human.

Availability. It should be clear that a FAP should be accessible at all times in order to be competitive.

Maintainability. The nature of e-commerce applications will change rapidly as new business value models emerge and new technologies are developed that enable different approaches to doing business. Therefore, the software architecture must be designed with maintainability in mind.

Security. Since we are dealing with confidential information and electronic payment, security issues must be addressed to establish a sufficient safety level.

More quality attributes could be added to this list. We refrain from doing so because the intent is to show the approach to software architecture by taking the FAP case as an example. Adding more quality attributes does not help in further clarifying the approach.

Two candidate software architectures will be presented for the Ad Association case. Both are based on a 3-tier architectural style in which a system is decomposed into three components, (1) the database, (2) the business logic, and (3) the user interface. This division emphasizes the principle of separation of concerns: a component should be responsible for one task only. Adhering to this principle minimizes the impact of change of one component on other ones. In addition, a 3-tier architectural style caters for distribution and scalability.

A software architecture should be capable of dealing with our previous identified scenarios: *place ad*, *redistribute ad* and *read ad*. The following two architectural variations have been designed: (1) a decentralized database (Figure 9) and (2) a centralized database (Figure 10). In the first alternative, each FAP maintains its own set of Ads that are offered to its clients while in the second alternative, the Ad Association maintains the set of all Ads. A client's request for an Ad will be forwarded by a FAP to the Ad Association.

For the sake of clarity, value exchanges (e.g., an amount of money in return of a delivered product or service) have not been modeled explicitly in the software architectures. A connection between components represents a complete value exchange as is being modeled in the business value architecture area.

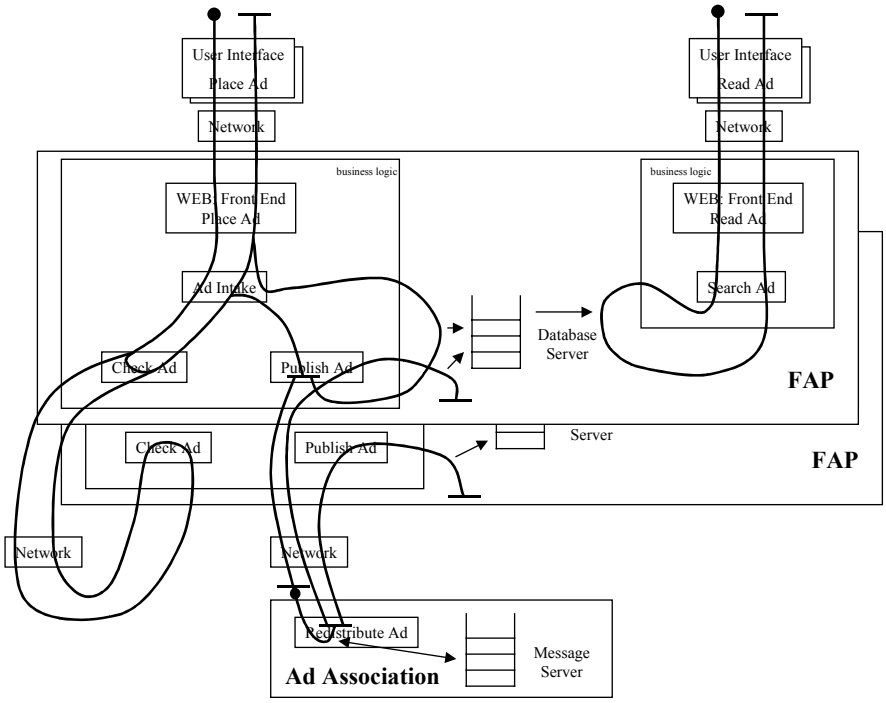

Figure 9 A decentralized architecture.

