5 eBusiness Value Modelling Using the *e³value* Ontology

Jaap Gordijn

Abstract

A first step in developing eBusiness ideas is to understand such an idea thoroughly. In the recent past, industry has clearly demonstrated that such an understanding lacks or at least is insufficient, resulting in failures and sometimes even bankruptcies. In this chapter we present an approach to design an eBusiness model, called the *e³value* approach. This approach is ontologically founded in business science, marketing and axiology, but exploits rigorous conceptual modelling as a way of working known from computer science. The aim is that an *e³value* eBusiness model contributes to a better and shared understanding of the idea at stake, specifically with the respect to its profit drivers. If the model is attributed with various assumptions, such as economic valuation of objects produced, distributed, and consumed, we can derive profitability sheets using the business model. These can be used to assess whether the idea seems to be profitable for all actors involved in the idea. We illustrate the *e³value* approach by a project we carried out for an Internet Service Provider.

Introduction

Over the past few years, many innovative eBusiness ideas have been considered. Such ideas reveal new value propositions, which are enabled by new technological possibilities, such as the widespread use of the Internet and associated technologies. The late 1990s was characterised by much hype about eBusiness. More recently, it became clear that numerous eBusiness initiatives were unsuccessful (Shama 2001). Many firms were able to create new revenue streams from eBusiness. Those which relied upon these revenue streams have largely gone out of business. An important reason for the failure of most eBusiness ventures was the lack of a sound value proposition for customers. Moreover, many ventures did not contribute sufficiently to the profitability of firms, especially where their efforts focused more upon maximizing market share and establishing a trusted brand name rather than revenue generation. Notwithstanding this point, many believe that eBusiness has the potential for offering firms the opportunity to utilise the Internet and related technical innovations in a profitable way. Indeed, some industries need to develop new value propositions if they are to survive. For instance, the digital content industry is facing challenges with respect to new value propositions utilizing Internet technology, e.g. how to earn money by streaming music to an end-consumer's device.

One of the key-problems with innovative eBusiness ideas is that, initially, they tend to be formulated vaguely, and often lack a precise description. As a result, many innovative eBusiness ideas are somewhat unfocused and inaccurate. This makes it different to put the idea into operation, and to develop a supporting information system. What is needed is an in-depth exploration process of an eBusiness idea, to understand the idea better as well as to formulate it more precisely, and to focus the idea into a direction that is feasible from an economical and technical perspective.

This chapter provides an ontological perspective on the exploration of such innovative eBusiness ideas. Our *e³value* ontology (see also (Gordijn 2002) to do so is on the one hand based on the analysis of *economic value* creation, distribution, and consumption in a multi-actor network. On the other hand, the *e³value* ontology is founded on *requirements engineering* and underlying *conceptual modeling* techniques, borrowed from the information systems community. Requirements engineering is the process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained (Loucopoulos 1995).

This chapter is structured as follows. Section 5.1 explains why we need an eBusiness ontology. Hereafter, section 5.2 introduces the *e³value* ontology in detail, and explains the use of the ontology by discussing a real-life project on Internet service provisioning that successfully applied the ontology. Section 5.3

extends the baseline ontology with facilities for operational scenarios, which are used to do profitability sensitivity analysis (section 5.4). Section 5.5 discusses related ontologies, such as the AIAI enterprise ontology, the TOVE ontology and the REA ontology. Finally, section 5.6 presents our conclusions and suggestions for further research.

5.1 Why an eBusiness model ontology?

Before presenting the *e³value* ontology in detail, we first discuss the rationale for the ontology. In short, the ontology is intended to design and analyse innovative eBusiness models. We restrict ourselves to *innovative* eBusiness models, meaning that we focus on eBusiness models with value propositions that are not understood by the large audience. Furthermore, we look at eBusiness models that are about doing business transactions between enterprises and/or end-consumers, often referred to as *e-commerce*.

As we will demonstrate later in the chapter, the main purpose of our *e³value* ontology is to articulate eBusiness models for networked enterprises for the following reasons:

- to reach a better understanding of the eBusiness model by the stakeholders involved, and
- to be able to do an analysis and profitability assessment of the eBusiness model for all parties involved.

5.1.1 Reaching a better understanding

EBusiness models often suppose that a *consortium* of enterprises *jointly* deliver a service to end-customers, in contrast to traditional business models where a *single* supplier offers a product to a *single* customer. Such multi-enterprise offerings require that all participating parties have a common understanding of the offering to be supplied. We have been involved in a series of business development tracks, and many of such multi-enterprise offerings result in unclear, and sometimes even inconsistent offerings. A main cause is mis-interpretation of the eBusiness idea underlying the offering.

In addition, even in a single enterprise, mis-interpretations of an eBusiness idea occur because different stakeholders are involved while formulating such an idea. We encountered stakeholders on the CxO level, but also parties responsible for design and execution of business processes (many eBusiness projects still fail because a good idea is not translated into consequences for operations), and ICT stakeholders (eBusiness relies on the enabling role of ICT). So, development of an eBusiness idea often leads to mis-interpretation of such an idea due to involvement of many enterprises, and a broad range of stakeholders representing these enterprises.

How can we contribute to avoid this mis-understanding? For this purpose, our *e³value* methodology (Gordijn 2002) provides an *ontology* to conceptualize and to visualize an eBusiness idea. An ontology provides concepts, relations between these, and rules which are supposed to be interpreted the *same way* by stakeholders, to conceptualize a specific domain. 'Conceptualization' means: describing rather formally a Universe of Discourse (UoD) (e.g. a business idea) to allow for understanding of, and reasoning about such a UoD. To create the required common understanding, our ontology borrows accepted terminology from the realm of business sciences, more specifically terminology on dynamic value constellations (Tapscott 2002, Normann 1993, Normann 1994, Porter 2001), marketing (Kotler 1998) and axiology (Holbrook 1999). For instance, *e³value* concepts are: *actor*, *value exchange*, *value activity*, and *value object*. Using these notions, we model networked constellations of enterprises and end-consumers, who create, distribute and consume things of economic value.

For conceptualization many description languages can be used. These languages differ in the statements they make about a UoD, in their level of formality, and also in their intended users. Since our audience consists of CxO's and business analysts, we have chosen for a *graphical* language. Most of these people do not have the time or the skills to read textually represented formal documents. To put it differently: A picture says more than 1000 words. The realm of computer science has invented many (semi) formal graphical languages to be able to easily communicate complicated aspects of computer software. As such our *e³value* approach utilizes terminology from business science, but borrows representation and visualization methodology from computer science.

5.1.2 Analysing an eBusiness model

In the recent past, industry has shown that many prospective eBusiness models have not been thoroughly analysed before put into practice. There were many examples of poor eBusiness models, leading to the failure of tens of thousands of dot.coms. This was largely because new ventures, having received first-round venture capital, failed to generate revenues, and therefore the ability to secure second-round funding critical to their survival (Cassidy, 2002).

The *e*^{3}*value* approach provides constructs to represent a networked business model, consisting of actors (enterprises and end-consumers) and what they exchange of economic value with each other. If we ask parties to assign economic value to objects they provide and obtain, we can reason about potential profitability of the eBusiness models. Moreover, we may use strategic scenario decision taking techniques to do sensitivity analysis on the eBusiness model under consideration. Since eBusiness models often suppose offerings provisioned by multiple enterprises rather than by one, it is important that all these enterprises have a reasonable chance to make profit. Otherwise, the multi-party offering falls apart.

5.2 The *e*^{3}*value* ontology

This section presents our eBusiness *ontology*, called *e*^{3}*-value* , which offers constructs for modelling eBusiness cases from an economical perspective. Cornerstone of this ontology is the notion of economic value, and how actors create, exchange, and consume objects of economic value.

The *e*^{3}*-value* is divided into three viewpoints, which each represent related statements on an eBusiness model:

- The **global actor** viewpoint shows:
    1. the *actors* involved in an eBusiness idea;
    2. the *objects of economic value* created, exchanged, and consumed by these actors;
    3. objects of value, which actors expect in return for an object of value delivered, also called the mechanism of *economic reciprocity*;
    4. objects which are offered or requested *in combination*;
    5. *phenomena*, such as consumer needs, that cause *exchanges* of objects between actors.
- The **detailed actor** viewpoint(s) shows:
    1. *partnerships* between actors, which show that actors request or offer objects of value jointly;
    2. *constellations* of actors, which need not to be seen on the global actor viewpoint, e.g. to avoid unnecessary complexity;
    3. plus: *expressions* as on the global actor viewpoint, but then only for actors expressed on the detailed viewpoint.
- The **value activity** viewpoint(s) shows:
    1. the value-creating or adding activities and their assignment to actors.

