

Requirements for Electronic Commerce Applications are created rather than elicited¹

Jaap Gordijn

Vrije Universiteit

De Boelelaan 1081a, 1081 HV Amsterdam,

The Netherlands, fax: +31 (0)20 444 7653

e-mail: gordijn@cs.vu.nl

Bakkenist Management Consultants

Postbox 23103, 1100 DP Amsterdam, The Netherlands

Hans Akkermans

Vrije Universiteit

e-mail: hansakkermans@cs.vu.nl

AKMC Knowledge Management

Klareweid 19, 1831 BV Koedijk, The Netherlands

Hans van Vliet

Vrije Universiteit

e-mail: hans@cs.vu.nl

Keywords

Business model, process model, software architecture, electronic commerce, requirements creation, requirements elicitation, value chain, value constellation.

Abstract

Electronic commerce applications are a challenging new class of information systems. Due to the ever-increasing technical possibilities, new, yet undiscovered ways of doing business are possible. Therefore, requirements for e-commerce systems can hardly be elicited but instead need to be *created* because elicitation requires a dormant view of users on requirements. Also, E-commerce systems are, in contrast to other information systems, an intrinsic part of the product or service a company offers to others.

Therefore, developing the business model, which represents the offering, and defining the requirements for a supporting e-commerce system, is not a sequential process and, neither can it be done in a decoupled way. Current requirements engineering methodology does not support requirements creation nor deals with information system requirements as an intrinsic part of developing business models. We outline a structured approach to e-commerce requirements creation called *e³-VALUE*. This approach identifies three views to be developed: (1) the e-business model view, (2) the e-business process view and (3), the e-software architecture view. We discuss the e-business model view in more detail and discuss a more formal way to represent business models. In our interpretation of 'business model', the notion of value is a central theme. We illustrate our *e³-VALUE* by presenting an experience report.

1 Introduction

Requirements elicitation is the commonly used term for the very first stage of a systems development project [Kotonya and Sommerville (1998)]. It suggests that requirements are already somewhere present in the minds of prospective system users, and thus only have to be discovered by the system developers. This may be so for many conventional information systems, but it is certainly misleading in e-commerce applications. Here, system requirements cannot be discovered because usually the underlying business activities themselves are also new and in a state of being designed. Thus, system requirements have to be created rather than discovered. The creation of realistic e-commerce requirements and e-business model development has to be done in an integrated way because e-commerce systems are an intrinsic part of the way doing business and should reflect the business well. Moreover, due to ever increasing technical

¹ An extended version of this paper has been submitted elsewhere

possibilities, new ways of doing business are in reach, so technology for e-commerce systems even influences and enables new business models. Currently, requirements methodology for such situations is inadequate. In this paper, we outline a structured approach, called *e³-VALUE*, that gives a better grip on requirements elicitation in open-ended business situations as commonly encountered in innovative areas as electronic commerce.

In this paper, we propose a framework of views to be developed when designing an e-commerce application. This framework consists of three views: (1) the e-business model view, (2) the e-business process view, and (3) the e-software architecture view. This provides a partial but useful separation of e-business design concerns. We suggest that the right entry point to e-commerce application development is value analysis for the various actors involved. Because the concepts which should be present in an e-commerce business model are hardly known, we propose a semi-formal way to represent an e-business model, showing the essentials of doing business between different actors. Our e-business model centres around the key concept of *value*, and how value is created and exchanged within the stakeholder network. This value-configuration model can be constructed with the help of conceptual modelling techniques adapted from the information systems analysis field. Such a semi-formal value-configuration model supports e-business design and decision-making in a much more structured fashion than allowed by current methods from the business and management literature. An important contribution of our paper is its proposal of a limited baseline set of generic concepts to be used in value modelling. We employ our value-configuration model to generate high-level requirements on e-commerce application systems. We discuss how value-based business considerations do have an identifiable impact upon architecture design of e-commerce applications. Both these features of our approach, value modelling for e-business design and subsequent identification of associated requirements for the envisaged software architecture, advance requirements methodology for creative business areas such as electronic commerce.