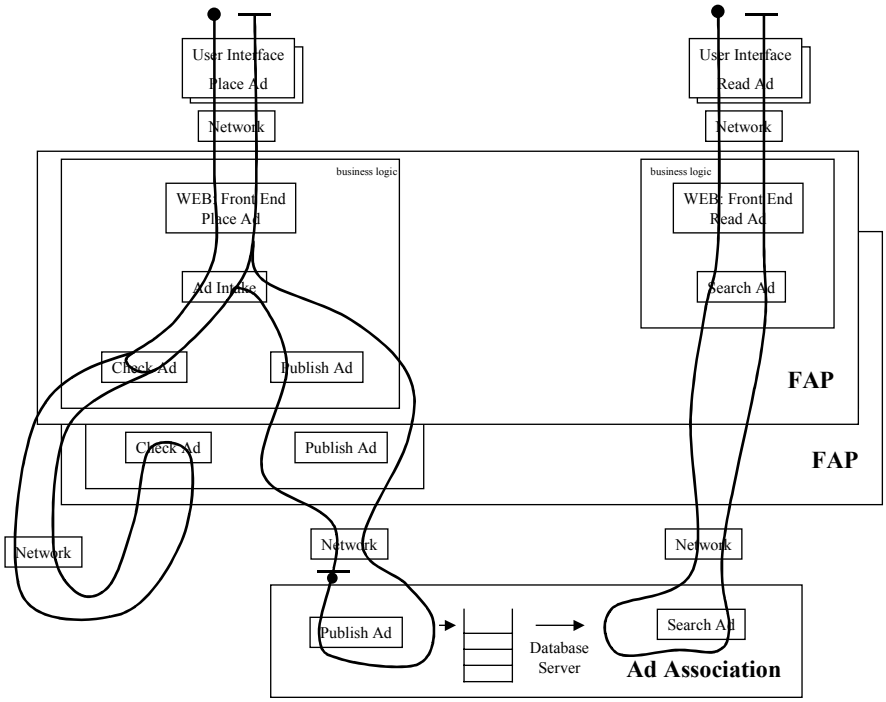

Figure 10 A centralized architecture.

Note that for the centralized architecture, the redistribution scenario and the read ad scenario are represented using one UCM path. Also note that each ad is redistributed on demand of the reading contact searcher while an FAP only pays *once* for an ad.

Having devised two candidate software architectures, we are now in the position to compare them. The pros and cons of these two software architectures with respect to the quality attributes are summarized in Table 6. It is beyond the scope of this paper to compare the three architectures in detail since this requires detailed knowledge of the components involved (e.g., performance and availability figures of database servers). In general, however, one can say that the decentralized database solution is better than the other solution as far as performance, availability and security are concerned. However, the centralized database solution is easier to maintain, since it is easier to upgrade a single site than having

to maintain a distributed solution. But again, detailed insight in the components being deployed is required to substantiate these claims.

	Decentralized	Centralized
Performance	+	–
Availability	+	–
Maintainability	–	+
Security	+	±

Table 6 Comparing three different architectures

By adding weight factors to quality attributes, we can pinpoint the best software architecture from a technical point of view. However, as will become apparent in the next section, this is not necessarily the software architecture that will be elaborated further.

3.5 Evaluation

In the previous section we have seen how candidate software architectures can be evaluated from a technical point of view by taking quality attributes like performance, availability, maintainability, and security into account. However, the technical point of view is not the only view to consider. As remarked before, a software architecture should support the business. The prime view from the business perspective is an economical one. In particular, the costs are crucial. For this reason, the costs involved in a software architecture should be assessed. It is important to realize that these costs can only be identified by actually designing software architectures.

In Table 7 and Table 8 we have accounted for the costs for implementing software architectures based on a decentralized and centralized solution, respectively. Note that sales revenues already have been identified in the business model. We distinguish the following costs: (1) network costs, (2) webserver/application costs, (3) database costs, and (4) message costs. The network costs involve costs to communicate via a computer network with another actor. We assume that communication is handled via the Internet. This requires an Internet connection which has a low, medium or high bandwidth consumption for the particular scenario. The classification of a scenario into one of these three connection classes depends on the bandwith consumption per scenario and the volume of the scenario executions in a timeframe. We assume that the number of times ads are read exceed the number of times ads are placed. Furthermore, we assume that a placed ad needs to redistributed to *each* FAP. We relate the three classes of connection to respectively low, medium or high network costs. The webserver/applicationserver costs represent costs involved in the business logic layer. The database server costs comprise all costs for having a local or central database server. Finally, for the decentralised scenario, we assume a message server (e.g. an SMTP server), which introduces costs. All these costs are accounted for on a per scenario basis. This means that no fixed costs exists, these are allocated to each individual execution of a scenario, based on the expected number of executions per time-frame.

