Value Framing: A Prelude to Software Problem Framing

Roel Wieringa[*]   Jaap Gordijn[†]   Pascal van Eck[‡]

Abstract

Software problem framing is a way to find specifications for software. Software problem frames can be used to structure the environment of a software system (the machine) and specify desired software properties in such a way that we can show that software with these properties will help achieve the required effects in the environment. Actually framing a software problem, i.e. finding suitable problem frames of a given situation, is creative activity for which no guidelines are currently known. In this paper, we propose to use an idea exploration technique called e^3-value to find software problem frames. The e^3-value methodology is an approach to help business analysts solve the problem of designing a networked enterprise, defined as a set of businesses or business units that make money by performing value exchanges over a computer network. The outcome of e^3-value is viewed by business managers as a solution, *but it is a* problem *for software engineers who have to implement this idea. In this paper we illustrate the combination of e^3-value with problem framing by means of a small example from real life, and discuss the research questions that come out of this.*

1 Introduction

Software problem diagrams structure the environment of a software system (the machine) into domains with various interfaces towards each other and towards the machine [6]. The interfaces are places where shared events occur. In addition, a problem diagram shows the *requirement,* which consists of desired behavior in some domains. It also shows *domain properties* and a *specification* of desired machine properties which is expected to ensure that the requirement will be satisfied. The machine specification, the domain requirement, and the given domain properties are related by a correctness argument. We call the analyst's concern to find a machine specification such that this correctness argument can be given the *problem concern.*

When we recognize a class of problems, we can abstract it into a problem frame. We are interested in classes of software problems that arise in the context of business automation, such as in the development of business information systems, workflow systems, groupware or e-commerce systems. More in general, the environment of these systems is a business context in which business actors (people playing organizational roles) perform activities to generate value. The question that interests us is: How to find the relevant problem frame(s) for these problems? Or, in less esoteric English: How to frame the business problems so that the software engineer can derive software specifications from his or her problem analysis?

In this paper we zoom in on one particular class of business software problems, namely the problem of developing multi-business information systems. Examples of this kind of problems are the problem of trading music, data, video or other digital content on-line. Other examples include digital service provision among business units of a large company. The relevance of this problem stems from the fact that business increasingly engage in cooperative networks of otherwise independent enterprises, and that large enterprises more and more structure themselves internally as networks of profit and loss responsible business units.

What makes problem framing difficult in these cases is that more than one enterprise is involved in these problems, each with their own universe of discourse, business goals, business processes, legacy systems, and infrastructures. It is hard to find the relevant problem domains, because different businesses may talk in different languages about their domains. This in turn engenders confusion about domain properties. It is also hard to reach agreement on the requirements because different businesses have different goals. Confidentiality of available software and business processes makes it even harder to discuss domain properties and requirements. And to compound the problem further, most businesses will be reluctant to build new systems for just this cooperation, and so software specifications that do not agree with available legacy systems will not be acceptable to most businesses. But the presence and nature of legacy system will not be easily revealed by the businesses.

All of this makes it hard to structure the software problems in business networks.

In the past, the e^3-*value* method has been used to explore e-commerce ideas in about five different e-commerce consultancy projects [3]. The outcome of using e^3-*value* is a specification of a network of business actors that perform value activities and that exchange value, and an estimate of the profitability of these value activities and exchanges for each of the business actors. In this paper, we discuss how this can be used as input to a software problem framing process. We do this using a small example, presented in the next section. To keep matters simple, we ignore value activities, performed internally by business actors, and show only value exchanges between business actors.

Before we present the example, we should point out that the value perspective taken in this paper is not yet common in requirements engineering. We think that addition of a value orientation is necessary to help the requirements engineer make choices in software problem analysis, and that a value orientation can be added in a way that avoids vague talk about marketing and commercial benefits. We think that including a value orientation in software requirements decreases the chance on irrelevant requirements and failed software.

In the next section, we introduce our running example and in section 3 we summarize the e^3-*value* method. Section four distinguishes the three abstraction levels that we distinguish in our problems, namely the value level, the business process level, and the business system level. Section 5 then shows how the value perspective can be used to explore solutions, whose consequences can then be traced to the business process and business system models. Section 6 shows how the business system models can be transformed into software problem diagrams, using the value perspective to identify the requirements.

