Lines in the Water

The Line of Reasoning in an Enterprise Engineering Case Study from the Public Sector

H.A. Erik Proper[1,2] and M. Op 't Land[3,4,5]

[1] Public Research Centre – Henri Tudor, Luxembourg
[2] Radboud University Nijmegen, Nijmegen, The Netherlands
[3] Capgemini, Utrecht, The Netherlands
[4] Technical University of Lisbon, Lisbon, Portugal
[5] Delft University of Technology, Delft, The Netherlands

Abstract. Present day enterprises face many challenges, including mergers, acquisitions, technological innovations and the quest for new business models. These developments pose several fundamental design challenges to enterprises. We take the perspective that the design of an enterprise essentially involves a rational process that is driven by creativity and communication. Being a rational process means that there should be some underlying line of reasoning in terms of verifiable assumptions about the environment, the requirements that should be met, trade-offs with regards to the alignment between different aspects of the enterprise, et cetera, while all being used to motivate scoping and design decisions.

The core driver for the research reported in this paper is the desire to better understand the line of reasoning as it is used in real-life enterprise engineering / architecture engagements. By documenting and studying the lines of reasoning followed in different cases, we aim to gain more insight into the actual reasoning followed in practical situations. This insight can then, for example, be contrasted to the line of reasoning as suggested by existing enterprise engineering / architecture methods, and more importantly be used to create more effective lines of reasoning in future cases.

The larger part of this paper is therefore dedicated to a discussion of a real-life case from the public sector, where we focus on the line of reasoning followed in this case. The case concerns a large transformation program in the context of *Rijkswaterstaat*, which is an agency of the Dutch Ministry of Transport, Public Works and Water Management. In discussing this case study, we will focus on the line of reasoning as it was actually followed by Rijkswaterstaat, while also briefly discussing some of the results that have been produced 'along the way'.

To be able to position / track the line of reasoning as it was followed in the case study, we also provide six possible *reasoning dimensions* along which we expect the line of reasoning to propagate. For each of these *reasoning dimensions* we will argue why it would be relevant to study its occurrence / use in real-life cases. When combined, these reasoning dimensions form a kind of a *reasoning map* for enterprise engineering / architecture. This map provides us with a basic a-priori understanding of the dimensions along which the line of reasoning followed in a specific case study may propagate. In discussing the Rijkswaterstaat case study we will indeed position the followed line of reasoning in relation to this *reasoning map*.

F. Harmsen et al. (Eds.): PRET 2010, LNBIP 69, pp. 193–216, 2010.

1 Introduction

Present day enterprises face many challenges, such as the recent economic turmoil, mergers, acquisitions, technological innovations, deregulation of international trade, privatisation of state owned companies and agencies, increased global competition and the quest for new business models. These changes are fuelled even more by the advances in eCommerce, Networked Business, Virtual Enterprises, Mashup Corporations, the availability of resources on a global scale, et cetera [1, 2, 3, 4]. Such factors all contribute towards an increasingly dynamic environment in which enterprises aim to thrive.

As a result, enterprises need to be agile to improve their chances of survival. In other words, they need the ability to quickly adapt themselves to changes in their environment, and seize opportunities as they avail themselves. Such agility has become a business requirement in many lines of business, from the defence industry (schedules for combat systems have shrunk from 8 to only 2 years) via the car industry (from concept to production, for a new model in a few months instead of 6 years) to the banking industry (time to market for a new product from 9–12 months to a few weeks).

These trends also trigger enterprises to re-structure themselves into specialised parts increasing the agility of the organisation as a whole [5, 6, 7]. Traditional fixed organisational structures are replaced by more dynamic *networked enterprises* [8, 9, 10, 11]. Such trends are certainly not limited to the private sector alone. Citizens and companies increasingly expect governments to operate more effectively and efficiently. This triggers governments to restructure the way they operate into more "customer focussed" agencies, while privatising executive agencies.

The above discussed developments pose fundamental design challenges to enterprises. For example, deliberate decisions have to be made on the division of tasks and responsibilities in networked enterprises, including topics such as business process outsourcing (BPO), and the use of shared-service centres (SSC) and cloud computing for (IT) services.

