Considering software quality requirements as networked business quality requirements

Vincent Pijpers & Jaap Gordijn

Free University, FEW/Business Informatics, De Boelelaan 1083a, 1081 HV Amsterdam, The Netherlands. (v.pijpers, gordijn)@few.vu.nl.

Abstract. Over the years, research on software architecture has identified architectural styles. In this paper we consider five of such styles, and analyze whether these styles can also be found in business value models for networked enterprises. To this end, we analyze 17 e^3value models, as discussed in 29 academic papers or book chapters. We also consider whether the quality attributes related to the software architecture styles, are also relevant for e^3value models. To demonstrate the relevance of our findings we present a case study.

1 Introduction

Today, Requirements Engineering (RE) increasingly treats *context* of information systems as a first-class-citizen to rationalize Information System (IS) requirements, and moreover, to properly align and trace IS requirements with business requirements. Examples of context-aware RE approaches include goal modeling (eg. i^* [33]), problem framing [15], value modeling [9,23], and strategic business modeling [27].

Understanding of the context of an information system often comprises understanding of *enterprises*. These enterprises are not only the enterprises *using* the information system, but also related enterprises, forming a network to satisfy complex consumer needs. Such networks, today largely enabled by IT, are referred to as *networked value constellations* [30], and are an important part of an enterprise's context.

In this paper, we focus on a single requirement engineering technique for analyzing context of multi-enterprise information systems: namely e^3value . The e^3value approach is used to explore the business context of networked value constellations with the aim to create a shared understanding of the business context, to do a financial analysis for early feasibility assessment, and to scope the requirements engineering process for information systems themselves, supporting the networked value constellation at hand. To this end, the e^3value approach has been used in combination with other requirements engineering techniques (eg. i^* [13], UML deployment diagrams [7,26] and UML activity diagrams [25]).

In a way, an e^3value model can be seen as a very early "*functional*" requirement expression. An e^3value model states the objects/services of *economic value*, enabled by IT, that are transferred between enterprises, and *what* these enterprises request in return for doing so. However, in general, it has been widely

recognized that *non-functional*, or quality, requirements are an important class of requirements too. These can not be really identified with e^3value yet.

There is however a significant body of knowledge on how to express quality requirements for information systems themselves. Examples of quality requirements in IS are requirements about reusability, modifiability, integratability, portability, and scalability. In this paper, we analyze, by revisiting 17 published e^3value case studies, whether the aforementioned IS quality requirements can also be meaningfully identified in e^3value models. If we can identify such requirements, this can be considered as the first step to arrive at a more comprehensive theory on quality requirements, not only considering information systems, but also spanning *business value* considerations. This is the main contribution of this paper.

To identify quality requirements, we use five known architectural styles found by [3] in *information system architectures*. If we find sufficient support for the existence of these architectural styles in the e^3value models and the related quality requirements, we then can use styles in e^3value models to identify relevant quality requirements. We illustrate this with a case study in which we (1) show why quality requirements are relevant to identify and (2) how these quality requirements can be found by exploiting architectural styles.

This paper is constructed as follows: first we discuss the e^3value approach. Then, we introduce our view on notions such as architectures, architecture styles and quality requirements. Hereafter we will discuss each of the architectural styles and how they are relevant for e^3value models. For each style an actual e^3value model will be shown as example. Next we will provide a small case study on how architectural styles in e^3value models can be used. We will end with reflecting on the results, forming conclusions and, making suggestions for further research.

2 The e^3value approach

To be self-contained, we give a brief overview on e^3value . The approach is discussed in detail in [9]. The e^3value approach provides modeling constructs for representing and analyzing a network of enterprises, exchanging things of economic value with each other. The methodology is ontologically well founded and has been expressed as UML classes, Prolog code, RDF/S, and a Java-based graphical e^3value editor as well as analysis tool is available for download (see http://www.e3value.com). We use an educational example (see Fig. 1) to explain the model constructs.