The main purpose of the *global actor* viewpoint is to explain the overall value model to all stakeholders, including CxO type of stakeholders, involved. It hides complexity, which can be shown on detailed actor viewpoints. The reason to introduce a *detailed actor* viewpoint can be twofold: (1) representation of constellations: a decomposition of a part of the global actor viewpoint to reduce complexity, and, (2) representation of partnerships: actors who decide to offer and/or request products or services as one virtual actor to/from other actors. The *value activity viewpoint(s)* shows what actors do to create profit or to increase value for themselves. Its main motivation is to separate discussions of who is participating in the eBusiness idea from who is doing what.

We illustrate the ontology by means of a project we have carried out in the free Internet service provisioning arena. The eBusiness idea underpinning this project is that users, in order to access the Internet, only have to pay a fee for a telephone connection, what they are used to do for other, paid, Internet access services also. In short, these telephone connection revenues are used to finance the entire operation. This eBusiness value model is shown in

Figure 2 (global actor viewpoint), Figure 7 (detailed actor viewpoint), and Figure 9 (value activity viewpoint).

5.2.1 The global actor viewpoint

The explanation of our ontology is structured by presenting a description for each concept, properties of the concept, relations with other concepts, and the way of visualization in a value model such as depicted in

Figure 2. A concept and relation is illustrated by one or more examples. Figure 1 presents the ontology graphically using UML class diagrams (OMG 1999, Rumbaugh 1991).

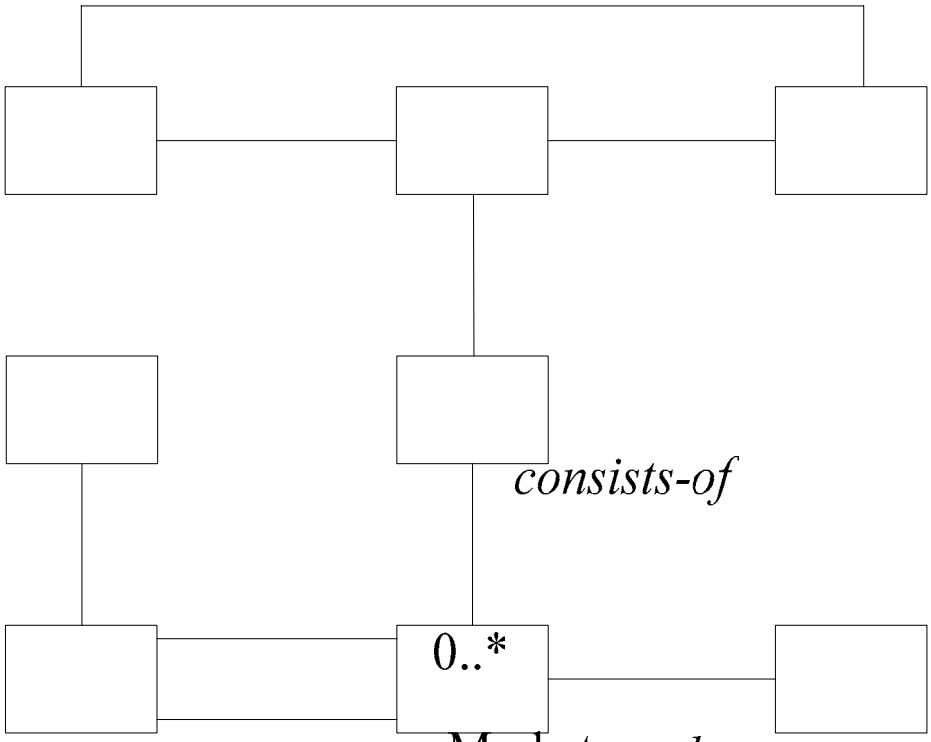

Figure 1: Concepts and relations of the *e*^{3}*-value* ontology[1] (global actor viewpoint).

[1] The notation is based on UML class diagrams. Rectangles are concepts, related by associations (lines). Concepts play a role in an association. Also, cardinality constraints are expressed.

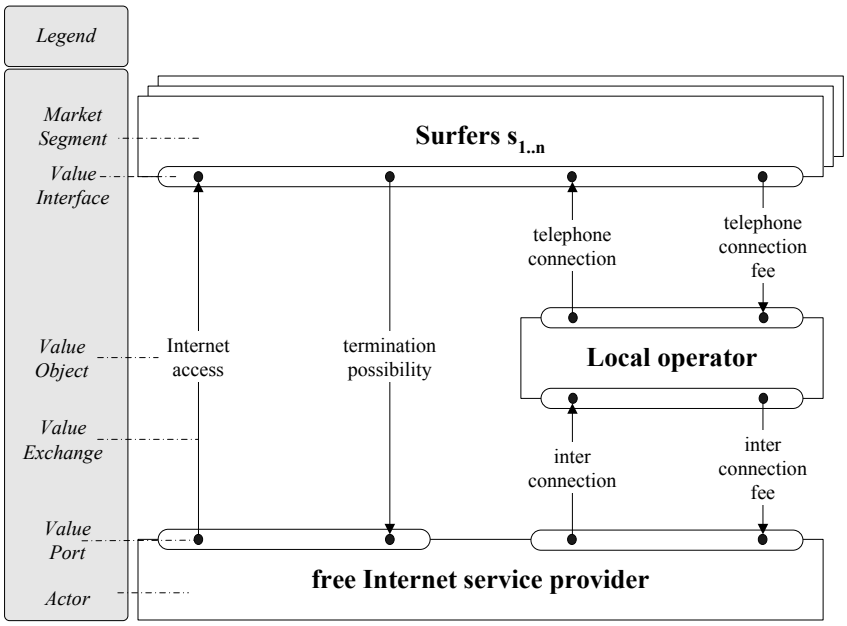

Figure 2: Value model for a free Internet access service: the global actor viewpoint.

Actor. An actor is perceived by his/her environment as an economically independent (and often also legal) entity. Enterprises and end-consumers are examples of actors. A profit and loss responsible business unit, which can be seen as economically independent is an actor, although such a unit needs not to be a legal entity.

Economically independent refers to the ability of an actor to be profitable after a reasonable period of time (in case of an enterprise), or to increase value for him/herself (in case of an end-consumer). For a sound and viable eBusiness idea, we require that each actor can be profitable or can increase his/her value. Nevertheless, we acknowledge that in the recent past, many eBusiness ideas were put in operation for which this was not the case. Such ideas are not sustainable and are consequently not in the scope of our research.

Properties. An actor has a name, e.g. a company name, or a name that represents the role such an actor plays.

Visualization. An actor is depicted by a rectangle, with his/her enterprise or role name.

Example. The global actor viewpoint (see

Figure 2) shows a *free Internet service provider* and a *local operator*. Also, *surfers* are presented as a market segment (to be discussed), which essentially is a set of actors valuing objects equally. The free Internet service provider is an actor who offers a service that the surfer is interested in: Internet access for free. The local operator exploits the local loop: the last mile of copper wire between a telephone switch and the home of a surfer. This loop is needed to set up a telephone connection between a surfer and the free Internet service provider. This telephone connection is used by the surfer's and provider's telecommunication equipment to access the Internet.

Value Object. Actors exchange value objects. A value object is a service, a good, money, or even an experience, which is of economic value for at least one of the actors involved in a value model. Actors may value an object differently and subjectively, according to their own valuation preferences (Holbrook 1999).

From a modelling point of view, we are interested in the *kind of* value objects which actors exchange, and not so much in the actual instances themselves. Therefore, when we speak about *value object*, we mean the kind of value object, or the prototype for all instances of a particular value object. In some cases, it is necessary to refer to the actual instances of objects of value exchanged by actors. We then call these objects *value object instances*.

Properties. A value object has a name. While choosing a name, one should keep in mind that it expresses the object from an economic value point of view.

Visualization. A value object is presented by showing the name of the object nearby a value exchange (to be discussed below), representing a potential trade of such an object, or by showing the name nearby value ports offering or requesting objects (see below).

Example. Many value objects in

Figure 2 speak for themselves. The value object *termination possibility* is however non-trivial. *Termination* in the world of telecommunication operators means that if someone tries to set up a telephone connection by dialling a telephone number, someone else must pick up the phone, that is, *terminate* the connection. If someone is willing to cause termination of a large quantity of telephone calls, most telecommunication operators are willing to pay such an actor for that. This is exactly what the free Internet service provider does: s/he aggregates a large number of *termination possibilities* from surfers and gets paid for that.

Also, the value object *interconnection* needs explanation. At the time the project was carried out there was in The Netherlands only one actor who operated the local loop, the last mile of copper wire between a telephone switch and the home of a surfer. From a surfer point of view, this local operator delivers an end-to-end telephone connection, in this case between the surfer and the free Internet service provider. However, the local operator does not operate a network that connects the surfer with the free Internet service provider directly. S/he only owns a part of that network. In such a case, the local operator must use an additional network, connected to the free Internet provider, which is owned by another operator to provide the surfer an end-to-end connection. In other words: the local operator must obtain *interconnection* from another Telco. In return for this, the local operator pays an interconnection fee.