This paper is structured as follows. Sec. 2 sketches our overall *e³-VALUE* framework for electronic commerce applications. This *e³-VALUE* approach is then illustrated by an industrial experience report concerning international web-based advertising. The experience report is based upon our consultancy experiences in the e-business field. Sec. 4 summarises the general key points of our approach to electronic commerce applications.

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

2.1 Three views in e-business requirements

The development of an electronic commerce business application is not, as many see it, a requirement elicitation process [Kotonya and Sommerville (1998), Potts (1993)], which presupposes that stakeholders have tacit requirements that “just need to be extracted” by system developers. Instead, it is much better seen as a requirements *creation* process. This is caused by the novel nature of electronic commerce applications, the innovative character of e-business models, and the rapid developments in internet and web technology. Moreover, the design of an electronic commerce system is not in the first place an IT-oriented activity. Rather, it consists of very different types of design problems which have to be tackled simultaneously. We distinguish the following design problems: (1) the business model design, (2) the business process design, and (3) the software architecture design (Figure 1). One reason to distinguish these design processes is the separation of concerns for different stakeholders. For example, general managers will want to take decisions about business models, but usually do not wish to be strongly involved in decisions regarding the software architecture of a system. Thus, requirements creation in electronic commerce applications must be grounded in different stakeholder *views* upon the systems to be built.

An important open issue in electronic commerce is what areas are relevant (our answer being given in Figure 1), and how these areas are to be developed.