Actors (often enterprises or final customers) are perceived by their environment as economically independent entities, meaning that actors can take economic decisions on their own. The Store and Manufacturer are examples of actors. *Value objects* are services, goods, money, or information, which are of economic value for at least one of the actors. Value objects are exchanged by actors. *Value ports* are used by actors to provide or request value objects to or from other actors. *Value interfaces*, owned by actors, group value ports and show

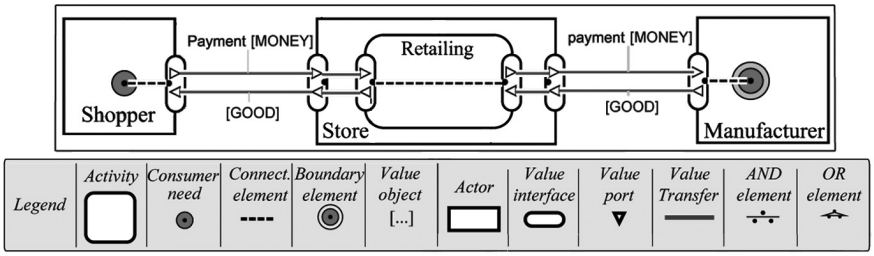

Fig. 1. Educational example

economic reciprocity. Actors are only willing to offer objects to someone else, if they receive adequate compensation in return. Either all ports in a value interface each precisely exchange one value object, or none at all. So, in the example, Goods can only be obtained for Money and vice versa. *Value transfers* are used to connect two value ports with each other. It represents one or more potential trades of value objects. In the example, the transfer of a Good or a Payment are both examples of value transfers. *Value transactions* group all value transfers that should happen, or none should happen at all. In most cases, value transactions can be derived from how value transfers connect ports in interfaces. *Value activities* are performed by actors. These activities are assumed to yield profits. In the example, the value activity of the Store is Retailing. *Dependency paths* are used to reason about the number of value transfers as well as their economic values. A path consists of *consumer needs*, *connections*, *dependency elements* and *dependency boundaries*. A consumer need is satisfied by exchanging value objects (via one or more interfaces). A connection relates a consumer need to a value interface, or relates various value interfaces internally, of a same actor. A path can take complex forms, using AND/OR dependency elements. A dependency boundary represents that we do not consider any more value transfers for the path. In the example, by following the path we can see that, to satisfy the need of the Shopper, the Manufacturer ultimately has to provide Goods.

3 Architecture, Styles and Quality Attributes

The aim of this paper is to identify architectural styles and related quality attributes in e^3value models. To this end, we elaborate first on the notions of *architecture*, *architectural styles*, and *quality attributes* themselves.

Architecture. An architecture is a structure that consists of *elements* and *relations* between these elements, which together create a *coherent system* that provides *value* to its environment [14,32]. Although this conceptualization of architectures is mainly used to describe information systems, the conceptualization is sufficiently broad enough to capture *processes*, leading to a process architecture [24] and *business value* (cf. e^3value) models, as done in this paper. An

e^3*value* model states the structure of enterprises (elements) and economic value transfers (relations) of a networked value constellation, which together creates a coherent system which provides value to its environment, and consequently can be considered as an architecture.

Architecture style. Architectures can have similar features and therefore may have a same *style* [3]. A number of specific architectural styles can be distinguished, but in this paper we will limit ourselves to five styles identified by [3]: *Independent Components*, *Data Flow*, *Data Centered*, *Virtual Machine* and, *Call and Return*. Each style has its own unique features determined by the following [3]: (1) a set of component types that perform some function, (2) a topological lay-out of the components, indicating their relationship, (3) a set of semantic constraints and, (4) a set of connectors that mediate the communication. We will discuss each architectural style more elaborate in the next sections.

It is possible that an architecture has multiple styles [3]. Styles can be: (1) locationally heterogeneous, meaning that an architecture will reveal different styles in different locations, (2) hierarchically heterogeneous, meaning that a component of one style is structured according to the rules of another style and, (3) simultaneously heterogeneous, meaning that any of several styles may well be an apt description of the architecture.

Quality Attributes. Quality attributes are non-functional requirements which an architecture's elements must satisfy to give the elements meaning [28, 31]. For example, the quality attribute "reuse" specifies that an actor (eg. a baker) must be able to reuse a value object (eg. flour) in an other stage (when baking bread). If the baker could not reuse the flour after purchasing it, then the function of the flour has no meaning for the baker. Although quality attributes are often vague, they do provide guidelines and goals for a system's design [3].

In the literature, various relations are mentioned between quality attributes and architecture styles [2,3,31]. Architecture styles can be used for requirements engineering purposes to *identify* quality attributes needed by an architecture's elements and objects [3, 28]. When architecture styles are identified in an architecture, they indicate the quality attributes needed by the elements and objects.