We take the perspective that at the core, the design of an enterprise (from its value propositions, via its business processes to its supporting IT) is a rational process which is driven by creativity and communication [12]. The American Engineers' Council for Professional Development [13] also refers to the duality between creativity and rationality by stating that engineering concerns "*the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilising them ...*". In other words, there should be some underlying line of reasoning in terms of verifiable assumptions about the environment, the requirements that should be met, trade-offs with regards to the alignment between different aspects of the enterprise, et cetera, while all being used to motivate scoping and design decisions. Some authors even compare the design of a system (such as an enterprise) to the *creation* of a mathematical proof [14].

In past real-world experiences [15, 16, 17, 18, 19, 20], we have also found that when (re)designing enterprises (and their IT support) it is highly beneficial to make a clear and fundamental distinction between core aspects, such as stakeholder motivations, desired functionalities, implementation independent construction, the actual implementation, system types, et cetera, but also to trace the line of reasoning that seems to flow between these aspects.

The line of reasoning that underpins the design of a (part of an) enterprise, may be constructed *a-priori* or *a-posteriori*. In the *a-priori* case, design decisions are constructed rationally in the sense that they are based purely on rational conjectures. In the *a-posteriori* case, design decisions are essentially made first and are then motivated or tested (in terms of their falsifiability). In sum, we think it is fair to state that when enterprises are (re)designed, the designers will construct an a-priori or a-posteriori *line of reasoning* to motivate the resulting design.

Our underlying *research* driver is the desire to better understand the line of reasoning as it is used in real world enterprise engineering / architecture engagements. Our underlying *practical* driver is the desire to further professionalise the field of enterprise engineering. By documenting and studying the lines of reasoning used in different cases, we aim to gain more insight into the actual reasoning followed in practical situations. This insight can then, for example, be contrasted to the line of reasoning as suggested by existing enterprise engineering / architecture methods, and more importantly be used to create more effective lines of reasoning in future cases.

This paper will therefore start (in Section 2) by discussing six possible dimensions that may be followed by the line of reasoning. For each of these *reasoning dimensions* we will argue why it would be relevant to study its occurrence / use in the real world. When combined, these reasoning dimensions essentially form a *reasoning map* for enterprise engineering. This map provides us with a basic a-priori understanding of the dimensions along which the line of reasoning followed in a specific case study may propagate. The map, however, is by no means intended as an a-priori limitation of the reasoning dimensions we will look for in future case-studies. As we conduct more case studies, we also expect the reasoning map to evolve.

The larger part of this paper is dedicated to the discussion of a case study from the public sector. The case study concerns a large transformation program in the context of *Rijkswaterstaat*, which is an agency of the Dutch Ministry of Transport, Public Works and Water Management. The *Rijkswaterstaat* agency (more details to follow in Section 3) is responsible for the construction, management, development and maintenance of the main infrastructure networks in the Netherlands. Needless to say that we can only touch upon some of the highlights of this case, since it involves a multi-year transformation program at the Rijkswaterstaat agency. In discussing this case study, we focus on the line of reasoning as it was actually followed in the case, most notably in terms of its so-called *DAShboard* (Section 4), while also briefly discussing some of the results that have been produced 'along the way' (Section 5).

2 The Reasoning Map

In this section we discuss the six *reasoning dimensions* in which we have an a-priori interest when investigating the line of reasoning that is followed in case studies. When combining these *reasoning dimensions*, they provide a *reasoning map* upon which the line of reasoning, as it is used in specific cases, can indeed be mapped.

Please note that the *reasoning map* should not be confused with an (attempt to create yet another) enterprise architecture / engineering framework. We also do not claim the dimensions as included in the reasoning map to be 'complete' in any way. They are

purely intended as a starting point to 'make sense' out of the lines of reasoning followed in specific cases. We will, however, for each of the six *reasoning dimensions* motivate why we consider it to be relevant to study how the line of reasoning in specific cases propagates, while also arguing the potential added value of making the reasoning in this dimension explicit. This also implies that we will only focus on dimensions which indeed support the creation of a line of reasoning. Over time, and based on multiple case studies, we may draw the conclusion that certain dimensions are missing from existing architecture / engineering frameworks, or conversely, that certain reasoning dimensions are not used much in practice. The latter would suggest that these dimensions are less important or even superfluous.