Value Port. An actor uses a value port to provide or request value objects to or from his/her environment, consisting of other actors. Thus, a value port is used to interconnect actors so that they are able to exchange value objects. A value object flowing in or out allows an actor to denotes a change of ownership, or a change in rights. The concept of port is important, because it enables an abstraction away from the internal business processes, and to focus only on how external actors and other components of the eBusiness value model can be 'plugged in'. This is the value analogue of the separate external interfaces familiar from technical systems theory (Borst 1997). Take, for example, a bipolar in+out value multi-port, which is a characteristic combination occurring in eBusiness value models: an e-service port out and a money port in, or the other way around. Such a bipolar value port combination can be very well compared to an electrical wall outlet. As an external user, you don't want to be involved in what happens behind the wall outlet as long as it gives the right quality of service. The same approach holds for how external parties in an eBusiness value model view the value ports of a service-offering actor: the ports only define how the external connections to other actors should be made.

Properties. A value port has a *direction*, which can have the values *in* (shortly called an in-port) or *out* (called an out-port) indicating whether a value object flows in or out of an actor (seen from that actor).

Relations. A value port *offers* or *requests* one value object. This cardinality constraint again emphasizes that we are not so much interested in value object instances themselves, but rather in the prototype for such instances. A value object can be *requested by* or *offered by* zero or more value ports.

Visualization. The value port is depicted by a small black filled circle (see

Figure 2). Value *in*-ports have an incoming arrow. The name of the value object offered/requested by the port can be depicted.

Value Offering. A value offering models what an actor offers to (an out-going offering) or requests from (an in-going offering) his/her environment, and closely relates to the *value interface concept* (see below). A value interface models an offering of an actor to his/her environment, *and* the offering such an actor requests in return from his/her environment. In contrast, an offering is a set of equally directed value ports exchanging value objects, and implies that all ports in that offering should exchange value objects, or none at all.

A value offering is of use for representing a number of situations. First, some objects may only be of value for an actor if they are obtained in combination. In-ports exchanging such objects then form an in-going offering. Second, actors may decide to offer objects only in combination to their environment. Ports offering such objects then form an out-going offering. An example of an out-going offering is the case of *mixed bundling*. Mixed bundling refers to the mechanism that an actor wants to offer value objects in combination rather than separately, because that actor supposes that different products sold in combination yield more profit than that if they were sold separately (Choi 1997).

Relation. A value offering *consists of* one or more equally directed value ports. A value port is *in* exactly one offering.

Value Interface. Actors have one or more value interfaces. In its simplest form, a value interface

consists of one offering, but in most cases, a value interface groups one in-going and one out-going value offering. It shows then the mechanism of economic reciprocity. *Economic reciprocity* refers to rational acting actors. We suppose that actors are only willing to offer objects to someone else, if they receive adequate compensation (i.e. other value object(s) in an in-going offering) in return. So, with the value interface, we can model that an actor is willing to offer something of value to his/her environment but requests something in return, whereas a value offering models that objects can only requested or delivered in combination.

The exchange of value objects is atomic at the level of the value interface. Either all ports in a value interface (via value offerings) each precisely exchange one value object instance, or none at all. This ensures that if an actor offers something of value to someone else, s/he always gets in return what s/he wants. How this is ensured is a matter of a robust business process design, trust and associated control mechanisms (see e.g. (Tan, 2002), legal agreements, or sometimes use of technology, but this is not expressed by the value model.

Relations. A value interface is *assigned to* zero or one actor and *consists of* one or two value offerings, in the latter case being an out-going offering and an in-going offering. Each actor has its own value interface. Multiple value interfaces can be assigned to an actor and a value offering belongs to exactly one value interface.

Visualization. The value interface is visualized by a rounded box at the edge of an actor. Value ports are drawn in the interior of the rounded box. Note that a value offering is not visualized explicitly. However, value offerings can be easily seen by grouping all out-going value ports in a value interface (the out-going offering), or by grouping all in-going value ports in a value interfaces (the in-going offering).

Example. Consider in

Figure 2 the surfer. The in-going offering consists of telephone connection and Internet access. These objects are seen as one offering because they are only of value in combination for the surfer. An Internet connection is worthless without the telephone connection that is used for data transport. Also, for a surfer, the telephone connection is not of value without Internet access. The out-going offering contains the compensations for the obtained telephone connection and Internet access. These two offerings are grouped into a value interface to show that a surfer compensates its environment for obtaining a telephone connection and Internet access, with a fee and a termination possibility.

Value Exchange. A value exchange is used to connect two value ports with each other. It represents one or more potential trades of value objects between value ports. As such, it is a prototype for actual trades between actors. It shows which actors are willing to exchange value objects with each other.

Relations. The value ports involved in a value exchange are represented by the *has in* and *has out* relations, which relate to exactly one in-port and to exactly one out-port. A value port may *connect* to zero or more value exchanges.

Figure 3 exemplifies a situation with a port connected to more than one value exchange. Value ports of actor *a*, offering/requesting value objects *y* and *z*, connect via value exchanges to ports of actor *b*, but also connect to ports of actor *c*. This situation models that actor *a* and actor *b* are willing to exchange objects of value, and so do actor *a* and actor *c*. Note that the model does not represent the number of value exchanges over time, nor their ordering in time.

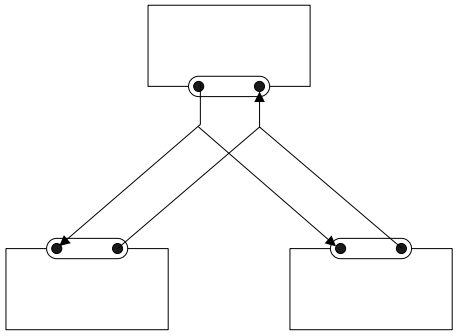

Figure 3: Actor a can decide to exchange value objects with actor b, or actor c.

Visualization. A value exchange is shown as line between value ports. The name of the value object which is exchanged, is presented nearby the value exchange.

Value Transaction. A value interface prescribes the value exchanges that should occur, seen from the perspective of an actor the value interface is connected to, because all ports in a value interface should exchange objects, or none at all. Sometimes, it is convenient to have a concept that aggregates all value exchanges, which define the value exchanges that must occur as consequence of how value exchanges are connected, via value interfaces to actors. We call this concept a value transaction. In its simplest form, a transaction is between two actors. However, a transaction can also be between more than two actors. We call such a transaction a *multi-party* transaction.

Figure 2 shows a multi-party transaction between a surfer, a local operator, and a free Internet service provider.

Relation. A value transaction *consists of* one or more value exchanges. Note that the exchanges in a transaction should be consistent with the way these exchanges are connected to value interfaces. A value interface requires that if a value object is exchanged via a port, also exchanges must occur via all its other ports. These exchanges must be also part of the transaction.

Figure 4 exemplifies why a value exchange can be in multiple transactions. In this example, actor *a* offers two value objects, and wants to have two value objects in return. There are two sets of actors who are a capable of participating in the exchange of values with actor *a*: actors {b_1,*c*}, and actors {b_2, *c*}. Clearly, actor *a* must exchange values with actor *c* (there is no alternative), but there is a choice between actor b_1 and actor b_2 for the other exchanges. Consequently, we can distinguish two transactions with overlapping value exchanges. Transaction 1 consists of the value exchanges e_1, e_2, e_3, and e_4 and transaction 2 consist of the value exchanges e_1, e_2, e_5, and e_6. Value exchanges, which are in more than one transaction, occur in multi-party transactions, of which

Figure 4 is an example.

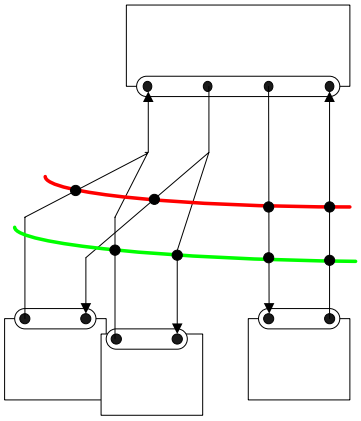

Figure 4: A value exchange can be in multiple transactions.

Visualization. A value offering is shown by a line intersecting the value exchanges it contains. The intersection points are shown by small filled circles.

Example.

Figure 2 shows a three-party offering between the free Internet service provider, a surfer, and a local operator. A surfer needs both to obtain Internet access, and to obtain a telephone connection, to be able to browse the Internet. From the surfer's value interface can be concluded that all four value exchanges connected to it are part of one transaction: either all ports of surfer's interface each exchange a value object or none at all.