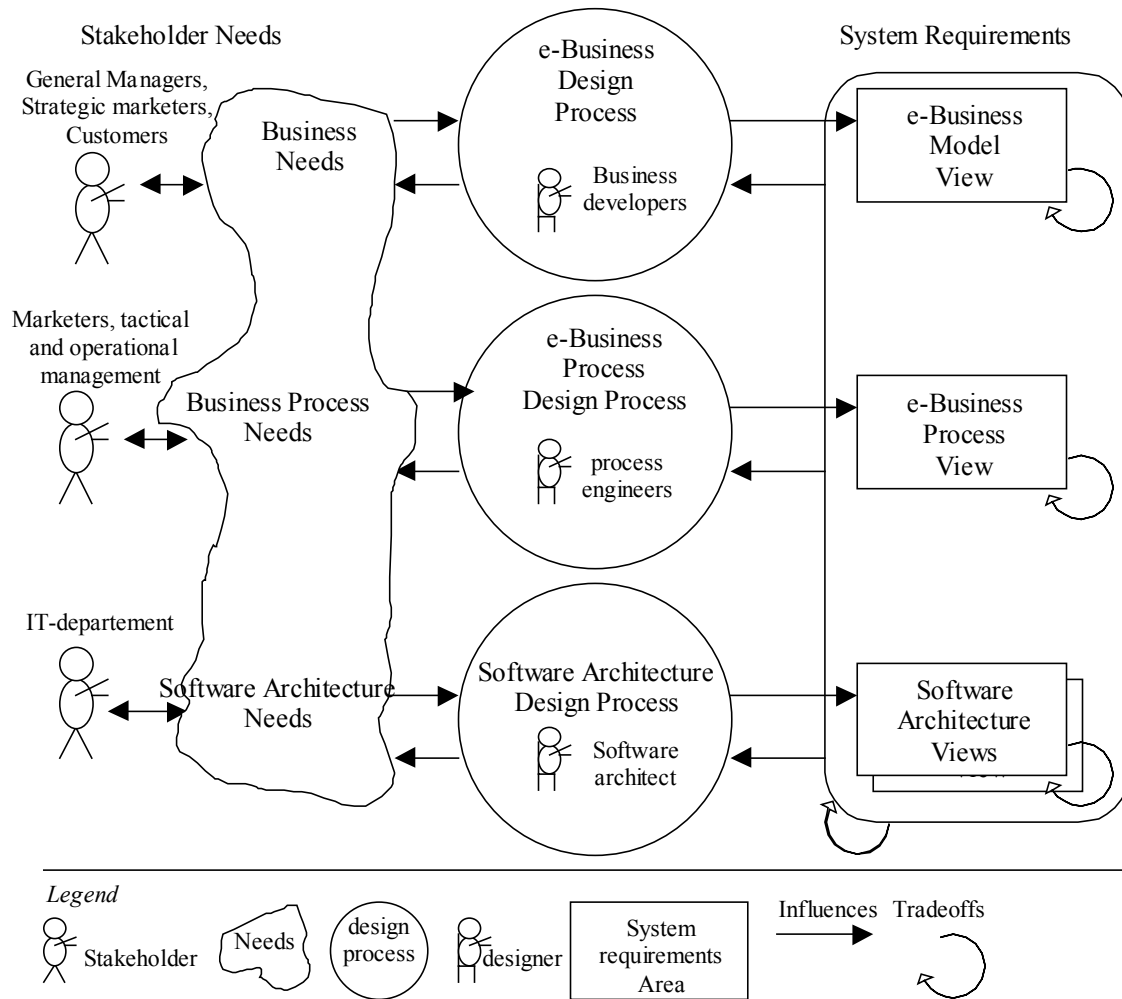


Figure 1 E-business design processes produce different views on system requirements, based on various stakeholder needs.

The top-level area of our electronic commerce framework concerns the electronic-commerce business model. The business model describes the way of doing business between actors, and so sparks off requirements at a business level. Stakeholders are general managers of companies participating in the execution of the business model, marketers and customers. Business developers are the primary designers of the model. An important reason to consider business models in some detail is that they are useful to analyse and solve tradeoffs between varying, and potentially conflicting, stakeholder interests. Tradeoffs appear in the separate design processes and should be addressed during these processes. However, tradeoffs may also exist between the business model, business process and software architecture. For instance, fraud constraints (as they are called at the business level) or security requirements (as they are called on the software architecture level) can either be addressed during business model design by moderating potential conflicts of interest, during business process design by creating a fraud-prevention protocol or by a software architecture design by applying secured components based on encryption technology, or all of these [Gordijn and van Vliet (1999)].

Design of business models is, therefore, a process which needs guidance by itself. However, there is hardly any scientific consensus or sound method how a business model should be represented. How to do this is a main focus in our paper. A key idea of our approach is that structured *value* analysis is a crucial activity in business model design, and that this enables one to properly address various design requirements tradeoffs

that exist across the three viewpoints depicted in Figure 1. 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, there is a need for a more formal representation of business models, and we propose such a representation below.

The business process area, the middle level in Figure 1, shows how activities should be performed and by whom. Also messages exchanged between activities performed by actors are represented. Stakeholders are managers on tactical and operational level since they are responsible for carrying out most processes, and marketers regarding detailed buy flows. Business process engineers are the most important designers. To represent a business process view a number of techniques are suitable, for instance UML activity diagrams with swimming lanes to represent actors [Fowler and Scott (1997)], high-level Petri Nets [Hofman (1994), van Hee (1994)], or role-based process-modelling techniques [Ould (1995)]. The CommonKADS approach for knowledge engineering and management [Schreiber *et al.* (In press)] has interesting facilities to model multiple-actor aspects of the business process view. It distinguishes a communication model which can be used to represent communication between actors on different levels of abstraction. Also, it offers different types of communication between actors.

The software architecture area, the bottom of Figure 1, shows the information system and its constitutive software components. In [Bass *et al.* (1997)], a software architecture is defined as the structure or structures of a system which comprises software components, the externally visible properties of those components, and the relationships among them. Multiple architectural views are used to represent different types of structures. Four commonly used views are the logical view, process view, physical view and development view [Kruchten (1995)]. Representation techniques for software architectures are in an early phase of development. Stakeholder is the IT-department in its role of managing and maintaining systems. The software architect is the most important designer. The focus in this paper, however, is the representation of business models and how this representation can be exploited to derive architectural requirements.

2.2 Business model view: core concepts underlying value-configuration modelling

We propose that the central concept in any business model 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 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 core concepts needed to represent a semi-formal business model.

Actor. An actor is an independent entity such as a company or a person. An actor corresponds to company A,B,C or person X,Y,Z. Actors perform one or more value activities.