2.1 Design Motivation

We regard an enterprise as a goal-oriented cooperative of people and means. This is in line with common definitions of organisation, e.g. "*organisations are (1) social entities, (2) directed towards a goal, (3) designed as systems of consciously structured and coordinated activities, and (4) connected with the external environment*" [21]. When indeed taking the view than an enterprise is a goal-oriented cooperative, we regard it as relevant to see if a goal-oriented *design motivation* dimension is present in the line of reasoning followed by real world cases.

In this reasoning dimension, we currently make a distinction between *motivation*, *requirements* and *design*. In the field of software engineering, this dimension comes mainly to the fore from the field of goal-oriented requirements engineering [22,23,24,25]. The *motivation* is captured in terms of the goals of stakeholders, which provide the underpinning of the *requirements* that should be met by the *construction* of the system. We currently expect these motivations to involve (at least) four classes of goals:

1. What are the goals of stakeholders for owning / having the system?
2. What are the goals of stakeholders for transforming the system?
3. What are the goals of actors for playing a role in the system?
4. What are the goals of actors for using the system?

Note: with system we refer to the general systems theory's notion of system. We are not using it as a synonym to software application, as it seems to have become common place among IT people. Software applications indeed are systems, but enterprises and information systems are systems too.

The requirements can pertain to the services/functions offered by the system being designed, the qualities of these services with regards to their delivery (e.g. availability and security), qualities pertaining to their upkeep (e.g. costs and maintainability), as well as qualities pertaining to their change (e.g. flexibility and scalability).

In the field of enterprise architecture, one typically makes a distinction between architecture principles and the actual 'architectural design' [26,27,28,29,30,31]. As discussed in [32], architecture principles are normative principles that limit the *design freedom* and provide a first translation of stakeholder's requirements towards a focussing of the design space. Therefore, normative principles take the form of *declarative statements* on essential properties of the system. This also implies the need for statements

that provide more tangible guidance to the implementers, as well as allow for analysis of the design to assess whether (in particular qualitative) requirements are met. In other words, instructive statements which more tangibly express *how* the system is to be constructed, e.g. in terms of value exchanges, transactions, services, contracts, processes, components, objects, building blocks, et cetera. These instructive statements can then be used to represent the 'architectural design' of a system. In [32] it is proposed to refer to these statements as *instructions* since they tell designers specifically what to do and what not to do. This use of the word *instruction* also concurs with its definition from the dictionary [33]: *an outline or manual of (technical) procedure* Enterprises typically use models expressed in languages such as UML [34], BPMN [35], TOGAF's [30] content framework, ArchiMate [36], or the language suggested by the DEMO method [37], (as a base) to express such *instructions*.

In line with the commonly made distinction between architecture principles and the actual architectural design, we are therefore also interested to follow the line of reasoning between these two levels of design. In sum, the design motivation dimension therefore distinguishes: *motivation*, *requirements*, *normative design* and *instructive design*.

2.2 Implementation Abstraction

Traditionally, the field of information systems engineering [38, 39, 40] uses a distinction between an implementation free design of the information system, referred to as the *conceptual model* and one or two (increasingly) implementation dependent levels, usually referred to as the *logical* and *physical levels*. The Zachman framework [41] for *information systems* architecture (i.e. not for enterprise architecture) also reflects this distinction in terms of a conceptual level in terms of a business model from the perspective of the owner of the information system, a logical level in terms of an (information) systems model from the perspective of the designer, and a physical level in terms of technology model from the perspective of the builder.

In the field of software engineering a similar distinction has emerged in the context of MDA (Model-Driven Architecture) [42], where a distinction is made between a Computation-Independent Model (CIM) which essentially provides an (information technology) implementation independent view on a domain, Platform-Independent Model (PIM) describing the implementation in terms of behaviour and structure of applications regardless of the chosen (information technology) implementation platform, and a Platform-Specific Model (PSM) which contains all required information regarding a specific platform that developers may use to implement the executable code.

In the field of enterprise architecture and enterprise engineering we also see the suggestion to distinguish between an implementation independent level and an implementation dependent level. The Integrated Architecture Framework (IAF), as developed by Capgemini [43], distinguishes a conceptual, logical and physical level. In TOGAF [30] we see the *logical level* represented in terms of so-called *architecture building blocks* and the *physical level* as *solution building blocks*. The DEMO [37] methodology for enterprise engineering identifies an implementation independent level in terms of an *ontological model* of the enterprise, next to its *implementation model*.