Market Segment. In the marketing literature (Kotler 1998), a market segment is defined as a concept that breaks a market (consisting of actors) into segments that share common properties. We employ the notion of market segment to show that a number of actors assign economic value to objects equally. This construct is often used to model that there is a large group of end-consumers who value objects equally. We realize that in practice no actor will value objects exactly the same, but supposing an equal valuation for some actor groups is a simplification needed to arrive at comprehensible yet understandable value models.

In most cases, the individual actors of a market segment are left implicit. With *implicit* we mean that we do not model these actors individually. This is also the modelling purpose of the market segment construct: to have a shorthand for a large number of actors. However, actors are independent companies or individuals. As such, a specific actor, being part of a market segment, may exchange also other value objects than those mentioned in that market segment. Consequently, a market segment groups *value interfaces* of actors, exchanging objects that are valued equally, rather than that it groups actors themselves. If an actor, who is part of a market segment, has additional value interfaces, which other actors in that segment do not have, we model such an actor also *explicitly*.

Finally, value exchanges drawn to a segment can be seen as a shorthand notation for value exchanges to all actors in that segment. If we assume that market segment *b* (implicitly) consists of actors b_1, b_2, and b_3, *and* these actors value objects the same way, Figure 5 (b) is a shorthand notation for Figure 5 (a).

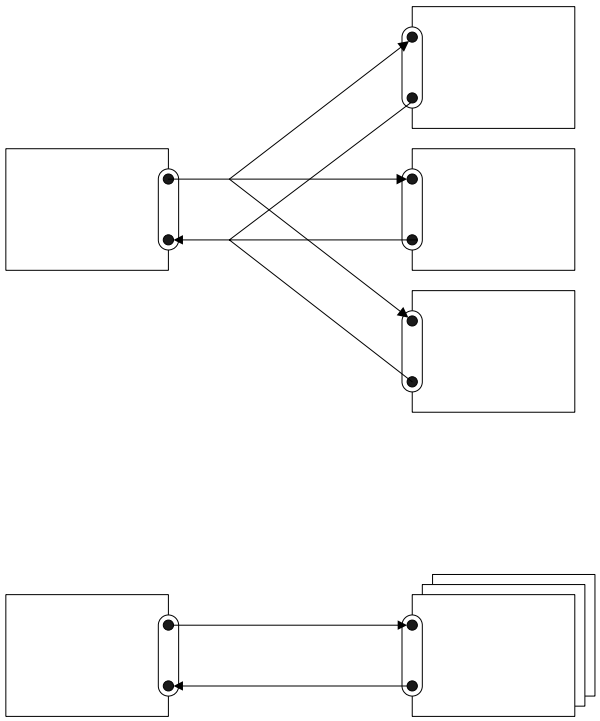

Figure 5: A value model without and with market segment.

Properties. A market segment is given a name, in must cases in plural form, such as customers, surfers, or alike. A market segment has a *count*, which indicates the number of actors in the segment. The count can be a number, unbound, or unknown.

Relations. Because a market segment is a set of actors, a value interface can be *assigned to* zero or one market segment, just as an interface can be assigned to an actor. Objects exchanged via this value interface are valued equally by actors in the segment.

An actor can be *in* a market segment. This relationship is needed to represent actors who have, besides value interfaces of a market segment, additional value interfaces of themselves. The additional interfaces are then related to the actor him/herself, while the relationship between actor and market segment is used to represent an actor's interfaces s/he has as a result of his/her membership in a market segment.

Visualization. A market segment is shown as three stacked actors. A value interface of a market segment is presented on one of the edges of the topmost actor. An explicitly modeled actor who is also part of a market segment is mentioned in the name of the market segment.

Example. The *surfers* segment (

Figure 2) consists of implicit actors who want to access the Internet.

Summary. In conclusion, the global actor viewpoint shows the top level actors in a value model, without discussing constellations and partnerships yet. Also, the assignment of value activities to actors is not shown by this viewpoint. The global actor viewpoint shows the objects of value exchanged between

actors. The market segment notion is useful if a large number of actors exists, who are supposed to assign economic value to value objects the same way.

The global actor viewpoint can be constructed in brainstorm sessions and workshops with all key actors. Also, this viewpoint can be used to present and explain the overall value model to stakeholders.

For the free Internet access service, the global actor viewpoint illustrates that the so-called free service is offered to surfers, but is not for free at all, since the surfer has to pay for a telephone connection. Also, this viewpoint shows that a local operator is needed to offer an Internet access service to surfers.

5.2.2 The detailed actor viewpoint

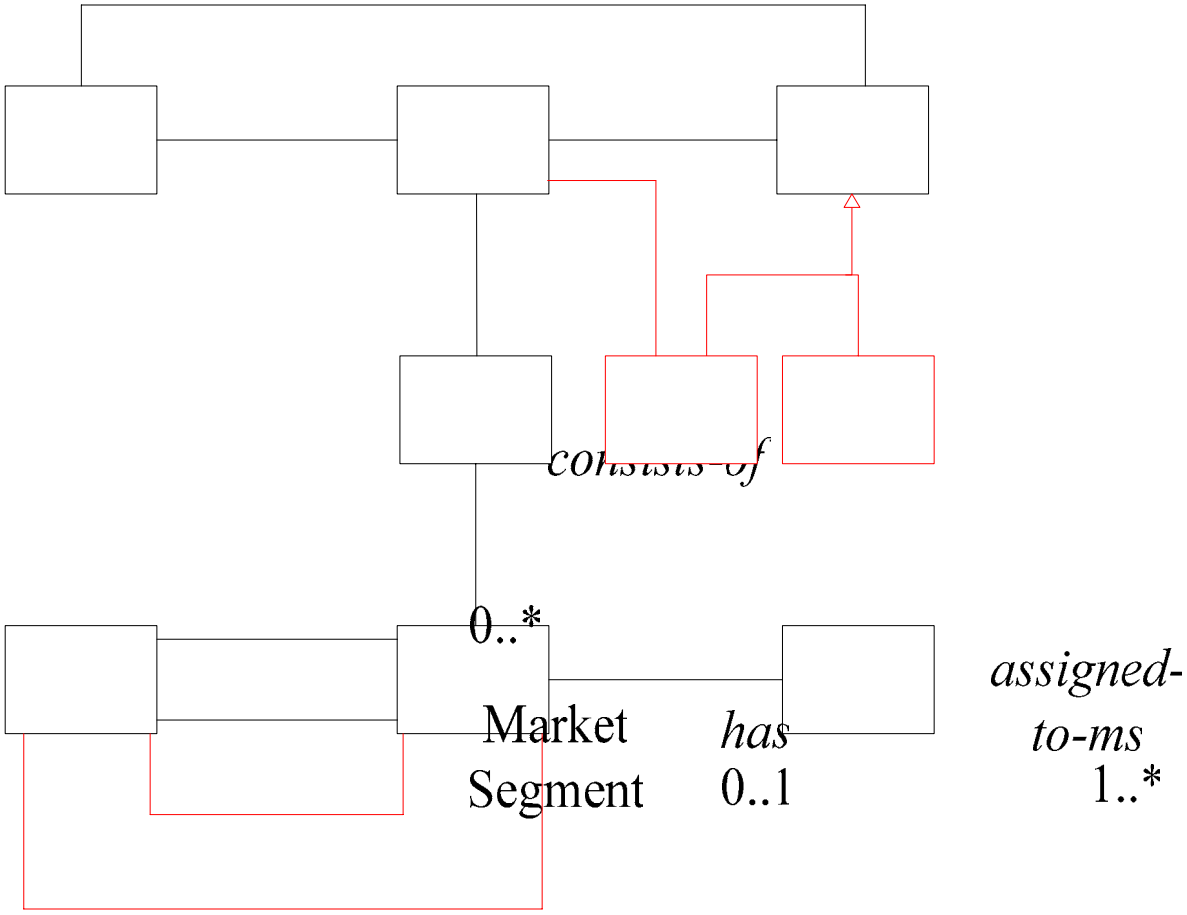

Figure 6: Concepts and relations of the e^3-*value* ontology extended for the detailed actor viewpoint. A composite actor and an elementary actor are generalized into an actor.