2   The NGO example

We illustrate our proposal with a real life example of a collection of European Non-Governmental Organizations (NGOs) in the domain of international voluntary service. Each NGO sends out volunteers from its own country to projects offered by NGOs in other countries (as well as its own projects) and accepts volunteers from other countries in its own projects. The purpose is to create possibilites for learning from other cultures and to help in local social development. The NGOs maintain contact with each other about projects offered and about volunteers, and there is a supranational umbrella organization that loosely coordinates the work of the (independent) NGOs. Some of the NGOs receive government subsidies, most do not. In the projects offered, only work is done that cannot be performed commercially.

Each NGO has a web site, a general ledger system for the financial administration, a simple workflow management system (WFM) to manage the workflow for each volunteer, a project database of running projects, and a customer relationship management system (CRM) to manage information about volunteers that have shown interest in voluntary service. Since the NGOs vary widely in age, size and level of professionalism, and since they are independent, the implementations of these systems also vary widely and do not provide compatible interfaces. Recently, an application service provider (ASP) has offered to handle the WFM/CRM systems of all NGOs. The question to be solved is how this can be done in such a way that the ASP makes money and the NGOs save money.

Although this example is about non-profit organizations, the arguments to enter the network of cooperating NGOs are stated in terms of value added and costs saved by the members in this cooperation. This makes e^3-*value* a useful technique to solve the business problem whether a cooperation can be organized in such a way that value is added for all concerned.

3   The e^3-*value* method

The e^3-*value* methodology is specifically targeted at the design of business networks, as for example in e-commerce and e-business. Business networks produce, distribute and consume things of economic value jointly. The rapid spread of business networks, and of large enterprises that organize themselves as networks of profit and loss responsible business units, is enabled by the capability to interconnect information systems of various businesses and business units. In all cases, the trigger of an application of e^3-*value* are the networking opportunities perceived to be offered by information and communication technology (ICT). The use of e^3-*value* is then to explore whether the networking idea can really be made profitable for all actors involved. We do so by thoroughly conceptualizing and analyzing such a networked idea, to increase shared understanding of the idea by stakeholders involved. The results of an e^3-*value* track are sufficiently clear to start requirements engineering for software systems. In the following, we will indicate networks of businesses and networks of business units by the blanket term *networked enterprises*. We will also call the software systems that support business processes *business systems*. Examples of business systems are information systems, workflow systems and special-purpose application software.

Before the requirements on the information technology used by networked enterprises can be understood, the goals of the network itself need to be understood. More precisely, before specifying the business systems and communications between these, it is important to understand how

various enterprises in the network create, distribute and consume objects of economic value. The e^3-*value* method has been developed in a number of action research projects[1] as a method to determine the economic structure of a networked enterprise.

3.1 The e^3-*value* methodology in a nutshell

We illustrate the concepts of e^3-*value* using Figure 1, which shows a value model of the current network of NGOs.

Actor. An actor is perceived by its environment as an independent economic (and often also legal) entity. An actor intends to make a profit or to provide a non-profit service. In a sound, sustainable, business model each actor should be capable of creating a net value. Commercial actors should be able to make a profit, and non-profit actors should be able to create a value that in monetary terms exceeds the costs of producing it in order to sustain. Each NGO, the umbrella organization, each project and each volunteer in our example is an actor.

Value Object. Actors exchange value objects, which are services, products, money, or even consumer experiences. A value object is of value to at least one actor. In Figure 1, Assigned volunteer and Assigned project are value objects.

Value Port. An actor uses a value port to show to its environment that it wants to provide or request value objects. A value port has a direction, namely outbound (e.g. a service provision) or inbound (e.g. a service consumption). A value port is represented by a small arrowhead that represents its direction.

Value Transfer. A value transfer connects two equidirectional value ports of different actors with each other. It is one or more potential trades of value objects between these value ports. A value transfer is represented by a line connecting two value ports. Note that a value transfer may be implemented by a complex business interaction containing data transmissions in both directions. The direction of a value transfer is precisely that: the direction in which value is transfered, not the direction of data communications underlying this transfer.

Value exchange. Value transfers come in economic reciprocal pairs, which are called value exchanges. This models 'one good turn deserves another': you offer something to someone else only if you get adequate compensation for it.

---

[1]These are real life projects in which the researcher uses the technique together with business partners, followed by a reflection on and improvement of the technique. For the business partners, these projects are not research but commercial projects where they pay for the results. The researcher has the dual aim to do a job for the business and to learn something from doing it.