4 Matching Information System Styles to e^3*value* Styles

For the five identifies architectural styles (*Independent Components*, *Data Flow*, *Data Centered*, *Virtual Machine* and, *Call and Return*) in Information Systems, we now analyze whether these styles can also be meaningfully identified in the 17 e^3*value* case studies as published in the past. For each style, we first give the original Information System interpretation and then we analyze how the style can be identified e^3*value* models.

4.1 Architectural Style: Data Flow

In information systems. The *data flow* architectural style is characterized by viewing a system as *separate parts* (elements) which perform operations on *suc-*

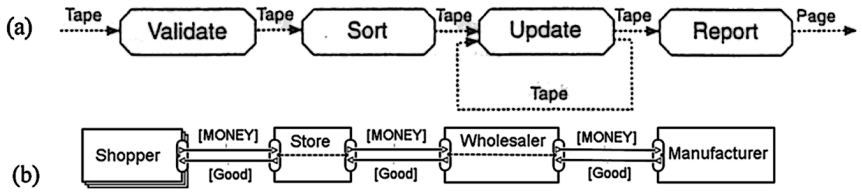

Fig. 2. (a) data flow architecture style, (b) data flow in e^3value

cessive pieces of input (objects) [3] (See Fig. 2a). The output of one part is the input for the next part. The quality attributes related to this architectural style are [3]:

- *Reusability*, stating that an element can be easily reused, for which it is necessary that elements agree on the type of received and produced objects.
- *Modifiability*, stating that elements can be easily added, removed or replaced.

In e^3value. We identified the data flow architecture style in e^3value models by searching for value objects which were transferred between successive actors (see Fig. 2b). The data flow architecture style was found in the following e^3value models [6, 8, 13, 17, 29].

The quality attribute *modifiability* is meaningful for the e^3value data flow architecture style since it very well possible to add or remove elements (actors), think of forward/backward integration/segmentation [16]. In e^3value models objects are however *value objects*, which are either *goods*, *service outcomes*, *money* or (valuable) *information*. So, the question is whether the quality attribute *reusability* is also meaningfully for all types of e^3value 's objects?

- *Goods.* We find the quality attribute meaningful for "goods" (see Fig. 2b). Here, an actor must be able to (re)use the value object s/he receives before s/he can offer (output) the object to the next actor.
- *Money.* Since money can be reused *by definition*, reusability is meaningful for this type of value object (Although it might be a obvious given which does not need to be elicited).
- *Service outcomes.* Reusability is also meaningful for service outcomes. A service outcome is the "result" of the execution of a service [1]. This outcome can be used to offer another actor the same or a similar service (eg. a telecom operator forwarding Internet acces, received from an ISP, to mobile phone users).

4.2 Architectural Style: Data Centered

In information systems. The *data centered* architectural style is characterized by viewing a system with a *central entity*, which is widely accessed by various

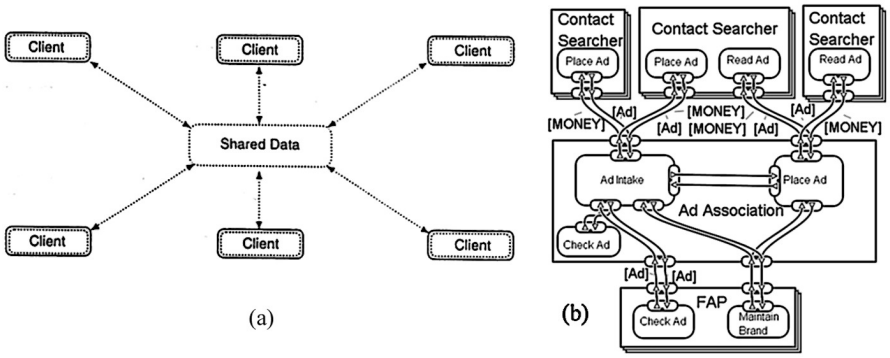

Fig. 3. (a) data centered architecture style, (b) data centered in e^3value

other entities [3] (See Fig. 3a). The various other entities place and retrieve all objects from a central entity. The central entity is able to *integrate* the various *objects*, such that a coherent system is created. The quality attributes related to this architectural style is [3]:

- *Integratability*, stating that the various object transferred to the central entity must be integratable to form either a new product or form a consistent storage area.

In e^3value. We identified the data centered architecture style in e^3value models by searching for various value objects which were transferred to a single actor, whom used these value objects to offer a (new) value object. Fig. 3b provides an example e^3value model in which the data centered style can be found. In this e^3value model, "ads" are centralized and integrated into a single actor. The data centered architecture style was found in the following e^3value models [7, 11, 12, 18].