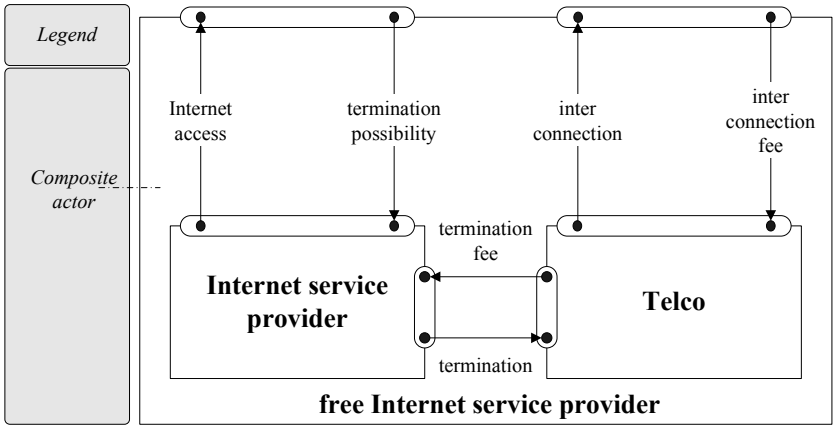

Figure 7: Value model for the free Internet case: the detailed free Internet service provider actor

view.

The purpose of a *detailed actor viewpoint* (see Figure 7) is twofold. First, a detailed actor viewpoint can be used to *detail* an actor identified on the global actor viewpoint into more actors. We call such an actor a *value constellation*. A value constellation can be used to isolate parts of the value model to a limited number of actors, who can decide on that specific part without consulting other actors participating in the eBusiness idea too much. A value constellation is also a way to reduce complexity on the global actor viewpoint, such that all actors can understand this viewpoint. A second reason to introduce a detailed viewpoint is the representation of *partnerships* between actors. As such, a number of actors may decide to present themselves, as a virtual enterprise actor, to their environment (see e.g. (Davidow 1992)). These actors then decide on one common value interface to their environment.

Composite actor and elementary actor. For both earlier mentioned modelling purposes, we specialize the actor concept into a composite actor, and an elementary actor (see Figure 6).

A *composite actor* groups value interfaces of other actors. Also, a composite actor has its own value interfaces to its environment. These composite actor's value interfaces allow us to (1) abstract away from the composite's internals, or (2) to show a common value interface from actors who decide to present themselves as a virtual enterprise.

An *elementary actor* does not contain value interfaces of other actors. Such an actor is the lowest decomposition level that can be reached from an actor perspective.

Note we group *value interfaces* and not *actors* into a composite actor. The reason for this is that in case of partnerships, an actor may decide to offer objects jointly with objects of other actors, but also may decide to offer other objects on its own. Consequently, it is not the actor that is grouped, but what s/he is offering for a specific case. The same holds for introducing a composite actor in case of value constellations. Such an actor can group a number of value interfaces of the actors it contains, while interfaces of these actors may also appear somewhere else in the value model.

Relations. A composite actor *is an* actor. An elementary actor *is* also *an* actor. This means that all properties and relations identified for actors, will also hold for composite and elementary actors. A composite actor *consists of* minimal two value interfaces of other actors. We need at least two interfaces to be able to group meaningfully.

Visualization. A composite actor is visualized by drawing a rectangle around the actors whose value interfaces are grouped. Inside this rectangle, the value interfaces of the actors must be shown, which are grouped by the composite actor.

Example. The free Internet service provider appears to be a value constellation, which consists of two other actors: (1) an *Internet service provider* offering Internet access (e.g. by exploiting access servers), and (2) a specific *Telco* handling interconnection of telephone calls between the Internet service provider and the local operator.

The detailed actor viewpoint shows also exchanges of value objects between the Internet service provider and *Telco*. The provider terminates connections by exploiting an Internet access server (effectively a large modem-bank), which answers telephone calls made by the modems of surfers. Termination of large quantities of telephone calls is of value for *Telco*. Consequently, *Telco* pays the Internet service provider a termination fee.

Value exchange revisited. We have introduced the value exchange concept earlier to relate ports of actors exchanging objects. These connected ports have *opposite* directions. The value exchange construct is also used to relate value ports of a composite actor to value ports of actors being part of the composite. In this case, connected ports have *equal* directions. An object offered via an out-port of a composite actor still has to be offered via an out-port of one of the actors in the composite. Also an object requested via a composite actor's in-port must be requested by an in-port of one of the actors it contains.

Properties. To represent the various applications of value exchanges, we distinguish four types (see Table 1). A type 1 exchange relates ports of actors trading objects, while a type 2 exchange relates ports of a composite actor with ports of the actors it contains. Other types are discussed in the remainder of this chapter.

Table 1: Various value exchange types.

Value exchange	*Relates port 1 of*	*With port 2 of an*	*Ports have*

type	*an*		*direction*
1	Actor	Actor	Opposite
2	Composite actor	Actor	Equal
3	Elementary actor	Value Activity	Equal
4	Value Activity	Value Activity	Opposite

Relations. To stress that a type 2 value exchange, which connects ports with equal directions is different from a type 1 value interface which connects ports with opposite directions, other associations are shown in the ontology. A value exchange *has a first* value port of the composite actor, and *has a second* value port of one the actors contained by the composite actor.

Example. Figure 7 exemplifies a type 2 value exchange. The ports of the composite actor free Internet service provider are mapped on ports of value interfaces of the Internet service provider and *Telco*.

Summary. The detailed actor viewpoint intends to represent actors jointly offering or requesting a product or service to their environment, also called a partnership. Moreover, the viewpoint is used to detail specific parts of an eBusiness value model, which are abstracted away on the global actor viewpoint (the value constellation). Strictly spoken, a composite actor groups value interfaces of other actors, not the actors themselves.

5.2.3 The value activity viewpoint

The main purpose of the *value activity* viewpoint is to illustrate the assignment of value activities to actors. Figure 9 shows this viewpoint for parts of the free Internet service provider. How value activities are assigned to the various possible actors is a free variable that, as a result of the extended enterprise network setting, leads to many design options and choices in eBusiness value models. Hence, this assignment is a key consideration in strategic eBusiness decision making.

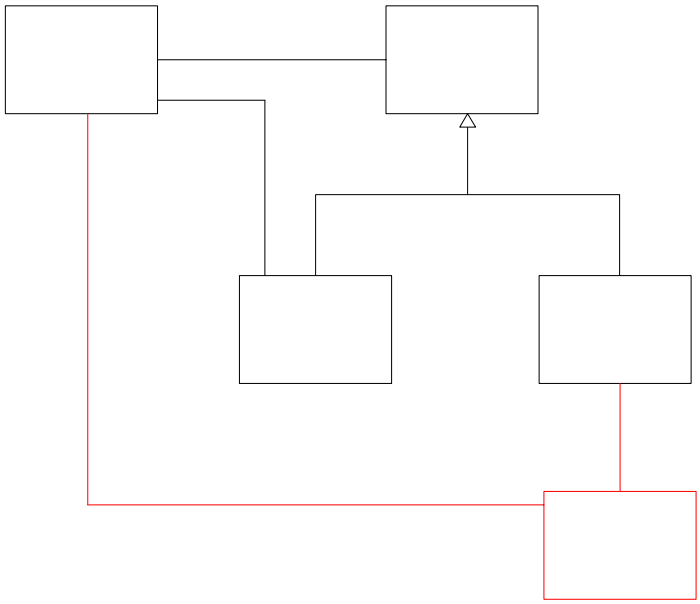

Figure 8: Concepts and relations of the *e*^{3}*-value* ontology extended for the value activity viewpoint.

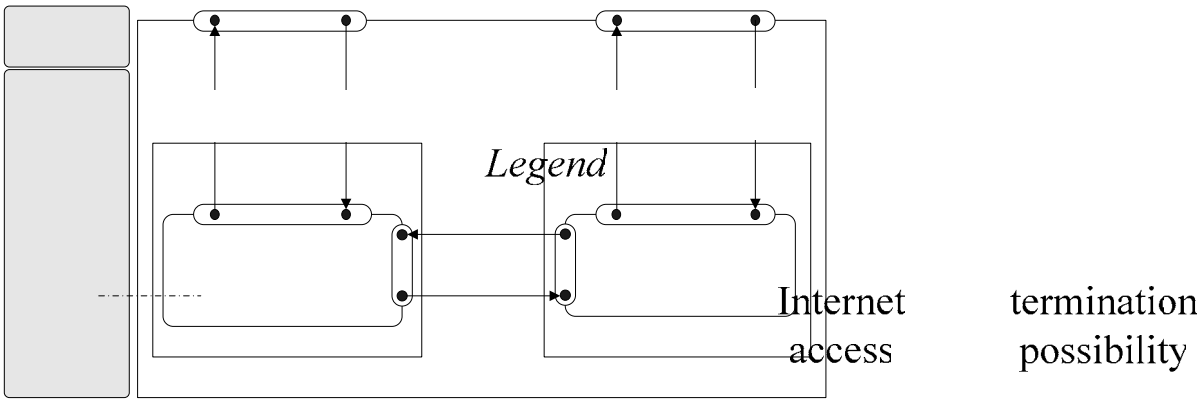

Figure 9: Value model for the free Internet case: the value activity view.