Value Interface. A value interface consists of ingoing and outgoing ports of an actor. Grouping of ingoing and outgoing ports model economic reciprocity: an object is delivered via a port, and another object is expected in return. An actor has one or more value interfaces, each modelling different objects offered and reciprocal objects requested in return. The exchange of value objects across one value interface is atomic. A value interface is represented by an ellipsed rectangle.

Market segment. A market segment is a set of actors that for one or more of their value interfaces, ascribe value to objects equally from an economic perspective. Naturally, this is a simplification of the real world, but choosing the right simplifications is exactly what modeling is about. A market segment is represented by a stack of actor symbols. NGOs is an example of such a market segment.

With the concepts introduced so far, we can describe who exchanges values with whom. If we include the end consumer as one business actor, we would like to show all value exchanges triggered by the occurrence of one end-consumer need. This considerably enhances a shared understanding of the networked enterprise idea by all stakeholders. In addition, to assess the profitability of the networked enterprise, we would like to do profitability computations. But to do that, we must *count* the number of value exchanges triggered by one consumer need. To reace an end-consumer need and do profitability computations, we include in the value model a representation of *dependency paths* between value exchanges. A dependency path connects value interfaces in an actor and represents triggering relations between these interfaces. A dependency path has a direction. It consists of dependency nodes and connections.

Dependency node. A dependency node is a stimulus (represented by a bullet), an AND-fork or AND-join (short line), an OR-fork or OR-join (triangle), or an end node (bull's eye). As explained below, a stimulus represents a trigger for the exchange of economic value objects, an end node represents a model boundary.

Dependency connection. A dependency connection connects dependency nodes and value interfaces. It is represented by a link.

Dependency path. A dependency path is a set of connected dependency nodes and connections with the same direction, that leads from one value interface to other value interfaces or end nodes of the same actor. The meaning of the path is that if a value exchange occurs across a value interface I, then value interfaces pointed to by the path that starts at interface I are triggered according to the and/or logic of the dependency path. If a branch of the path points to an end node, then this says "don't care".
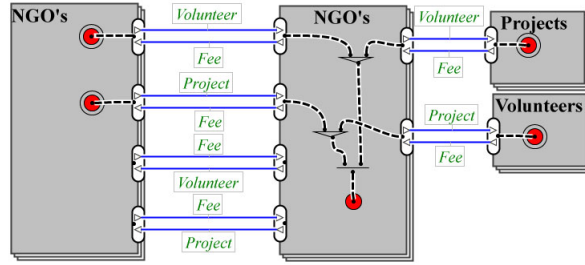
Figure 1. Value Model of the current NGO network.

Dependency paths allow one to reason about a network as follows: When an end consumer generates a stimulus, this triggers a number of value interfaces of the consumer as indicated by the dependency path starting from the triggering bullet inside the consumer. These value interfaces are connected to value interfaces of other actors by value exchanges, and so these other value interfaces are triggered too. This in turn triggers more value interfaces as indicated by dependency paths inside those actors, and so on.

Our value model now represents two kinds of coordination requirements: Value exchanges represent the need to coordinate two actors in their exchange of a value object, and dependency paths indicate the need for internal coordination in an actor. When an actor exchanges value across one interface, it must exchange value across all value interfaces connected to this interface. This allows us to trace the value activities and value exchanges in the network triggered by a consumer need, and it also allows us to estimate profitability of responding to this need in this way for each actor. For each actor we can compute the net value of the value objects flowing in and those flowing out according to the dependency path.[2]

Note that an e^3-*value* model should not be seen as some business process model [5]. A value model shows only objects that are of *economic value* for someone, while a business model shows many objects that are not of direct value (but are e.g. needed to control the process flow). Additionally, business process models do not have the notion of economic reciprocity, which is in e^3-*value* captured by the 'value interface' concept. It is our experience that decisions on the exchange of economic values should not be confused with decisions on the business process that put such exchanges into operation. The relation between value models and process models is further explored in [9].

---

[2]The concept of a dependency path is reminiscent to that of use case maps [1], but it has a different meaning. A use case map represents a sequential scenario. Dependency paths represent coordination of value interfaces, and dependency paths in different actors may among each other not have an obvious temporal ordering, even if triggered by the same stimulus.

3.2 Example: The NGOs in the current situation

Figure 1 represents the current situation as a value model. The diagram shows the NGO market segment twice, because we want to show that there exists interaction between NGOs. An NGO serves two types of actors: Volunteers and projects. The task of a NGO is to match a volunteer to a project. If a match is successful, the project obtains a volunteer, and a volunteer obtains a project. Both the volunteer as well as the project pay a fee for this service.