The question is whether *integratability* is meaningful for all possible value objects (money, goods and, service outcomes) in an e^3value model. For all these value objects the quality attribute integratability is indeed meaningful.

- *Goods.* If a central actor acquires various parts (goods) from various actors to create a new good, then the various must be *integrated* to create the new good (eg. manufacturing a car).
- *Money.* The same holds for money, it is the very nature of money that various (small) amounts can be integrated to create a larger amount.
- *Service outcomes.* Services, and their outcomes, can also be integrated (eg. service bundling [1]). If one of the service outcomes is not integratable with the other service outcomes, then the service bundle obviously cannot be created, making the quality attribute also meaningful for service outcomes.

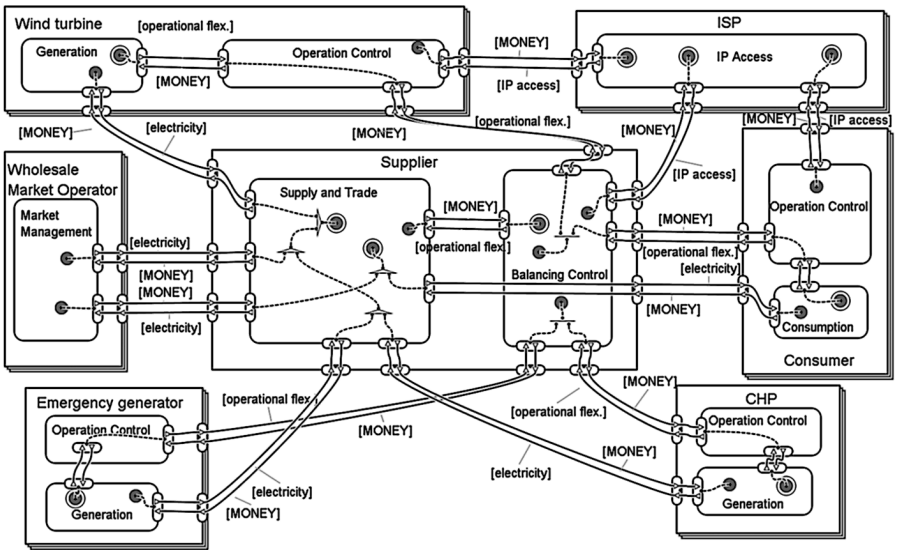

Fig. 4. Independent component in e^3value

4.3 Architectural Style: Independent Components

In information systems. The *independent component* architectural style is characterized by viewing a system as independent components which exchanges objects with each other [3]. Basically an operation is divided in to smaller parts and distributed over various components of the system. The quality attributes related to this architectural style is [3]:

- *Modifiability*, stating that it is (relative) easy to modify one of the *components* without affecting the rest of the system because all parts operate independent on a specific part of the operation.

In e^3value. To find this style in e^3value we searched for e^3value models which as a whole provide a product/service to their environment. Each of the actors is responsible for a specific activity of this product/service. Fig. 4 provides an example layout. The e^3value model shows a balancing system for an electricity marketplace. Various actors perform different functions to create the balancing system. The system as a whole is therefore easily modifiable. The independent component architecture style was furthermore clearly identified in the e^3value models in [7, 10, 17, 19, 26].

Does *modifiability* also hold for e^3value architectures? If we take Fig. 4b) as an example we can see that various component (actors) perform unique activities and exchange various value objects. From a business value perspective it would be relative easy to *modify* this architecture be adding/removing actors are differently distributing value activities over the actor. This indicates that

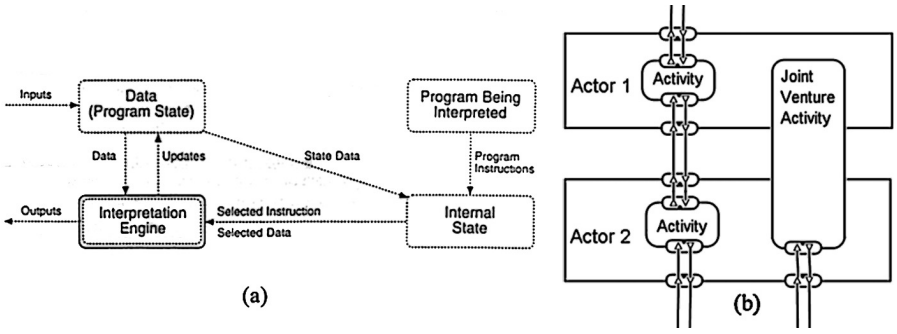

Fig. 5. (a) virtual machine architecture style, (b) virtual machine in e^3value

the quality attribute modifiability is indeed meaningful for this types of e^3value models.