From a research perspective we are interested to see if a distinction between an implementation independent level and an implementation dependent level is indeed present in the line of reasoning used in practice. At present we will not make a distinction between a logical and a physical implementation level, as the borderline between the two seems to be rather difficult to make. We will use the terms *essential model* and *implementation model* to refer to the implementation independent and implementation specific model respectively. When considering the two key meanings of the word *essence* as provided by the dictionary [33]:

1. the permanent as contrasted with the accidental element of being,
2. the individual, real, or ultimate nature of a thing especially as opposed to its existence.

we believe that *essential model* best captures the intention of an implementation independent model. Terminology such as conceptual (*of, relating to, or consisting of concepts* [33]) or ontological (*relating to or based upon being or existence* [33]) apply equally well to the implementation independent level as well as the implementation dependent level (which also exists, and can be described in terms of concepts).

Our scope is the design of enterprises. In other words, not just the design of information systems, and most certainly not just the design of computerised information systems. An enterprise is a socio-technical system. In other words, it primarily consists of human actors that are supported by different forms of technologies (including information technology). We take the stance that the role of technology in an enterprise is always supportive. More specifically, technology can never be (legally, morally, ethically, et cetera) responsible for its own actions. Whichever the level of technological support, human beings remain responsible. When an ATM at a bank 'thinks' it should not issue money to us, we might sometimes express our frustrations by 'vandalising' this piece of technology. However, human beings remain responsible for specifying the business rules used by the ATM's software to determine that it will not give us money. In that sense we should really be 'vandalising' them. For example, in DEMO [37] this is made explicit by stating that an enterprise is primarily being a social systems in which the core elements are social individuals, where the operating principle is the fact that the constituent social individuals enter into and comply with commitments regarding the products or services to be created or delivered.

When regarding an enterprise as consisting of human actors supported by technology, one does need a refined view on the notion of *implementation model*. The implementation of the *essential model* of an enterprise involves the "implementation" of responsibilities in terms of human actors and the implementation of technologies that support these human actors in their responsibilities. In that sense, the ATM of a bank supports cashiers in their responsibility of issue cash to clients. The implementation in terms of responsibilities for human actors is what we will refer to as the *social implementation* and the implementation in terms of the underlying technologies as the *technological implementation*. The *social implementation* focuses on the division of the essential tasks and responsibilities identified at the ontological level among human actors, while the *technological implementation* then identifies the technological means that can be used by these human actors to support them in their tasks and responsibilities.

We expect that identifying a *social implementation* level also invites designers of enterprises to carefully think about the impact of their design decisions on the well being of the human actors. In doing so, one would expect designers to also take properties such as work load, ethical burden, cognitive load, et cetera, into consideration in weighing between design alternatives. To illustrate this point, consider a small example in terms of a Pizza delivery service. At an ontological level (see e.g. [28]), the driving transaction of a Pizza delivery service is the ordering and delivering of a Pizza. In the essential model, this corresponds to a single transaction *complete Pizza purchase* involving an actor (role) *customer* and *Pizza order completer*. At the essential level, the costumer requests a Pizza, while the completer delivers the Pizza.

Now consider the following commonly used social implementation of this essential model. The taking of the Pizza order is done by some *order taker* functionary who answers phone calls. The Pizza is then baked by a functionary *baker*, and delivered by a functionary *deliverer*. This means that the essential transaction of *complete Pizza purchase* as it is performed by the single (essential) actor *Pizza order completer*, is actually implemented in terms of three functionary types. And indeed, in most practical situations this is highly defendable. It allows for an efficient use of time, technological means and furthermore enables an effective build up of specialised skills. The disadvantage of this implementation is, however, is the commitment to deliver (the right!) Pizza to the customer is done by another functionary than the functionary who actually delivers the Pizza. As a consequence, when the wrong Pizza is produced, or the Pizza is baked incorrectly, the Pizza deliverer is confronted with the angry customer while the deliverer can hardly influence these qualities. When this happens often, this is likely to lead to stress and general negative feelings with those who execute the functionary role of Pizza deliverer.

An alternative social implementation would be to only have a *Pizza order completer* functionary. As a consequence, the person who takes the order, would also be the one baking your Pizza, and then driving out to your house to personally deliver the Pizza. This is likely to be inefficient from a time and resource perspective. Nevertheless, in the case of failures, only the directly responsible person is confronted with the complaints / angry client. A hybrid implementation would, for example, be to indeed use the first division into multiple functionary types, but to ensure regular job rotation among the human beings who execute the different functionary types.