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 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. However, we are not yet interested in the actual way of performing these activities: this reflects the separation of concerns, and is the main concern in business process design (e-business process view in Figure 1). Value activities for a specific case are represented as 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 leaf value activities have a unique mapping 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.

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. A value activity may have multiple value interfaces. There are two motivations for having multiple interfaces. 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.

3 An e-commerce experience report: the Ad Association

3.1 The Ad Association

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 report, 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. There are a number of business objectives which are important. First, FAPs want 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 brandnames 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 marketshare 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.

First, we present a business model with the FAPs as the primary actors. To show that multiple business models are possible, we next present a business model with the Ad Association as the primary actor. These differ in the kind of value activities and the assignment of those activities to actors.

3.2 A FAP centred business model

In the FAP centred business model, the following actors participate: (1) contact searcher, (2) FAPs and (3), Ad Association. Ads have to be placed and read by contact searchers. This results in the value activities (1) place ad and (2) read ad. Value activity (3), ad intake, ensures that an ad is placed internationally. 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. In Table 1 value activities, their value interfaces, value ports and value object types are concisely presented. Dashed lines separate different value interfaces of a value activity. Note that a value interface is defined by enumerating its value ports.

	<i>Value activities</i>	<i>Value interfaces consisting of value ports with value object type</i>
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
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
6	Redistribute ad	Out port: Read ad of type 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 1 Specific value activities, value interfaces, value ports and value object types for the FAP centred business model.

The business model is graphically depicted in Figure 2. Note that in this model, the value activities (and their value interfaces and value ports) have been assigned to actors and value activities are related to each other. The relationship between value ports shows possible exchanges of values between ports of actors. It does not show messages to be exchanged or the sequence of values to be exchanged.

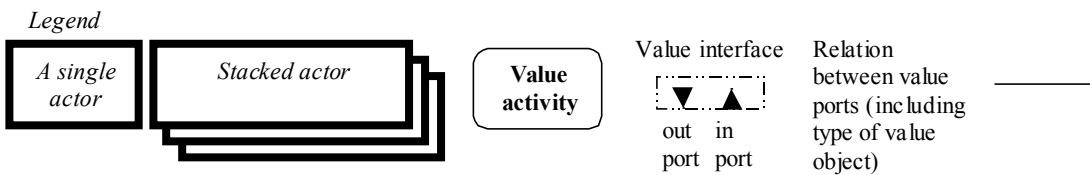
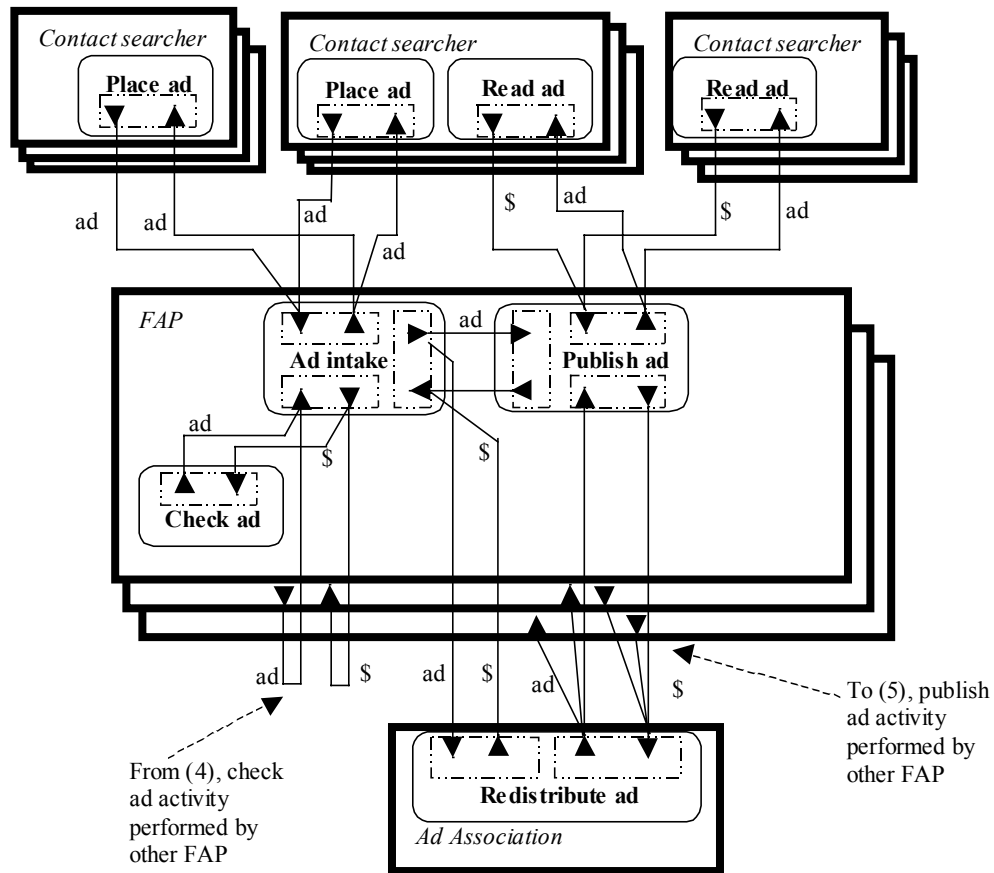