4.4 Architectural Style: Virtual Machine

In information systems. The *virtual machine* architectural style is characterized by viewing a system which simulates some functionality and is not native to one of the components of the system [3] (see Fig.5a). The system simulates, using functions of various components, a system which has no physical properties. The quality attributes related to this architectural style is [3]:

- *portability*, meaning that the system is not bound to a specific platform / location to offer its service.

In e^3value. In terms of enterprises, virtualization may refer to the notion of *virtual organizations*. Although the term virtual organization is often used in literature on e-businesses [4], it is not a common style found in e^3value models. Still it is possible to design a virtual organization (the virtual system) which simulates some functionality based on the functions of the underlying components. Examples are joint-ventures which do exist on paper but *not* in the physical physical. Furthermore, these joint ventures offer value objects which are not native to one of the actors. This is also shown by [22] in which the virtual machine architecture style was clearly identified. Fig. 5b provides the e^3value model of [22]. The model shows how joint-ventures, without physical presence, can be modeled in an e^3value model.

The quality attribute "portability" is relevant for such e^3value models. Since these virtual organization mostly exist on paper, it can be very well possible that the physical organizations forming the virtual organization desire to "relocate" their virtual organization to other countries (eg. due to changing (tax) regulations).

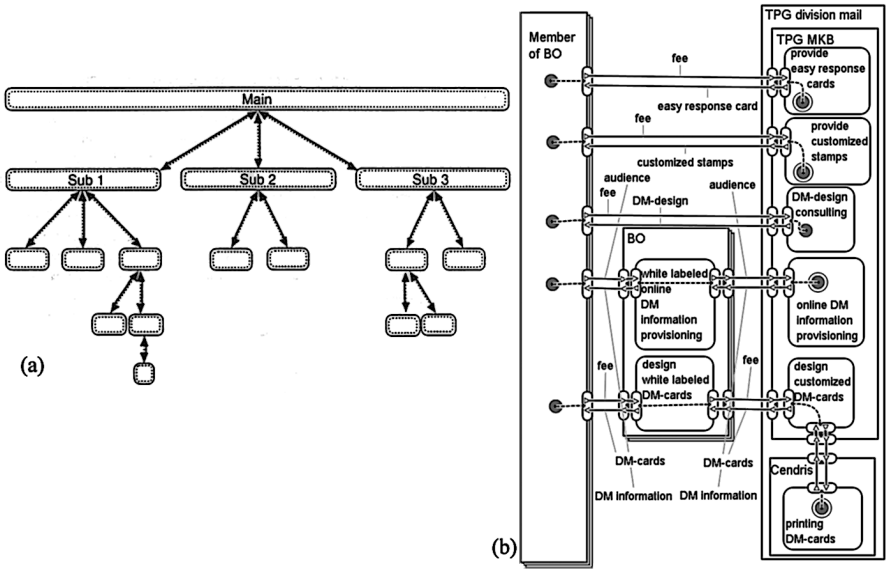

Fig. 6. (a) call & return architecture style, (b) call & return in e^3value

4.5 Architectural Style: Call & Return

In information systems. The *call and return* architectural style is characterized by viewing a system as a main entity which can call smaller sub-entities to perform an action. The outcome of this action is returned to the main entity (see Fig. 6a). The quality attributes related to this architectural style are [3]:

- *Modifiability*, indicating that one of the sub-systems can be easily modified without influencing other (sub)-systems.
- *Scalability*, indicating that the system can handle an increased or decreased in the number of users, without affecting the function of system.

In e^3value. To find this architectural style in e^3value , we had not to examine the networked value constellations modeled in e^3value models per se, but look more at individual actors within a networked value constellation. Often one actor is composed out of various smaller "actors", or departments, and value activities. The 'main' actor requests (call) value objects from the smaller actors, which they provide (return). The call & return architecture style was clearly identified in the e^3value models in [5, 20, 21]. Fig. 6b provides the e^3value model found in [21]. This model shows a main actor/entity (TPG) which calls upon smaller entities (TPG, MKB and Cendris) to perform (return) services, TPG in turn can offer combinations of these services to the outside world.