Notice that web/application server, database server, message server and network server are not part of the business value model, so their impact on the costs cannot be assessed by evaluating a business value model in isolation.

Costs↓ Actor→	FAP	Ad Association	Contact searcher
Place ad	FAP handling *placing* ad: Web/application Check ad $fee_{self\ or\ other\ FAP}$ Low network(for remote checking) Medium network(for distribution from FAP to AA)	Redistribution $fee_{FAP\ placing}$ Medium network(for distribution from FAP to AA)	Low network
Redistribute ad	FAP *receiving* the ad: Redistribute ad $fee_{Ad\ Assocation}$ Database Medium network (for distribution from AA to FAP)	Medium network Message	-
Read ad	FAP offering *reading* the ad: Web/application Database High network	-	Read ad fee_{FAP} Low network

Table 7 Decentralized database software architecture

Costs↓ Actor→	FAP	Ad Association	Contact searcher
Place ad	FAP handling *placing* ad: Web/application Check ad $fee_{self\ or\ other\ FAP}$ Low network(for remote checking) Medium network(for distribution from FAP to AA)	Redistribution $fee_{FAP\ A}$ Medium network(for distribution from FAP to AA)	Low network
Redistribute ad and read ad	FAP *receiving* the ad and offering *reading*: Redistribute ad $fee_{Ad\ Assocation}$ Web/application High network	Database High network	Read ad fee_{FAP} Low network

Table 8 Centralized Database Software Architecture

From these tables the following observations can be made.

Ad Association perspective. If we assume that network costs are much cheaper than database server costs, and we assume that a message server is much cheaper than a database server, the redistribution costs in the centralized solution are greater than the redistribution costs in the decentralized solution. This assumption about network and database server costs seems valid. There is trend that network costs will be eliminated altogether in the future because of the liberalization of the telecom market. The second assumption is reasonable too. A mail server can be implemented using a low-cost machine with nearly free software. A databaseserver which is capable of a large number of queries and updates per minute, is a high investment, both in hardware and software. Also maintenance costs are substantial, for instance for performance tuning. Recall that redistribution costs are charged to FAPs and eventually the Ad readers. It is interesting to observe that the Ad Association becomes a more dominant player in the centralized database solution, that is, more cash is flowing towards the AA.

FAP perspective (in the role of offering a *read ad* service). The database costs in the centralized solution are less than the database costs in the decentralized solution under the assumption that a single database server is cheaper than N smaller database servers having the same total capacity as the single server. If we neglect the network costs, the FAPs costs are predominated by the database costs, and therefore, the centralized database solution is most cost-effective.

FAP perspective (in the role of handing *placement of an ad*). The costs are indifferent with respect to the two software architectures.

An interesting conclusion that can be drawn from these observations is that from an cost-based point of view the centralized database architecture is a better solution for everyone. However, as discussed in the previous section, from a technical point of view one should favor the decentralized database architecture. Of course, it is possible to invest in a more powerful and fault-tolerant database server and in high-speed networks connecting the FAPs with the Ad Association in order to offer a centralized solution that is on par with the decentralized solution. However, this involves additional costs, which have to be analyzed carefully.

In discussing these results with a number of FAPs and the Ad Association it seemed that, besides cost considerations, other issues are important in deciding for a centralized or decentralized variant. In the centralized variant, the value activity *redistribute ad* increases in importance, on the expense the value activity *publish ad* performed by FAPs. It can be seen as a shift in power from FAPs to the Ad Association if choosing for the centralized variant and therefore some FAPs are not in favor of the centralized variant, despite the decreasing costs for them. Also, it is not clear what the business consequences are if a cheaper (centralized) solution will be chosen. Does the contact have to pay less compared to the decentralized variant? Or makes the Ad Association more profit?