Figure 2 A FAP centred business model.

There are three types of contact searchers. The first type only places ads, the second type only read ads and the third type does both. In a business model, many similar actors can exist. Similar actors have (1) the same number of instances of specific value activity classes and belonging value interfaces and value ports, and (2), value ports are interconnected in the same way. Such actors are called stacked actors and are presented as stacked rectangles. In this business model, contact searchers as well as FAPs are examples of such actors. However, at the individual actor level, there might be differences. For instance, each FAP can have its own price for the ad to be read by searchers. Not all connections between value ports of actors are shown, especially in the case of stacked actors. For instance, only the value ports of the topmost stacked contact searcher and the first stacked FAP are shown while each contact searcher is connected to each FAP. In some cases, value ports of stacked actors themselves are connected. For instance, a FAP can ask another FAP to check an ad. An connection between the second stacked FAP and the topmost FAP represents this. Finally, note that value activity (5), publish ad, has two instances of the value interface modelling ads to be received from value activities. The first instance represents that an ad is received value activity (3), ad intake, performed by the same actor while the second instance models a received ad from the Ad Association. These are different value interfaces since the price for a received ad may depend on its source.

3.3 Alternative business model: Ad Association centred

This business model assumes that the Ad Association performs most value activities. The Ad Association exploits brandnames of FAPs by offering an international contact ad service using brandnames of each FAP. Contact searchers use the brandname of their local FAP (e.g. www.localfap.com) to *find* the service *offered* by the Ad Association. The Ad Association pays each FAP a fee for the usage of brandnames. Such a fee can be fixed but preferably depends on the number of ads placed and read mediated by the brandname of a FAP.

The actors/stakeholders are the same as identified in the previous section. A new value activity (7), maintain brand, is added to already identified value activities. This activity models the effort FAPs undertake to increase value of their brandname. In Table 2, value activities, value interfaces, value ports and value object types are presented. For brevity we only report changes.

	<i>Value activities</i>	<i>Value interfaces consisting of value ports with value object type</i>
3	Ad intake In port: Brand of type brand Out port: Payment for use of brandname of type money
5	Publish ad In port: Brand of type brand Out port: Payment for use of brandname of type money
7	Maintain brand	In port: Payment for use of brandname of type money Out port: Brand of type brand

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

Value activity (7), maintain brand. Its value interface has a value-out port representing that the brand is of value and a value-in port representing payment for using the brand.

Value activity (3), ad intake and (5), publish ad. Value activities interacting with contact searchers profit from the FAP's brandname. Therefore, value activities (3) and (5) have an additional value interface. These interfaces have a value-in port of type Brand each representing usage of the brandname and a value-out port representing payment for using the brandname.

The allocation of value activities to actors is shown in Figure 3.