Even though we have theoretical and ethical reasons to expect / want to find a distinction between a social and a technological implementation, we expect to find little of such a distinction in practice. We base this scepticism mainly on the lack of such a distinction in existing architecture frameworks and engineering frameworks. In our view, most (if not all) of these frameworks take a technology-minded perspective on the implementation model. So, even if a distinction is made between an essential level and an implementation level, the latter is mainly focusing on the technology implementation. Nevertheless, we hope to find at least 'traces' of taking properties such as work load, ethical burden, cognitive load, et cetera, into consideration in weighing between design alternatives.

In sum, the implementation abstraction dimension therefore distinguishes: the *essential level*, the *social implementation level*, and the *technological implementation level*.

2.3 Construction Abstraction

In the engineering of systems in general, a well known distinction is the one between a *black box* and a *white box* perspective[1]. Typically, from a *black box* perspective, one regards a system (such as an IT system, an information system, or an enterprise) solely in terms of its input, output and transfer characteristics without any knowledge of its internal workings. In other words, its internal construction is "opaque" (black). The opposite of a *black box* perspective on a system, is the *white box* perspective where the inner construction of the system can indeed be observed.

For example, in the DEMO [37] methodology for enterprise engineering, this distinction is made explicit in terms of the *function* and *construction* perspectives. In the ArchiMate [36] standard, this distinction comes to the fore in terms of the *internal* and *external* perspectives.

The construction abstraction dimension, therefore distinguishes: the *black-box perspective* and the *white-box perspective*.

2.4 System Types

We are interested in knowing the system types that are being used in specific cases, as well as how these system types are linked. For example, the DEMO [37] methodology identifies a *B-organisation*, *I-organisation* and *D-organisation*, representing a system type focussing on business processes, information processing and data processing respectively. AchiMate [36] distinguishes a *business layer*, *application layer* and *technology layer*, representing system types focussing on business processes, computerised information processing, and the underlying IT infrastructure, respectively. Similarly, the Integrated Architecture Framework [43] (also used as a base in the Rijkswaterstaat case) distinguishes between business activities, information (processing) needed for the business, computerised information systems, and the technology infrastructure needed for these latter systems.

This reasoning dimension does not have a pre-defined set of values. There seems to be a general understanding in (IT focussed) approaches that there is a general *business system* level that uses *computerised information systems*, which on their turn depend on underlying *infrastructure systems*. However, since the field of enterprise architecture / engineering increasingly moves beyond the IT centric focus, we think it is not wise to a-priori fix the values in this dimension, and rather observe the values used in practice.

2.5 Design Evolution

Enterprises are hardly ever created from scratch. In other words, one has to deal with existing products, processes, information systems, et cetera. This also implies that when design a new step in the evolution of an enterprise, one cannot just look at the future 'version' of the enterprise in isolation. One has to consider the existing situation (and its past) to understand / rationalise some of the design decisions underlying the next steps in the evolution of the enterprise.

[1] See e.g. `http://en.wikipedia.org/wiki/Black_box_(systems)`.

The TOGAF methodology [30] traditionally makes a distinction between a *baseline architecture* and a *target architecture*. Other sources may refer to the *ist* and *soll* situation. More recently, TOGAF also introduced the concept of *transition architecture* to allow for the fact that a transformation from the baseline situation towards the target situation, is likely to involve multiple steps (or plateaux).

Similarly to the system types, we currently do not provide a pre-defined set of values. We are also interested in observing the kinds of values used in real case studies. Generally, however, one would expect three types of values: (1) *initial state*, i.e. the design of the enterprise at the start of a (proposed) transformation, (2) *intermediary states*, several intermediary states in terms of e.g. plateaux and (3) *final state*, the design of the future enterprise as it is currently envisaged.

Within this dimension we are also interested to see if real-world cases indeed explicitly use knowledge of the existing situation to support design decisions on the final state. Even more, it will be interesting to see if the intermediary states will be motivated in terms of a gap analysis between the initiate state and final state, arguing how this gap will be bridged in a series of intermediary steps. Since the execution of enterprise transformations tend to take longer periods of time, it is also interesting to see if in the identification of intermediary states, one has taken into consideration that during the execution of an actual transformation the final state will change due to new requirements.