Value Activity. An important issue in value model design is the *assignment* of value activities to actors. Therefore, we are interested in the collection of operational activities which can be assigned as a whole to actors. Such a collection we call a value activity. Actors perform value activities, and to do so, a value activity must yield profit or should increase economic value for the performing actor. Consequently, we only distinguish value activities if at least one actor, but hopefully more, believes that s/he can execute the activity profitable. Value activities can be decomposed into smaller activities, but the same requirement stays: the activity should yield profit. This also gives a decomposition stop rule.

Relations. A value activity *has* one or more value interfaces, just like actors and market segments. A value interface belongs to exactly zero or one value activity. A value activity is *performed by* precisely one elementary actor. Finally, multiple value activities can be *performed by* an actor.

Visualization. A value activity is graphically presented by a rounded box, which is drawn inside the actor who performs the activity.

To draw readable diagrams, we sometimes omit value interfaces, ports and exchanges. In Figure 9, the Internet service provider shows no value interfaces anymore, while Figure 7 shows for the same actor two value interfaces. If a value interface of an actor has the same structure as a value interface of a value activity s/he performs, we may decide not to present the value interface of the actor. Two value interfaces have the same structure if each port of the first value interface can be matched with precisely one port of the second value interface, and vica versa. Matching of two ports is possible if both ports have the same direction and if they exchange the same value object. However, an omitted value interface conceptually exists, and also value exchanges to connect an actor's value interface to a value interface of his/her value activity conceptually exist. The same holds for composite actors: we may decide to omit value interfaces of a composite actor if they have the same structure as the value interfaces of actors the composite actor exists of.

Example. The Internet service provider performs an Internet access provisioning activity. This activity comprises investment in and maintenance of Internet access servers. Another activity, which might be thought of is e.g. a web hosting service. *Telco* executes an activity named call delivering. This activity is the exploitation of a physical network between the local operator and the Internet service provider for data transport. For all these activities, we assume that they are, after some period, profitable for the actors performing these activities.

Value exchange revisited. We also use the value exchange to connect ports of value activities with ports of the actor performing these activities. These are called type 3 value exchanges. Such ports must have the same direction. Also, ports of value activities, which are performed by the same actor can be connected by using type 4 value exchanges. These exchanges represent 'internal' trades of an actor. Such exchanges connect ports with an opposite direction.

Summary. The value activity viewpoint represents the assignment of value activities to actors. By assuming that a value activity is commercially interesting to be performed by at least one actor, but preferably more actors, we can shift activities from one actor to another actor, thereby discussing who is doing what. Especially if roles of actors are not clear, which is often the case for innovative eBusiness projects, negotiating the assignment of activities to actors is an important part of the exploration track.

5.3 The e^3-*value* ontology and operational scenarios

Operational scenarios are used to capture parts of the eBusiness idea and to contribute to a common understanding between stakeholders. Moreover, we use operational scenarios to evaluate an eBusiness model. In this section, we focus on a scenarios role to capture parts of an eBusiness value model, and more specifically we show how scenarios are used to specify by what phenomena exchanges of objects are caused. To represent operational scenarios, we utilize Use Case Maps, a generic lightweight scenario representation mechanism. The following sections discuss UCMs, and bind UCMs to our *e*^{3}*-value* ontology.

5.3.1 Use Case Maps

A UCM is a visual notation to be used by humans to understand the behaviour of a system at a high level of abstraction (Buhr 1998). It is a scenario-based approach intended to explicate cause-effect relationships by travelling over paths through a system.

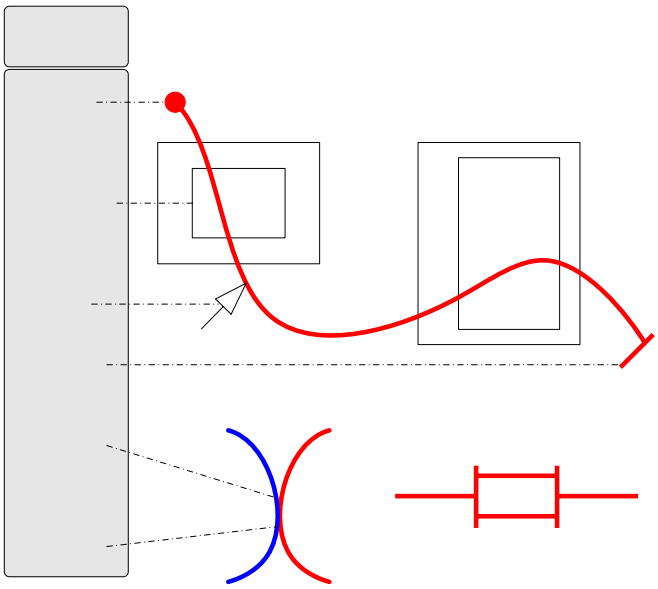

Figure 10: UCM constructs.

The basic UCM notation is very simple, and consists of three basic elements: responsibilities, paths and components. The term component should be interpreted in a broad 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 10. 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 (bullet at the top). Next, the pointer moves along the indicated scenario 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 in the UCM itself since the concept of state is not part of a UCM; typically, this effect is described in natural language. Finally, the end position is reached (stroke perpendicular to the scenario path) and the pointer is removed from the diagram.

In the same Figure 10, two frequently used UCM constructs are shown. The AND construct is used to spawn (AND-fork) and synchronize (AND-join) multiple parallel scenario paths. The OR construct is a means to express that a scenario path continuous in alternative directions.

To be meaningful, the UCM notation must be bound to some other notation, in our case the *e*^{3}*-value* ontology. More specifically, we have to articulate the components UCM scenario paths can touch using responsibility points. Therefore, we present UCM's the same way as we did for our *e*^{3}*-value* ontology, and relate scenario paths to *e*^{3}*-value* ontology constructs.

5.3.2 An ontology for Use Case Maps

A UML-model for the representation of Use Case Maps is shown in Figure 11. It is based on a UCM UML model by (Amyot 2000). Below we discuss the various UCM constructs, and exemplify their use in the free Internet access project. Value viewpoints enriched with Use Case Maps are shown in Figure 12 (the

global actor viewpoint), and Figure 13 (a detailed actor viewpoint).

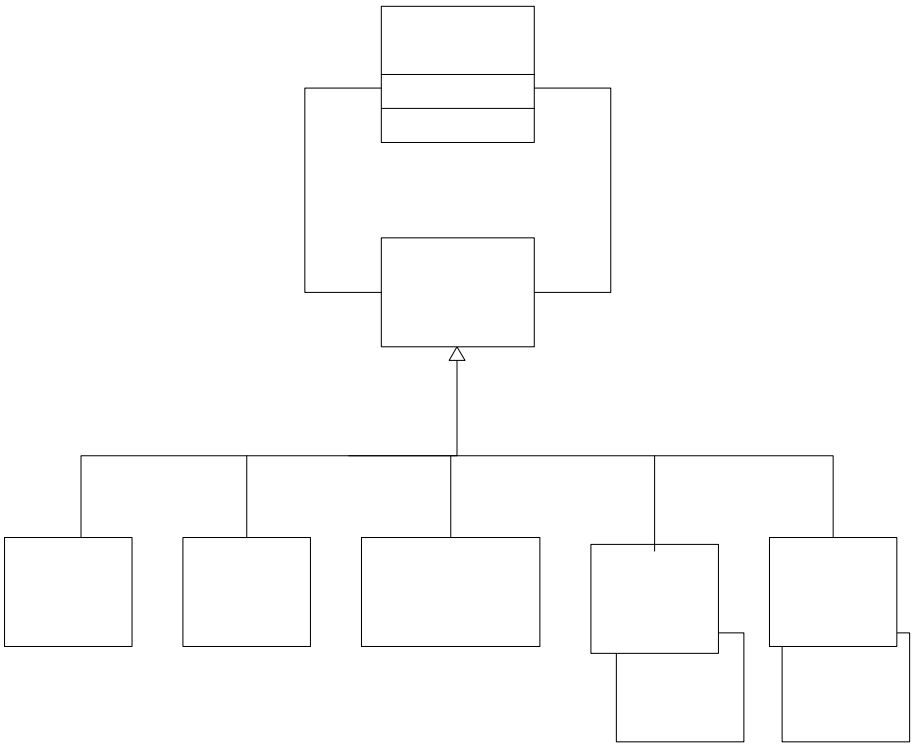

Figure 11: An ontology of Use Case Maps based on Amyot and Mussbacher.