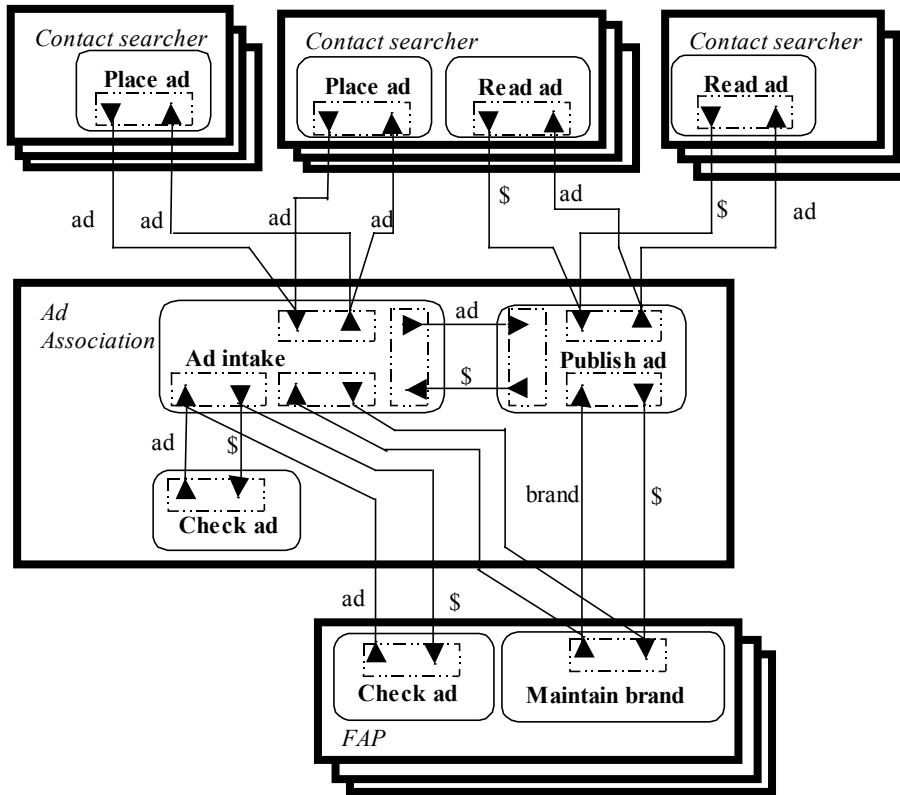


Figure 3 Ad Association centred business model.

Note that value activity (6), redistribute ad, is not allocated to any actor, since redistribution of ads is not necessary anymore.

3.4 Trade-offs in business modelling and implications for requirements on the information system architecture

The previously discussed business models have nearly the same value activities, value interfaces and value ports. The most important design trade-off shows up when allocating value activities to actors. In the first business model, FAPs perform the most important value activities ((3), ad-intake, (4), check ad, (5), publish ad) themselves and only use other FAPs for checking ads expressed in foreign languages. The Ad Association has a limited task in offering a redistribution infrastructure, for which FAPs pay a small fee. In the second business model, the Ad Association is the most dominant actor. FAPs add value by (1) offering a brandname to the Ad Association which is trusted and known by searchers and, by (2) checking ads expressed in a language the Ad Association can not handle. Since the Ad Association adds far more value in the second business model, we expect that FAPs will receive fewer revenues compared to the first model. However, the amount of work for FAPs also decreases compared to the first business model. Note that if the business model is implemented well, the contact searcher does not see anything of the choice made for the first or second business model.

A number of implications for the information system architecture can be drawn from the business models.

- The FAP centred business model requires a component, operating on behalf of the Ad Association, that redistributes ads to other FAPs. Such a component is not necessary in the Ad Association business model. The space in designing a redistribution component is quite large. One solution might be a message based component which receives ads from FAPs and redistributes ads using messages to other actors. An entirely other approach can be a database located at the Ad Association. All ads are stored in this database. A FAP queries this database when a contact searcher queries the FAP-site. The first

approach redistributes the ad only once, the latter solution delivers the ad to a FAP when necessary and in some cases multiple times. Both these solutions can respect the business model because in both cases it is possible that a FAP pays only once for each ad.