2.6 Design Horizon

This final dimension takes into account that in engineering / architecting an enterprise one can take a short term or a long term perspective. More specifically, it seems (a-priori) reasonable to distinguish between three levels:

Strategical design horizon – This is the level of the enterprise's strategy, including sub-strategies dealing with business aspects, human resourcing issues, IT, et cetera.

Tactical design horizon – This is the level at which the enterprise's strategy is made more concrete in terms of general requirements on, design principles for and high level designs of, (classes of) sub-systems within the enterprise, as well as the identification of programmes needed to execute a proposed transformations in terms of changes / creation of the identified sub-systems.

Operational design horizon – At this level, we are at the level of the design of specific sub-systems of the enterprise, projects filling in the enterprise transformation, et cetera.

The *tactical design horizon* might be referred to as the *architecture* level, while the *operational design horizon* might be called the traditional *design* level. However, to avoid confusion with our more general use of the word design, and to acknowledge the fact that in general the distinction between strategy, architecture and design is still open to debate, we refrain from using these words. At the same time, we do not claim that the strategical / tactical / operational distinction solves this. However, these terms do enable a more neutral observation of what happened in a specific case, separate from the discussion if the way a specific case used the term architecture is indeed correct from a specific definition of architecture.

3 Rijkswaterstaat and the Berthing-Place Domain

The case study reported on in this paper, was conducted at the agency Rijkswaterstaat (RWS), which is the Directorate-General for Public Works and Water Management in the Netherlands. Under the command of a departmental Minister and State Secretary, RWS is responsible for the construction, management, development and maintenance of some of the main infrastructure networks in the Netherlands, namely the networks for the transportation of water, road traffic and water traffic. Since a significant part of the Dutch economy is directly dependent on the logistics sector, including transport of goods from / to the Rotterdam and Amsterdam ports, as well as from/to the Schiphol airport, RWS has a very important role to play in the Netherlands.

In addition to the road and railway[2] networks, an important part of the logistical infrastructure in the Netherlands are its waterways. Several rivers run through the Netherlands, allowing goods to be transported by ships from / to Belgium, Germany and beyond. These rivers are also connected by several major canals. To regulate both the water flow and shipping traffic, several locks have been put in place. RWS is responsible for the maintenance and management of these waterways and associated infrastructure. A further responsibility of this executive agency is the management of the network of dikes, dams, and other means needed to keep the Netherlands from flooding. Needless to say that to a country which is positioned largely below sea level this is a task of some importance, especially when this is combined with one of the busiest network of waterways in the world.

The case which we focus on in this paper is concerned with a specific aspect of *Shipping Traffic Management* (SVM; In Dutch: ScheepvaartVerkeersManagement), namely the use of berthing places. A berth place is an area where a ship can dock; essentially a "parking spot" for ships. Some of these berths are used as holding areas at busy locks or bridges, while others are used for ships and their crews to stay overnight, et cetera. Not all berthing places are owned by the central government. A large number of them are owned by (e.g. municipal or provincial) port administrations, companies with their own ports and associated berthing places, et cetera.

Having good and sufficient berthing places is essential for efficiency and safety on the Dutch water ways. E.g. skippers not only need to take rest, but they are also lawfully obliged to do so. RWS is required to facilitate this. Failing to do so may lead to delays on the waterways and may even increase the risk of accidents. A survey held by RWS before the start of this case-study revealed that the users of berthing places were not satisfied with the amount, quality and location of berthing places. At the same time, the financial aspects should not be underestimated. Berthing places are quite costly to create and to maintain. Therefore, 'simply' creating more berthing places is also not the answer. In other words, proper management is needed, carefully weighing the needs in terms of efficiency and safety of shipping traffic, the interests of the skippers, and the financial aspects of creating and maintaining berthing places.

To better support the SVM activities, including the regulation of the use of berthing places, RWS initiated a long term transformation program involving the creating (and realisation) of a *Domain Architecture SVM* (DAS; In Dutch: DomeinArchitectuur

[2] The management of the Dutch railway network is not part of RWS's responsibilities.

Scheepvaartverkeersmanagement). The design of an improved management system with associated procedures and IT support, is part of this transformation program.