Do the quality attributes *modifiability* and *scalability* have meaning for such an e^3value system?

– As can been seen in the example e^3*value* model 6b) if one of the sub-actors or value activities is removed then the other entities are not, or barely, affected (eg. they are not connected via dependency paths or (internal) value transfers). This indicates that an e^3*value* model with this style is easily *modifiable.*
– The *scalability* of such an e^3*value* model can be easily affected by adding / removing (sub)-actors to meet the increased demand of consumers. For instance, if "Cendris" is unable to able to print sufficient "DM-cards", TPG could hire an external organization to print additional "DM-cards" (basically TPG ads another sub-system).

Style	**Constituent Parts**			**Quality Attributes**		**Found In**
	Topology	Components	Connectors	Quality	Relevant for	
Data Centered	Star	Actors	Value transfers	Reusability, Modifiability	Value objects	[6, 8, 13, 17, 29]
Data Flow	Linear	Actors, value activities	Value Transfers, Dependency paths	Integratability	Value objects	[7, 11, 12, 18]
Independent Component	Arbitrary	Actors	Value Transfers	Modifiability	Actors	[7, 10, 17, 19, 26]
Virtual Machine	Arbitrary	Actors,	Value transfers joint ventures	Portability	Actors	[22]
Call & return	Hierarchy	Actors, value activities	(Internal) value transfers	Modifiability, Scalability	Actors	[5, 20, 21]

Table 1. Overview architectural styles

5 How to use architectural styles in e^3*value* models

So far, we have only analyzed if architectural styles could be found in existing e^3*value* models and if they lead to the identification of meaningful quality attributes for e^3*value* models. In this section we demonstrate how to find architectural styles in an e^3*value* model and how related quality attributes can be used by organizations to get better understanding of their business value context.

Case Study. Mobzilli is an Internet company offering the service "location based advertisement" (www.mobzilli.com). This service offers organizations the possibility to bound advertisements to geographical locations. Potential customers can request the advertisements utilizing a small application on their mobile phone. So if a customer would be in a shopping center s/he would be able to request the advertisements of the shops in her/his vicinity using her mobile phone.

The e^3value model. Fig. 7 provides the e^3*value* model for Mobzilli. We do not elaborate on the design process of this model in this paper. The model shows

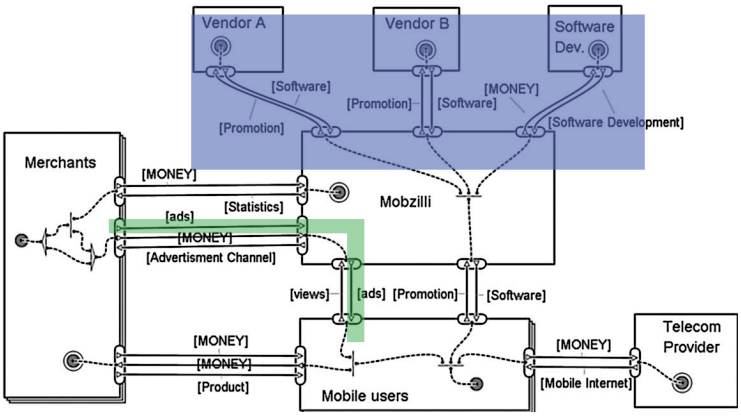

Fig. 7. Mobzilli e^3*value* model

that Mobzilli acquires a piece of (open-source) software from "Vendor A" and another piece of (open-source) software from "Vender B". Furthermore, Mobzilli hires "Software Developers" to create additional software and integrate all the pieces of software. Mobzilli offers their service to "Merchants", who pay for this service and provide "Ads". Mobzilli transfers these "Ads" to "Mobile Users".

Finding styles. In Mobzilli's e^3*value* model, it can be seen that the value objects "Software" (twice) and "Software Development" are acquired by a single actor: Mobzilli (see blue highlighted area). This lay-out matches the *data centered* architectural styles described in section 4.2. The quality attribute related to this style is *integratability*. As a result, the value objects "Software" and "Software Development" must meet the quality attribute *integratability* to have meaning to Mobzilli.