Volunteers need a project to work for; Projects need volunteers. These needs are shown in Figure 1 by stimuli.

The match itself is represented as an AND-join. Following the paths connected to the join, it can be seen that for a match, a volunteer and a project is needed. These volunteers and projects can be obtained from the NGO's own customer base, or can be obtained from other NGO's as is represented by OR-joins.

Note that Figure 1 shows only part of the dependency path. Specifically we represent that for matching purposes the rightmost NGO uses volunteers and projects from its own base or from other NGO's. However, the leftmost NGO's also do matching. Paths associated with these matchings are not presented.

We will skip the profitability estimations for this example, because these play no role in the following argument. The interested reader can find examples in earlier publications [4, 3].

4 Value Models, Business Processes and Information Systems

Let us now introduce a manager, a business process designer, and a technology designer in each NGO. In each NGO, these might very well be the same person, but we treat them as different roles.

For the manager, a value model represents a business solution, namely how a networked enterprise does business. For the business process designer, it represents a problem, namely what processes are needed to realize these value ex-

changes? For the current situation of the NGOs we know the solution to this problem, namely the processes currently operating at the NGOs. The current business processes operating in the NGOs are these:

- Network management: Entry and exit of an NGO in the network of NGOs.

- Core processes: (a) Acquisition of own projects, (b) Acquisition of own volunteers (c) Matching: (i) placement of incoming volunteers in own projects, (ii) placement of own volunteers in projects of other NGO. (d) Volunteer preparation (training)

- Financial processes

- ICT support processes

- HRM processes

- ICT management processes

- Controlling processes: (a) Policy making (b) Quality control (c) Incident response

Operationalizing the value model into a business process model requires us to think through the consequences of each end-consumer stimulus from start to finish, also including all possible exceptions. For example, how are the projects and volunteers informed about the matching? What if they refuse the matching proposed by the NGO? What if they don't respond? What if no match can be found? What if there is more than one match?

In their turn, business processes are supported by technology, ranging from buildings and furniture to ICT, telephone and fax. A design of business processes is a solution for the business process designer, but a problem for the technology designer. We are interested in the information technology required to support the business processes.

Figure 2 shows a communication diagram of the currently available business systems in each NGO. Communication diagrams are undirected graphs in which the nodes represent systems and the edges represenmt communication channels. Nesting of nodes represents containment. Communication diagrams are used to represent the (de)composition of a system (in our case the composition of the context of the systems of interest) [10]. The difference with Jackson's context diagrams is that the nodes in the diagram represent systems, not sets of coherent phenomena, and that we do not single out one system as the machine of interest. This difference is very slight because any system can be viewed as a set of coherent phenomena, and any coherent set of phenomena can be viewed as a system.

Figure 2 shows a number of business systems in NGO1. Each system consists of people and technology, such as software, paper, telephones, fax, etc. Each NGO has systems with the same functionality, but different NGOs may
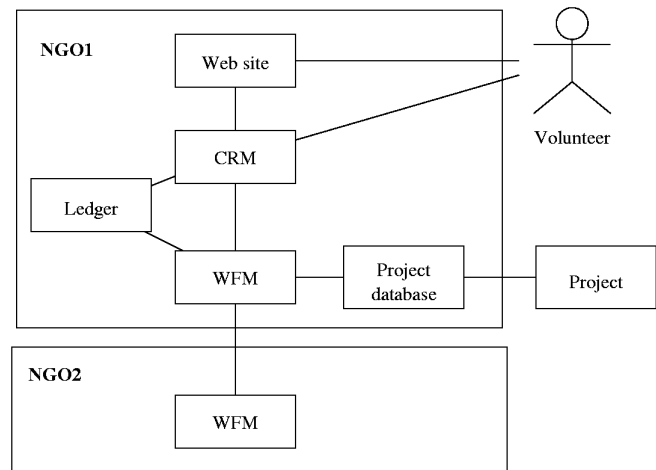

Figure 2. Communication diagram of current situation.

use different people-technology combinations to implement this functionality.

The diagram also shows a number of communication channels between these systems. Each communication channel is a means to share information between two actors. The meaning of the diagram is that each channel is reliable and instantaneous; if we want to include an unreliable communication channel with delays, we should include the channel as system in the diagram and connect it with lines to the systems communicating through the channel; the remaining lines then represent reliable instantaneous communication channels. The WFM system of NGO1 communicates with WFM systems in all other NGOs. Not shown is the fact that the communication between WFMs of different NGOs currently is done mostly by telephone, fax, email and paper mail. For brevity, we do not list the shared phenomena occurring at the interfaces indicated by the communication channels.