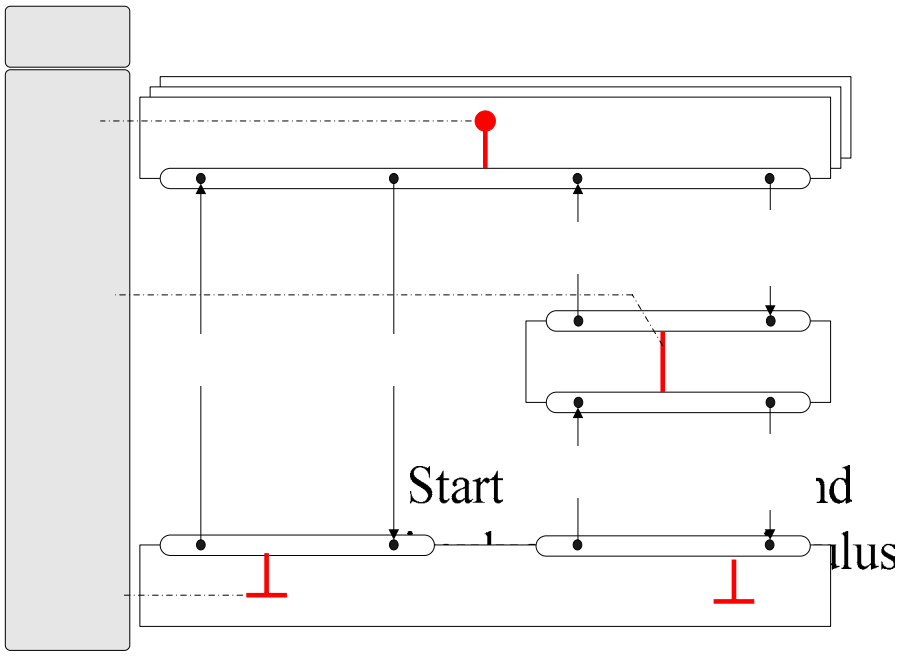

Figure 12: Use Case Maps applied to the global actor viewpoint.

Legend

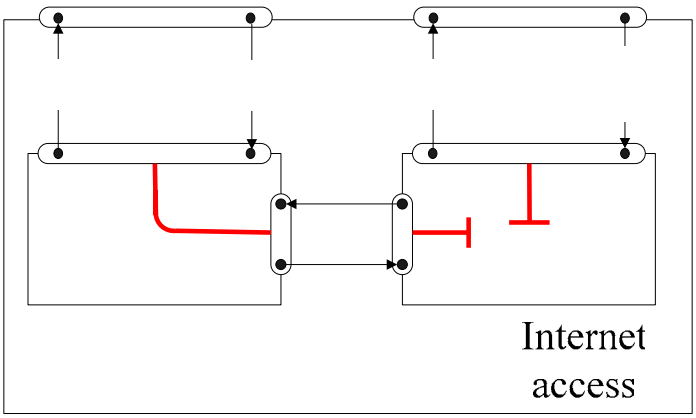

Figure 13: Use Case Maps applied to the detailed actor viewpoint.

We utilize a simple form of Buhr's Use Case Maps. The two constructs used are *dependency elements* and *connection elements*.

Dependency element. A scenario is expressed by dependency elements interconnected by connection elements (see below). Essentially, a scenario gives dependencies between value interfaces (a kind of connection element) so that we can reason for an entire model what happens with other value interfaces if we exchange values via one particular value interface.

Properties. Each dependency element can have a textual label for naming purposes.

Relations. A dependency element has one upper connection element and one lower connection element.

Visualization. Dependency elements are drawn using normal lines.

Connection element. A connection element connect various dependency elements. Dependency elements can be start and stop stimuli, AND/OR forks or joins and value interfaces (see below).

Properties. Each connection element can have a textual label for naming purposes.

Relations. A connection element has zero or more up-dependency elements. The same holds for down-dependency element.

Visualization. Connections elements are visualized, depending on their specific kind (see below).

Stimulus element. Scenarios start with one or more *start stimuli*. A start stimulus represents an event, possibly caused by an actor. If an actor causes an event, the start stimulus is drawn within the box representing the actor. A scenario also has one or more *end stimuli*. They have no successors.

Visualization. A start stimulus is visualized by a filled circle; an end-stimulus is presented by a line, placed in an angle of ninety degrees on the line visualizing a dependency element. If an actor causes a stimulus, it is drawn in the interior of such an actor.

Example. The need for an actor to surf on the Internet is an example of a start stimulus. Such a stimulus results in a number of value exchanges between the actors participating in the value model.

AND and OR continuation elements. An *AND fork* connects a dependency element to one or more dependency elements, while the *AND join* connects one or more dependency elements to one other dependency element. It splits a scenario into more sub scenarios or merges sub scenarios into one scenario (see for a path the discussion below). An *OR fork* models a continuation of the scenario into one direction, to be chosen from a number of alternatives. The *OR join* merges two or sub scenarios into one scenario.

Visualization. An *AND* fork/join is shown as a line, placed in an angle of ninety degrees between lines visualizing dependency elements. An *OR* fork/join is presented by a number of lines joining into one (a join), or by a line splitting into more lines (a fork).

Value interface. Another way to connect dependency elements is to use a value interface. We use value interfaces (connected by dependency elements) to create profitability sheets on a per actor basis to assess profitability (see the following section). Such a sheet shows when objects of value are leaving or

entering an actor as a result of scenario path execution.

5.4 Profitability sheets

If we ask enterprises and end-consumers to assign economic value to the value objects they obtain and offer, and we know the number of start stimuli per time frame (say a month), we can assess potential profitability for each actor involved. We capture profitability numbers in profitability sheets on a per actor bases. Table 2 shows the structure of such a sheet.

Table 2: Structure of a profitability sheet.

Actor	*actor name*	
Scenario path	path name	
#occurences/timeframe	#occurrences	
	Value Object In	*Value Object Out*
	Euro x_1	Euro y_1
□	□	

To create profitability sheets for actors, we utilize our UCM scenario paths. These paths put into operation a scenario, and show which value objects are exchanged by actors via their value interfaces, as a result of the occurrence of one or more start-stimuli. Profitability sheets are constructed by following each scenario path. By doing so, we find the objects of value each actor exchanges as a result of executing the path. Each time we cross a value interface, we add the object(s) flowing out the interface of that actor to the actor's profitability sheet in the column *value object out*, while the objects flowing into an actor are added to the actor's profitability sheet in the column *value object in*.

For enterprises, profitability analysis boils down to a net cash flow analysis: for enterprises we only take into account value objects denoting money. This is conform traditional investment analysis (Horngren 1987), that takes the net present value of all ingoing and outgoing money streams (including an initial investment). Given a certain discount rate, this net present value should be at least a positive number for each enterprise involved.

On the other hand, for end-consumers, profitability analysis is far more difficult. End consumers are interested to increase their economic utility, or to do a best-for-money deal. The problem here is that we have to assign economic value (in terms of a monetary unit) to non-money objects such as Internet access. According to axiology literature (Holbrook 1999) valuation by end-consumers is subjective and may include many factors. Nevertheless, we have found it very useful to discuss with stakeholders at least factors which might influence valuation of objects by end-consumers. This leads to a better understanding of enterprises why an end-consumer wants to buy a value object in the first place.

Finally, focussing on profitability numbers themselves is not of particular interest since there are so many assumptions they are based upon. We have learned that doing sensitivity analysis is far more important. With sensitivity analysis, we vary factors in the model we are uncertain about, for instance the number of start stimuli, the way actors assign economic value to objects, or the structure of the model itself. More information on doing sensitivity analysis on *e*^{3}*value* models can be found in (Gordijn 2003).

5.5 Related enterprise ontologies

A number of other enterprise ontologies exist. Here we discuss the AIAO enterprise ontology, the TOVE

ontology, the REA ontology and Osterwalder's ontology.

5.5.1 AIAI enterprise ontology

The AIAI enterprise ontology (Uschold 1998) defines a collection of terms and definitions relevant to business enterprises. Two enterprise ontology concepts relate to our ontology but have a different interpretation: (1) *activity* and (2) *sale*. In the enterprise ontology, *activity* is the notion of actually doing something, the how. Our related definition, *value activity*, abstracts from the internal process and in contrast stresses the externally visible outcome in terms of created value, independent from the nature of the operational process. Thus, the defining boundary of what an activity is differs: in the *e*^{3}*-value* ontology the decomposition stop rule is to look at economically independent activities; business process or workflow activities have different decomposition rules, as such activities need not be economically independent. The enterprise ontology further defines a *sale* as an agreement between two legal entities to exchange one good for another good. In our ontology, the concept of sale roughly corresponds to the concept of *transaction*, with the important difference that a sale is an actual agreement, while a transaction is only a potential one. A transaction contains *value exchanges*. In the enterprise ontology, only two goods are exchanged in a sale. In contrast, in our ontology a transaction contains an arbitrary number of value exchanges. This is needed to model a *bundle* of goods that is offered or requested as a whole. Furthermore, our ontology is capable of multi-party transactions. The project in this chapter illustrates the need for such a concept.