Apart from the aforementioned emotional considerations, the important thing is that the *e^3-VALUE* framework provides the means to play *what-if* games on objective grounds.

4 Conclusions and Lessons Learned

We have argued that e-commerce system should be designed from different perspectives, which we call architectural areas. We have identified three architectural areas that we consider crucial for the development of e-commerce systems. The e-business value model enabled us to discuss with

stakeholders the objects of value offered to customers and the assignment of value adding activities to actors. The e-business process area detailed the e-business value model in message exchanges between roles performed by actors. It clarifies which value exchanges map onto the actual messages being sent and the usage of resources. The e-software architecture shows the possible realizations of the value and process model by means of software and hardware components.

We used scenarios (UCMs) to relate the architectural areas and to assess the economical en technical feasibility of the commerce system. By assessing the scenario for each architectural area, we gained global insight in costs involved when realizing the commerce-system, without loosing ourselves in details.

However, the designs revealed an unexpected characteristic of the commerce system to the stakeholders. The centralized variant was, despite its lowest costs, not the most interesting alternative for all stakeholders, because it implies a shift in power from local FAPs to the Ad Association. This was not visible at the e-value model level at a first sight.

References

1. L. Bass, P. Clements, R. Kazman, 1997. Software Architectures in Practice. Reading, Massachusetts: Addison-Wesley.
2. R. J. A. Buhr ,1998. Use Case Maps as Architectural Entities for Complex Systems. *IEEE Transactions on Software Engineering* Vol. 24(No. 12):1131-55.
3. L. J. Camp, 1996. Privacy & Reliability in Internet Commerce. Pittsburgh: Carnegie Mellon University, School of Computer Science.
4. S.-Y. Choi, D. O. Stahl, A. B. Whinston, 1997. The economics of doing business in the electronic marketplace. Indianapolis: Macmillan Technical Publishing.
5. M. Fowler, K. Scott, 1997. UML Distilled - Applying the Standard Object Modeling Language. Reading, Massachusetts: Addison-Wesley.
6. J. Gordijn and J.C. van Vliet ,1999. On the Interaction between Business Models and Software Architecture in Electronic Commerce. *To Be Presented on the European Software Engineering Conference '99, Toulouse* .
7. J. Gordijn, J.M. Akkermans and J.C. van Vliet ,1999. Requirements for e-commerc applications are created rather than elicited. *NOSA'99 - Proceedings of the second nordic workshop on software architecture*. J. Bosch, Ronneby, Sweden.
8. J. Hagel III, A. G. Armstrong, 1997. Net Gain - Expanding markets trough virtual communities. Boston Massachusetts: Harvard Business School Press.
9. R. S. Kaplan, D. P. Norton, 1996. The Balanced Scorecard. Boston, Massachusetts: Harvard Business School Press.
10. Marco Iansiti and A. MacCormack ,1997. Developing products on the Internet Time. *Harvard Business Review* (September-October 1997).
11. R. R. R. Normann, 1994. Designing Interactive Strategy - From Value Chain to Value Constellation. 2 ed. Chichester: John Wiley & Sons.
12. M. A. Ould, 1995. Business Processes - Modelling and Analysis for the Re-engineering and Improvement. Chichester, England: John Wiley & Sons Ltd.
13. M. E. Porter and V.E. Millar ,1985. How information gives you competitive advantage. *Harvard Business Review* (July-August 1985):149-60.
14. J. F. Rayport and J.J. Sviokla ,1995. Exploiting the Virtual Value Chain. *Harvard Business Review* (November-December 1995).
15. C. Shapiro, H. R. Varian, 1999. Information Rules. Boston, Massachusetts: Harvard Business School Press.
16. R.J. Wieringa (1996). *Requirements Engineering: Frameworks for Understanding*. John Wiley & Sons Ltd., Chichester.