Figure 2 also shows the context of NGO1, which consists of volunteers and projects (and other NGOs). We now add a requirement to the diagram:

R1 Match a volunteer to a project.

This is the mission of the network of NGOs and it is the purpose of the core business process of each NGO. Note that each of the business processes listed earlier will generate a requirement like this. We call the resulting diagram (Figure 3) a *solution diagram* because it shows the current solution to this requirement. Unlike a problem diagram, a solution diagram does not represent a machine to be specified

In the current solution, requirement R1 can be satisfied in two ways, either by matching volunteers and own projects
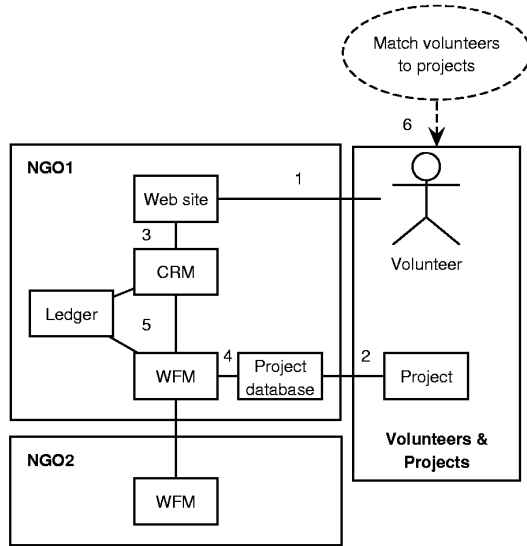
5

Figure 3. Solution diagram of current situation.

of the NGO, or by asking another NGO to place the volunteer on one of its projects. We here consider the first option. The concern raised by R1 is whether the phenomena at the interfaces shown in the problem diagram are sufficient to guarantee R1. In other words, the concern is whether in the current solution the following correctness argument is valid:

1. If a volunteer entered his or her data and preferences correctly through the NGOs web site,

2. and the project data have been entered correctly in the project database,

3. and the CRM system obtains the volunteer data from the web site

4. and the WFM system can access the project data from the database

5. and can access volunteer data from the CRM system

6. then the WFM system must match volunteer and project data as required.

Numbers refer to lines in the problem diagram. Now, what kind of argument is this? We are not specifying any of the systems shown in the diagram, so why bother about this at all? Obviously, the NGOs currently can be *observed* to do their work, so why produce an *argument* that they can do their work?

The point of constructing the argument is that it gives a reason for the presence of each of the interfaces in the problem diagram in terms of achieving R1. R1 in turn is needed

to perform a business process, which in turn was needed to implement the value model. The above correctness argument thus allows us to understand the value currently added by current technology of NGOs. The technology correctness argument also allows us to validate our understanding of the business processes: Did we really think through all consequences of an end-consumer stimulus? Did we really take care of all exceptions?

Thirdly, the problem concern can also be used to specify desired behavior of any of the systems in the NGO, still assuming the current way of working. Suppose we want to redesign the WFM system. If all other systems in the NGO are given, we can then extract desired interface behavior of the WFM from the problem concern. This allows us to find the specification of a reimplemented WFM system that plays a role in achieving R1 as indicated by the correctness argument.

5 Value-Oriented Idea Exploration

5.1 Option (1): ICT Outsourcing

A main concern for NGOs and the umbrella organization is to have cost-effective ICT support for their processes, while preserving or improving the quality of service offered to volunteers. Specifically, NGOs have indicated that the different WFM and CRM systems present in the NGOs are candidates for cost-cutting operations. We saw in our current problem analysis that each NGO exploits its own WFM and CRM. One option for cost-cutting is therefore to replace all these WFM and CRM systems by one system, to be used by all NGOs. This system can be placed at the umbrella organization, who then acts as an Application Service Provider (ASP). This means that NGOs connect to the Internet, and use the WFM and CRM system owned by the umbrella organization. To keep costs low, NGOs use a browser to interact with the WFM and CRM system of the umbrella. This leads to the value model in Figure 4.

The exchanges introduced in Figure 1 remain intact. The umbrella organization acting as ASP is introduced in the value model. In the value model we see that the ASP offers a matching service, which is the same main functionality of the old WFM and CRM application offered to the NGO. This functionality is offered by the umbrella orgizantion, while the rest remains the same. This implies that the NGO interacts from a value perspective exactly the same as in Figure 1.