5.5.2 TOronto Virtual Enterprise ontology

The TOVE ontology (Fox 1998) identifies concepts for the design of an agile enterprise. An agile company integrates his/her structure, behaviour and information. The TOVE ontology currently spans knowledge of activity, time and causality, resources, cost, quality, organization structure, product and agility. However, the interfaces an enterprise has to its environment are lacking in TOVE. Generally, the notion of the creation, distribution, and consumption of value in a stakeholder network is not present in the TOVE ontology. Hence, the TOVE ontology concentrates on the internal workflow of a company, whereas our ontology captures the outside value exchange network.

5.5.3 Resource Event Agent ontology

The Resource Event Agent (REA) ontology (Geerts 1999) shows from an ontological perspective many similarities with the *e*^{3}*-value* ontology. REA calls *actors agents*. Agents are offering or requesting *resources* (in *e*^{3}*-value* called *value objects*) by economic *events*. The latter can be compared to *value ports* in *e*^{3}*-value*. REA relates economic events of different actors by *exchanges* which correspond to *e*^{3}*-value value exchanges*. Finally, economic events of an agent are related by a *duality* relation. This models economic reciprocity which is handled by *e*^{3}*-value* by the notion of *value interface*.

From an ontological perspective, *e*^{3}*-value* and REA differ with respect to the notion of *value activity*. This concept lacks in REA, but is important for eBusiness idea exploration. A value activity is a potential profitable activity for one or more actors. Because eBusiness development tracks are characterized by shifts in actors performing these activities, it is important to model value activities explicitly.

From a methodological point of view, REA is not an approach for business development, whereas *e*^{3}*-value* provides a methodology for doing so, e.g. by value model construction and reconstruction, and by profitability-based sensitivity analysis.

5.5.4 Osterwalder and Pigneur ontology

Osterwalder and Pigneur (2003) propose an ontology for business models consisting of four pillars: (1) product innovation, (2) customer relationship, (3) infrastructure management, and (4) financial aspects (see also Chapter Four). The product innovation pillar covers aspects of value that the firm offers to its customers. The customer relationship aspect is about the definition of target customers, the channels used to reach and communicate with customers, and the kind of relationship a firm wants to establish with a customer. Infrastructure management focuses on the capabilities, value configurations, and partnerships necessary in order to create value and reach the customer. Finally, their ontology includes facilities for representing financial aspects of a business model. Ontologically, this ontology is rather comprehensive, but not sufficiently lightweight. The latter is for instance important for having a tractable instrument in

workshops.

From a methodological viewpoint, the ontology currently lacks a convenient way for visualizing business models, which is important for using the ontology in a practical way. Additionally, the ontology seems not so much intended for designing business models themselves, but is more biased towards ontologically stating what a business model actually is.

5.6 Conclusion and further research

In this chapter, we have presented the *e*^{3}*value* ontology for eBusiness models. Our ontology provides a rigorous approach for specifying and analyzing eBusiness models. The intent of our modelling methodology is twofold.

First, the methodology should contribute to a better understanding of a particular eBusiness model. Since most eBusiness models involve many enterprises, the risk of not having a shared understanding is high. Moreover, even for a single enterprise, participation of a wide range of stakeholders is required, who may not speak the language of another participant. We address this issue by providing a rigorous framework of concepts and relations between these, which together can be used to conceptualize an eBusiness idea. In addition, we have for most concepts in the ontology a visual presentation. By using this visualisation, we can present the heartbeat of an eBusiness model graphically.

In addition to the creation of a shared understanding of the eBusiness idea there is a second reason to use the *e*^{3}*value* ontology. Once an eBusiness model has been conceptualized and visualized, it can be analyzed (e.g. using ontological rules that state what a value eBusiness model is), and it can be assessed for potential profitability. This makes the ontology more than a conceptualizing instrument.

There are plenty of opportunities to do more research along the lines of the *e*^{3}*value* ontology. Below, we mention a few of such lines. For example, models made by using the *e*^{3}*value* ontology focus on economic value only, whereas most eBusiness idea require a (re) implementation of inter organizational business processes and cooperating information systems. How can we use business models to construct such processes and information systems? Another research track is in the field of trust. Business models assume that value exchanges via interfaces are always atomic. In other words: if A pays B for a good, B will deliver the good to A. Needless to say that this is not always the case in practice. So, how can we extend the ontology with constructs to guarantee this atomicity of value exchanges, for instance by trust means? Finally, *e*^{3}*value* ontological concepts themselves need to be worked out. For instance, what kinds of value objects exist? Is there a principal difference between services and goods in the context of business models? What different forms of actor (de) compositions are convenient for modelling? We now have partnerships and value constellations, but perhaps other ones are applicable?

In sum, ontologically founded business value modelling is a starting, yet promising field of research. A particular strength of business value modelling is the capability to overcome the bridge between verbally articulated strategic intents of networked enterprises, and the way of putting these intents into operation by means of inter-organizational business processes and information systems, in a concise yet thorough way.

References

Amyot, D and Mussbacher, G (2000) On the extension of UML with use case maps concepts. In A. Evans, S. Kent, and B. Selic, (eds) UML 2000 - The Unified Modeling Language. Advancing the Standard., volume 1939 of LNCS, pages 16–31, Berlin, D, 2000. Springer Verlag.

Borst, W. N., Akkermans, J.M (1997). Top. Engineering ontologies. International Journal of Human-Computer Studies, 46: 365–406.

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

Cassidy, J. (2002) Dot.con. Perennial: Harper Collins.

Choi, S.J., Stahl, D.O., and Whinston, A.B (1997). The Economics of Doing Business in the Electronic Marketplace. MACMillan Technical Publishing, Indianapolis, IN.

Davidow, W.H., and Malone, M.S (1992). The Virtual Corporation - Structuring and Revitalizing the Corporation for the 21st Century. HarperCollings, New York, NY.

Fox, M.S., and Gruninger, M (1998). Enterprise modelling. AI Magazine, Fall, 19 (3): 109–121.

Geerts, G., and McCarthy, W.E (1999). An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. IEEE Intelligent Systems and Their Applications, July-August, pp 89-94.

Gordijn, J., and Akkermans, J.M. (2003). Value based requirements engineering: Exploring innovative e-commerce idea, Requirements Engineering Journal, in print, Web version: http://link.springer.de/link/service/journals/00766/contents/03/00169.

Gordijn, J (2002). Value-based Requirements Engineering: Exploring Innovative e-Commerce Ideas. PhD thesis, Free University Amsterdam, NL. Also available via http://www.cs.vu.nl/~gordijn/thesis.htm

Holbrook, M.B. (1999). Consumer Value: A Framework for Analysis and Research. Routledge, New York, NY.

Horngren, C.T., and Foster, G (1987). Cost Accounting: A Managerial Emphasis, sixth edition, Prentice-Hall, Englewood Cliffs, NJ.

Kotler, P (1988). Marketing Management: Analysis, Planning, Implementation and Control. Prentice Hall, Englewood Cliffs, NJ.

Loucopoulos, P and Karakostas, V (1995). System Requirements Engineering, McGraw-Hill, Berkshire, UK.

Normann, R and Ramírez, R (1993). From value chain to value constellation: Designing interactive strategy. Harvard Business Review, (july-august): 65–77.

Normann, R and Ramírez, R (1994). Designing Interactive Strategy - From Value Chain to Value Constellation. John Wiley & Sons Inc., Chichester, UK.

OMG Unified Modeling Language Specification, Version 1.3. 1999.

Osterwalder, A., and Pigneur, Y (2003) Towards Strategy and Information Systems Alignment through a Business Model Ontology, to appear in the proceedings of the 23rd Annual International Conference of the Strategic Management Society (SMS) Baltimore.

Porter, M.E (2001) Strategy and the Internet. Harvard Business Review, (march): 63–78.

Porter, M.E. and Millar, V.E (1985). How information gives you competitive advantage. Harvard Business Review, (July-August): pp.149–160.

Rumbaugh, J., Jacobson, I., and Booch, G (1999). The Unified Modelling Language Reference Manual. Addison Wesley Longmann, Inc., Reading, MA.

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

Tapscott, D., Ticoll, D., and Lowy, A (2000). Digital Capital - Harnessing the Power of Business, Nicholas Brealy Publishing, London, UK.

Tan, Y.H, (2002). Formal aspects of a generic model of trust for electronic commerce. Journal of Decision Support Systems and Electronic Commerce, (Forthcoming).

Uschold, M., King, M., Moralee, S., and Zorgios, Y (1998). The enterprise ontology. The [Knowledge Engineering Review, 13(1): 31–89.