- The FAP centred business model supposes a number of components which should be implemented by each FAP, such as a webserver, a databaseserver containing ads, application components for reading, placing and checking ads, and a high quality connection to the internet, to be reachable by searchers. These components need to be implemented only once for the Ad Association centred business model.
- The Ad Association centred business model needs an accounting system which administrates the use of brandnames of individual FAPs. If FAPs are paid on a per-ad placed/read basis, such a system should be able to relate brandnames to ads placed and read. Moreover, such an administration should be trusted by the FAPs.
- Each FAP should be capable of handling payment. Setting up a payment infrastructure can be rather costly in some countries. In an Ad Association centred model, only one payment infrastructure needs to be set up. However, this infrastructure should presumably offer multiple payment methods because a particular method is not always applicable in every targeted country.

4 Conclusions and further research

Requirements engineering for electronic commerce applications has a first stage of requirements creation, rather than that system requirements can be elicited or acquired. A requirements engineering approach for such applications should address the issue that setting up business models, business processes and information systems are all design problems and have interrelated design choices. A reason for this is that business models are mostly undiscovered yet, amongst others due to ever increasing technological possibilities

We have put forward an approach to requirements creation for e-business that integrates both business and technology considerations. Discussed key aspects of our e^3 -*VALUE* approach are:

- Requirements creation should start from structured value-based analysis at the business level, within the relevant actor/stakeholder network in electronic commerce application.
- We have proposed a novel, semi-formal way to define a business model. We have identified a limited and generic set of core concepts (i.e., which can be the basis of an ontology) to construct a business model, including the notions of actors, value activities, objects, ports, and interfaces.
- The resulting value-configuration models enable one to analyse tradeoffs between different business models, and at the same time to generate high-level requirements regarding the electronic commerce systems architecture.

Our e^3 -*VALUE* approach has been demonstrated by means of a real-life web-based advertising based experience.

There is a definite need for an integral design view upon relevant business and technology matters in electronic commerce applications. Value-based business modelling, as developed in this paper, is a powerful tool to achieve such an integrated approach to business model decision-making and systems requirements engineering.

The next step in our research will concentrate on doing case-studies which relate our three identified views; business model, business process and software architecture view. We plan to use scenarios (based on use case maps [Buhr (1998)]) as the integrating glue between our three views.

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 at the European Software Engineering Conference '99, Toulouse*.
7. J. Hagel III, A. G. Armstrong, 1997. *Net Gain - Expanding markets through virtual communities*. Boston Massachusetts: Harvard Business School Press.
8. W. J. Hofman, 1994. *A conceptual model of a business transaction management system*. 's Hertogenbosch: UTN Publishers.
9. R. S. Kaplan, D. P. Norton, 1996. *The Balanced Scorecard*. Boston, Massachusetts: Harvard Business School Press.
10. G. Kotonya, I. Sommerville, 1998. *Requirements Engineering - Processes and Techniques*. Chichester, England: John Wiley and Sons Ltd.
11. P. B. Kruchten, 1995. The 4+1 view model of architecture. *IEEE Software* 12(6):42-50.
12. R. R. R. Normann, 1994. *Designing Interactive Strategy - From Value Chain to Value Constellation*. 2 ed. Chichester: John Wiley & Sons.
13. M. A. Ould, 1995. *Business Processes - Modelling and Analysis for the Re-engineering and Improvement*. Chichester, England: John Wiley & Sons Ltd.
14. M. E. Porter and V.E. Millar, 1985. How information gives you competitive advantage. *Harvard Business Review* (July-August):149-60.
15. C. Potts, 1993. Software Engineering Research Revisited. *IEEE Software* 10(5):19-28.
16. A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. d. Hoog, N. R. Shadbolt, W. v. d. Velde, B. J. Wielinga, In press. *Knowledge Engineering and Management - The CommonKADS Methodology*. Boston, MA: MIT Press.
17. C. Shapiro, H. R. Varian, 1999. *Information Rules*. Boston, Massachusetts: Harvard Business School Press.
18. K. M. van Hee, 1994. *Information systems engineering - A formal approach*. Cambridge: Cambridge University Press.