This solution has no impact on the business processes but it has impact on the technology situation, as shown in the communication diagram of Figure 5. From an ICT perspective, there is now only one WFM/CRM application (instead of many different ones), buth there are still as many instances of it as there were applications in the old situa-
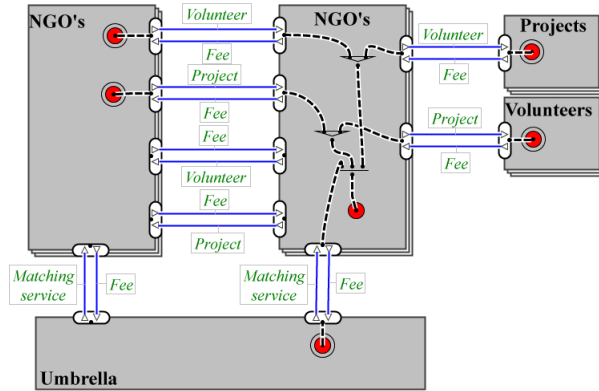
Figure 4. Value Model for an ASP solution (ICT outsourcing).

Figure 5. Communication diagram of software systems in the ASP solution (ICT outsourcing).

tion, only they are now provided by one party, and they are all exactly the same. By doing so, the umbrella organization can exploit economies of scale and thus yields a more cost-effective ICT service for the NGOs. The NGOs use the WFM/CRM applications exactly in the same way as before; their business processes do not have to change.

5.2 Option (2): Business process outsourcing

A second option is to outsource not only the software, but the entire business process to do the matching to the umbrella organization. Figure 6 show the value model of this. The matching is now done for all NGOs using the same base of volunteers and projects. This allows for doing global matching, rather than doing local matching for each NGO separately.

In this solution, there is a drastic change in the value exchanges: Each NGO pays for a match to the umbrella organization. The role of a NGO is not so much the matching itself, but attracting volunteers and projects in their specific region. So, exchanges between NGOs disappear. They exchange now value objects using the umbrella organization as an intermediate.

From an ICT perspective, there is one matching (WFM/CRM) system for all NGOs (Figure 7). These NGOs only need a (web-based) client to interact with the matching engine. From a functional point of view complexity is added, because the WFM engine/process should cover all NGO's coherently, whereas in the ICT outsourcing alternative the scope of each WFM is limited to a single, specific NGO. This situation may be more cost-effective than the ICT outsourcing solution, but it may nevertheless be unattractice to the NGOs because they seem to lose their primary function.
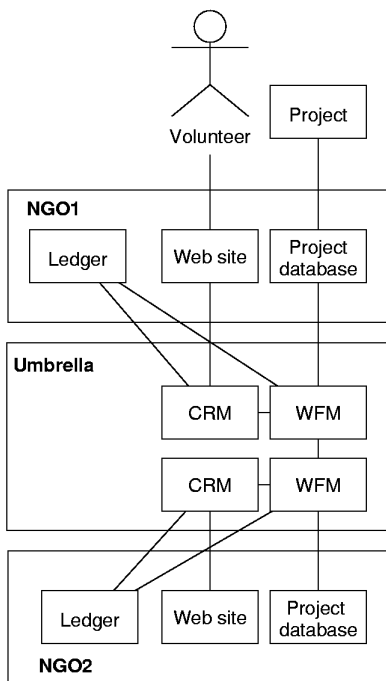
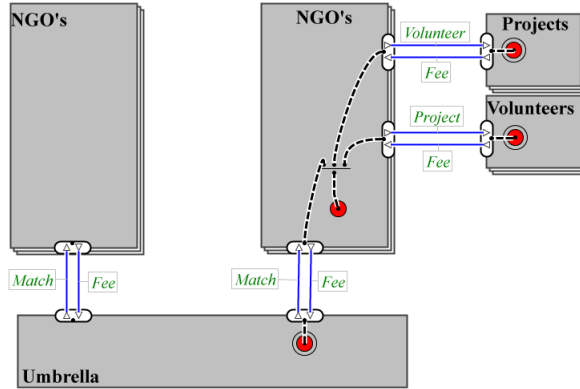## 6 Software Problem Concerns