There is a second architectural style visible in Mobzilli's e^3*value* model (green highlighted area). The value object "Ads" is first transferred from "Merchants" to Mobzilli and then from Mobzilli to "Mobile users". This layout matches the architectural style *data-flow* described in section 4.1. Although this is a very small of example of this style, a successive piece of information is still transferred between actors *and* is reused by an actor (it is changed from a normal advertisement into an advertisement suitable for mobile users). The quality attributes related to this style, and thus the value object "Ads" is *reusabability*.

Quality Attribute Requirements. The e^3*value* model for Mobzilli seems to be financially feasible, but for this configuration to be meaningful, other requirements must be met, namely the aforementioned quality attributes:

- *Integratability.* If the value objects "Software" from "Vendor A", "Software" from "Vendor B", and "Development" from "Software Developers" are not *integratable*, Mobzilli will not be able to offer its service.

– *Reusability.* If the value object "Ads" acquired from "Merchants" is not *reusable* (eg. proper file format) by Mobzilli, then Mobzilli is not able to forward the "Ads" to "Mobile Users". This would make their service obsolete.

So, we were able to find architectural styles in Mobzilli's e^3value model which were used to identify meaningful quality attributes. For Mobzilli to make their service meaningful they must meet these requirements to avoid problems in later stages of IS development.

6 Conclusion

In this paper we have shown that we were able to identify architectural styles in e^3value models. In addition, we were able to use these styles to find meaningful quality attributes for value objects and actor in e^3value models. By using the work of [3] on architectural styles we were able to find five similar styles in most e^3value models created between 1999 and 2007. In addition, we have shown why and how quality attributes related to the five architectural styles are also meaningful for e^3value models. Furthermore, we have demonstrated with a small case study how we can use architectural styles in e^3value models to reason about quality attributes during the early stages of requirements engineering. This early-phase requirements engineering should aid in not only a better business configuration but should also lead less problems in later stages [33].

The next logical step would be to formalize the styles exactly and determine if more styles and related quality attributes can be found in e^3value models.

Acknowledgments The authors wish to thank Wojtek Chowanski and Reza Ladchartabi from Mobzilli for providing case study material and for having many fruitful discussions. This work has been partly sponsored by NWO project COOP 600.065.120.24N16.

References

1. H. Akkermans, Z. Baida, and J. Gordijn. Value webs: Ontology-based bundling of real-world services. *IEEE Intelligent Systems*, 19(44):23–32, July 2004.
2. J. Asundi, R. Kazman, and M. Klein. Using economic considerations to choose amongst architecture design alternatives. Technical report, Software Engineering Institute, 2001.
3. L. Bass, P. Clements, and R. Kazman, editors. *Software Architecture in Practice*, chapter Moving from Qualities to Architecture, pages 93–121. Addison Wesley, 1998.
4. W.H. Davidow and M.S. Malone. *The Virtual Corporation*. Harper Business, 1992.
5. Zs. Derzsi and J. Gordijn. Value-based business-ict alignment: A case study of the mobile industry. In *Proceedings of the 12th Research Symposium on Emerging Electronic Markets (RSEEM)*, pages 83–95, Amsterdam, (NL), 2005. VU.