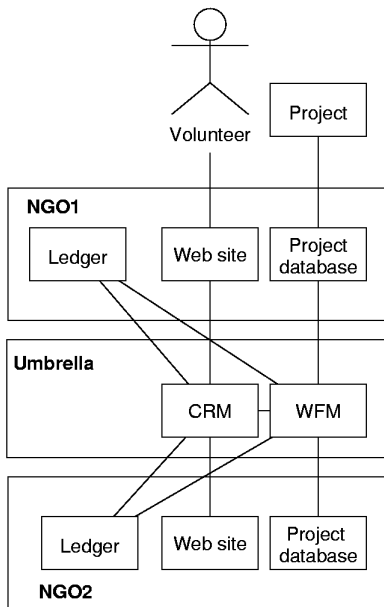**Figure 6. Value model for a the business process outsourcing solution.**

Figure 7. Communication diagram of software systems in the business process outsourcing solution.

From this point on, the value concerns do not play a role anymore. We are now worried about costs, not values. The value models allow managers to organize their thoughts and discuss possible business solutions. But for the software engineer implementing the software part of a business solution, value models are problem statements. One problem concern for the solutions proposed is to implement requirement R1 mentioned earlier.

In the ICT outsourcing solution, one way to implement this, as before, is to do a local search. The corresponding software concern is the same as before, but now it refers to

a different solution diagram (Figure 8):

1. If a volunteer entered his or her data and preferences correctly through the NGOs web site,

2. and the project data have been entered correctly in the project database,

3. and the CRM system obtains the volunteer data from the web site

4. and the WFM system can access the project data from the database

5. and can access volunteer data from the CRM system

6. then the WFM system must match volunteer and project data as required.

We can now turn the business solution diagram of Figure 8 into a software problem diagram by designating the CRM or WFM as the machine to be specified. For example, if we need to develop a CRM, this will be the machine of interest. We can drop all interfaces that do not play a role in the correctness argument and omit all systems that thereby become disconnected from the machine of interest. Space limitations prevent us from working this out in detail.

In the business process outsourcing solution, local search is not possible and we get the following argument, stated in terms of the business process outsourcing problem diagram (Figure 9):

1. If a volunteer entered his or her data and preferences correctly through the NGOs web site,

2. and the project data have been entered correctly in the project database,

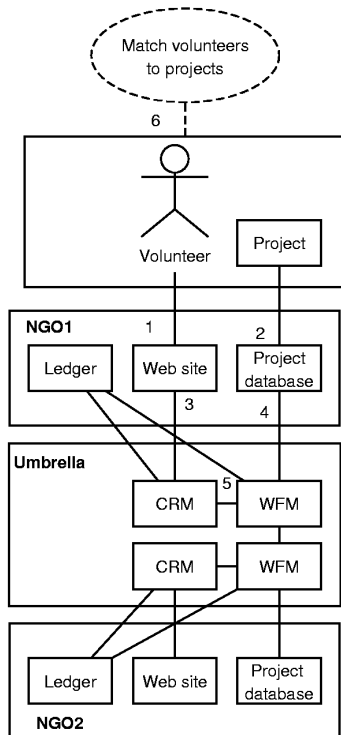3. and the CRM system obtains the volunteer data from the web sites of all NGOs

Figure 8. ICT outsourcing solution software diagram.

Figure 9. Business process outsourcing solution software diagram.

4. and the WFM system can access the project data from the project database of all NGOs

5. and can access volunteer data from the CRM system

6. then the WFM system must match volunteer and project data as required.

Although this looks very similar to the ICT outsourcing correctness argument, in this case there is one WFM that must interface with many project databases, where in the ICT outsourcing solution there are many WFMS each interfacing with one project database. This yields different required properties of the WFM.

Note also that in both solutions, the solution diagrams make clear that the CRM and WFM must interface with the ledger business systems of many NGO's to support other requirements (not discussed here) stemming from other business processes. This will add to the cost of either solution. It will also require different competencies from the people operating and the people managing the ledger systems.
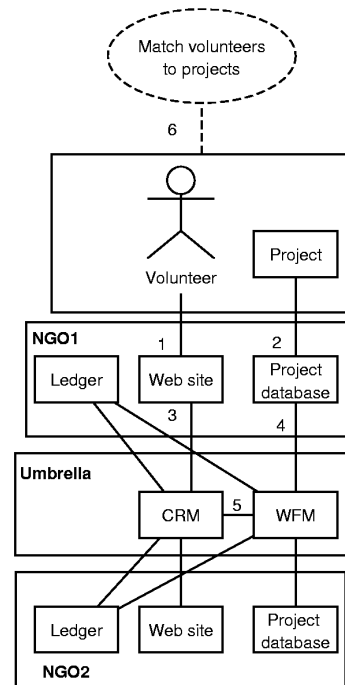
7 Discussion and Conclusions

We dinstinguished three problem-solving levels for business system specification in networked enterprises:

- The value level, where we the business designer is concerned with value activities and value exchanges.

- The process level, where the process designer is concerned with operationalizing the value exchanges and activities identified in the value model.

- The business system level, where the requirements engineer is concerned with specifying the business systems that support business processes.

Value models are used as problem diagrams at the value level. The role played by shared phenomena in problem diagrams, is played by value exchanges in the value models. Value models are operationalized in process models and these are a source of requirements for business systems that support these processes.

Value models have the following heuristic roles in software problem framing.

- Value models help us identify the core, economically oriented, exchanges in a networked business and this

9

helps us identify the requirements to be satisfied by business systems.

- Value models help us identify the actors involved in a network business process, and this helps us identify both the relevant business systems in a proposed business solution, and relevant domains in this business solution.

It remains to be seen whether value modeling helps us *framing* software problems. For example, are there classes of value models that lead to classes of software problems? Conversely, can the known catalog of software problem frames help us structuring a business solution diagram?

Another line of investigation arises from the fact that participating business are generally not eager to develop new business systems for implementing a networked enterprise. So the components of business solution diagrams such as figures 9 and 8 may very well be given. The question then is how we can assemble and configure existing components into a solution that supports the desired business processes. Since the concern remains to satisfy the requirements, can we still find a correctness argument for the configured business systems?

Our approach is related to goal-oriented approaches such as KAOS [8] because it identifies business goals to be achived by software. Also, like KAOS, we do not distinguish the systems to be specified from the systems given to us until we are at a level where we can specify desired properties of these systems. A major diffence is that KAOS goals are based on temporal logic whereas our goals are based on a commercial value analysis.

In the Tropos approach [7], business structures as those identified by Mintzberg are formalized and used as input for a goal-oriented requirements process using i*. We do not formalize business structures but use a simple technique to visualize actors and value exchanges in a network, without starting from a library of business structures. Also, we arrive at software requirements using existing process modeling techniques such as UML activity diagrams [9].

Cox and Phalp [2] discuss how a business process model can be transformed into a problem frame. We work at a higher level of abstraction because we work with value models, that do not represent a business process. Elaboration into a process model is part of the implementation of a value model.

This paper is a proposal for a line of investigation into the relation between value engineering and software requirements engineering for networked business systems, and it raises more questions than it answers. We foresee the following research questions.

- Can this technique really work in practice? We intend to study this by action research.

- Can we find patterns in value models that lead to classes of software problems? Since the value models of networked enterprises are ICT-enabled, there should be a close relation between the two kinds of models. We intend to study this using case study research.

- Can the known catalog of software problem frames help us structuring a business solution diagram? We intend to study this by applying the known problem frames on a number of networked enterprises.

- How does this approach work in a situation where software is not developed but configured from existing components?

References

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

[2] K. Cox and K. Phalp. From process model to problem frame – a position paper. In C. S.-B. Achour, B. Regnell, and E. Kamsties, editors, *9th International Workshop on Requirements Engineering – Foundations for Software Quality (REFSQ'03)*, pages 93–96. Pre-Proceedings, www.refsq.org, 2003.

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

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

[5] J. Gordijn, J. M. Akkermans, and J. C. van Vliet. Business modelling is not process modelling. In S. W. Liddle and H. C. Mayr, editors, *Conceptual Modeling for E-Business and the Web*, volume 1921 of *LNCS*, pages 40–51, Berlin, D, 2000. Springer Verlag. Also available from http://www.cs.vu.nl/~gordijn/.

[6] M. Jackson. *Problem Frames: Analysing and Structuring Software Development Problems*. Addison-Wesley, 2000.

[7] M. Kolp, P. Giorgini, and J. Mylopoulos. Organizational patterns for early requirements analysis. In *Conference on Advanced Information System Engineering (CAiSE 03)*. Springer, 2003. LNCS 2681.

[8] A. v. Lamsweerde, R. Darimont, and E. Letier. Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926, November 1998.

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

[10] R. Wieringa. *Design Methods for Reactive Systems: Yourdon, Statemate and the UML*. Morgan Kaufmann, 2003.