6. Zs. Derzsi and J. Gordijn. A framework for business/it alignment in networked value constellations. In T. Latour and M. Petit, editors, *Proceedings of the workshops of the 18th International Conference on Advanced Information Systems Engineering (CAiSE 2006)*, pages 219–226, Namur, B, 2006. Namur University Press.
7. Zs. Derzsi, J. Gordijn, K. Kok, H. Akkermans, and YH. Tan. Assessing feasibility of it-enabled networked value constellations: A case study in the electricity sector. In J. Krogstie, A. Opdahl, and G. Sindre, editors, *19th International Conference, CAiSE 2007, Trondheim, Norway, June 2007, Proceedings*, volume 4495 of *LNCS*, pages 66–80. Springer Verlag, 2007.
8. J. Gordijn. E3value in a nutshell. Technical report, HEC University Lausanne, 2002.
9. J. Gordijn and H. Akkermans. Value based requirements engineering: Exploring innovative e-commerce idea. *Requirements Engineering Journal*, 8(2):114–134, 2003.
10. J. Gordijn and H. Akkermans. Business models for distributed energy resources in a liberalized market environment. *The Electric Power Systems Research Journal*, 77(9):11781188, 2007.
11. J. Gordijn, H. Akkermans, and H. Van Vliet. Requirements for e-commerce applications are created rather than elicited. In *NOSA'99 - Proceedings of the second nordic workshop on software architecture*, 1999.
12. J. Gordijn and H. De Bruin. Scenario methods for viewpoint integration in e-business requirements engineering. In *Proceedings of the 34th Hawaii International Conference On System Sciences*. IEEE, 2001.
13. J. Gordijn, E. Yu, and B. Van Der Raadt. E-service design using i* and e3value modeling. *IEEE Software*, 23(3):26–33, May 2006.
14. IEEE Architecture Working Group. Ieee recommended practise for architectural description of software-intensive systems. *IEEE std 1471-2000*, 2000.
15. Michael Jackson. *Problem frames: analyzing and structuring software development problems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.
16. G. Johnson and K. Scholes. *Exploring Corporate Strategy*. Pearson Education Limited, Edinburgh, UK, 2002.
17. V. Kartseva, J. Gordijn, and YH. Tan. Analysing preventative and detective control mechanisms in international trade using value modelling. In M. Janssen, H. Sol, and R. Wagenaar, editors, *Proceedings of the 6th International Conference on Electronic Commerce*, pages 51–18. ACM Press, 2004.
18. V. Kartseva, J. Gordijn, and YH. Tan. Towards a modelling tool for designing control mechanisms in network organisations. *International Journal of Electronic Commerce*, 10(2):57–84, 2005.
19. V. Kartseva, J. Hulstijn, J. Gordijn, and YH. Tan. Modelling value-based inter-organizational controls in healthcare regulations. In R. Suomi, R. Cabral, J. Hampe, A. Heikkila, J. Jarvelainen, and E. Koskivaara, editors, *Proceedings of the 6th IFIP conference on e-Commerce, e-Business, and e-Government (I3E'06)*, volume 226 of *IFIP International Federation for Information Processing*, pages 278–291. Springer, 2006.
20. S. de Kinderen and J. Gordijn. Matching complex consumer needs with e-service bundles. In P. Walden, M. Markus, J. Gricar, and G. Lenart, editors, *Proceedings of the 19th BLED conference*, Maribor, SL, 2006. University of Maribor.
21. S. de Kinderen and J. Gordijn. A consumer needs-driven approach for finding it-service bundles. Unpublished, 2007.

22. C. Kort and J. Gordijn. *Modeling strategic partnerships using the e3value ontology - A field study in the banking industry.* 2007. Accepted as book chapter in the Ontologies Handbook, edited by Peter Rittgen.
23. A. Osterwalder. *The Business Model Ontology - a proposition in a design science approach.* PhD thesis, University of Lausanne, Lausanne, Switzerland, 2004.
24. V. Pijpers. Alignment through styles and qualities. Master's thesis, University of Twente, 2005.
25. V. Pijpers and J. Gordijn. Bridging business value models and process models in aviation value webs via possession right. In *Proceedings of the 39th Hawaii International Conference On System Sciences (HICCS)*. IEEE, 2007.
26. V. Pijpers and J. Gordijn. Does your role in a networked value constellation match your business strategy? - a conceptual model-based approach. In M. Markus, J. Hampe, J. Gricar, A. Pucihar, and G. Lenart, editors, *Proceedings of the 20th BLED conference*. University of Maribor, Maribor, SL, 2007.
27. V. Pijpers and J. Gordijn. E3forces : Understanding strategies of networked e3value constellations by analyzing environmental forces. In J. Krogstie, A. Opdahl, and G. Sindre, editors, *Proceedings of the 19th International Conference, CAiSE 2007, Trondheim, Norway*, volume 4495 of *LNCS*, pages 188–202. Springer Verlag, 2007.
28. M. Shaw. Comparing architectural design styles. *IEEE Software*, November 1995.
29. J. Soetendal, J. Gordijn, and E. Paalvast. Governance selection in value webs. In M. Funabashi and A. Grzech, editors, *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government. Proc. 5th IFIP Conf. e-Commerce, e-Business, and e-Government*, pages 17–31, Berlin, D, 2005. Springer,.
30. D. Tapscott, D. Ticoll, and A. Lowy. *Digital Capital - Harnessing the Power of Business Webs.* Harvard Business School Press, Boston, MA, 2000.
31. R.J. Wieringa. *Design Methods for Reactive Systems.* Morgan Kaufman Publishers, San Fransisco, CA, 2003.
32. R.J. Wieringa, H.M. Blanken, M.M. Fokkinga, and P.W. Grefen. Aligning application architecture to the business context. *CAISE*, 2003.
33. E. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)*, pages 226–235, 1997.