4 The Rijkswaterstaat *DAShboard*

Large organisations such as Rijkswaterstaat (RWS) continually change their products and services, quite often in close cooperation with other value-chain partners. Such changes in value propositions are likely to have a deep impact on the structures and operations of the organisation as well as its supporting IT. To timely, coherently and consistently govern the transformation processes required to implement such large changes, RWS applies two main instruments: (1) a standardised process for Integrated Governance and (2) a reasoning framework, called the *DAShboard* [44]. The first instrument prescribes RWS's standard approach to move from an initial idea via an architectural exploration and analysis of alternatives, to annual change plans. These latter plans including the way in which stakeholders should be involved and decisions should be taken. In the remainder of this section we will concentrate on the second instrument, the DAShboard. First we will introduce its roots and underlying concepts, relating it to some of Section 2's *reasoning dimensions*, then we will present its contents.

The DAShboard is the standard reasoning framework of RWS, as applied by the *D*omain *A*rchitecture *S*VM; hence the nickname *DAS*hboard. It builds primarily on the Integrated Architecture Framework (IAF) [43], from which it borrows its two main axes: Aspect Areas and Abstraction Levels, as well as the notion of the Artefacts. Compared with IAF three changes have been made: (1) a Transformation Level is added, (2) the Aspect Area "Information" is split into "Information delivery" and "Data Management" and (3) instead of listing Artefacts the DAShboards sums up a number of key *questions*. The Artefacts as defined in IAF can certainly be used to provide an answer to these questions, where the listed questions can provide a clearer focus enabling a more conscious selection of those Artefacts that are the most effective for answering these questions. Table 1 shows the DAShboard, made specific for the "Berthing places" case.

The DAShboard discerns five Abstraction Levels, which allow problems to be split into separate aspects, enabling a stepwise solution:

- *Contextual level*, answering the "Why" question, such as drivers, objectives, principles and scope;
- *Conceptual level*, answering the "What" question, what are the requirements , what services should the solution deliver;
- *Logical level*, answering the "How" question with an "ideal" solution;
- *Physical level*, answering the "With what" question with physical means: people, organisations,
- *Transformation level*, answering the "When" question by providing a transformation path from AS IS to TO BE, and its underpinning by a Business Case.

Compared with Section 2's *reasoning dimensions*, the Abstraction Levels used by RWS correspond to multiple dimensions at the same time. More specifically:

- The *contextual*, *conceptual*, *logical* and *physical* levels together cover the *design motivation* dimension.

Table 1. DAShboard, with questions for the case "Berthing places Management (BpM)"

Contextual				
What are the goals of BpM? Who are the internal and external stakeholders for BpM and what are their interests and requirements? What types of Berthing places can be discerned, and how are they used?	Which laws and regulations are applicable for, or influencing, BpM? What are the main changes in the environment of BpM? What policies exist concerning the Business/Information, Applications and Infrastructure in the area of BpM?		Which principles and standards are applicable to BpM? Which running programmes / projects are influencing BpM?	
Business	*Information*		*Applications*	*Technology Infrastructure*
	Information supply	*Data management*		
Conceptual				
What business services does BpM supply and use? What is the required quality of these business services? What business actors deliver these services, and which business actors and need these services?	What information is used by the business actors? What is the required quality of the information services? What information actors deliver these services?	What data are used by the information actors? What business actors do create the original facts? What is the required quality of the data services? What data actors deliver these data services?	What application services support the business, information and data actors? What is the required quality of these application services? What application component deliver these application services?	What infrastructural services do support the business, information and data actors and the application components? What is the required quality of these infrastructural services? What infrastructural component deliver these infrastructural services?
Logical				
Which business objects are observed or changed when delivering the business services? How do the processes of the business actors operate?	How are the information products composed by data objects? How do the processes of the information actors operate?	How are the data objects composed, and which states of the business objects do they concern? In which way do the processes of the data actors operate?	How are the application interfaces structured? In which way do the application components operate, what are their mutual interactions, and what is the interaction with human actors?	How are the infrastructural interfaces structured? In which way do the infrastructural components operate, what are their mutual interactions, and what are the interaction with human actors?
Physical				
With what people and means are the business actors implemented, and on what locations? What are the operational costs of this implementation?	With what people and means are the information actors implemented, and on what locations? What are the operational costs of this implementation?	With what people and means are the data actors implemented, and on what locations? What are the operational costs of this implementation?	With what software products have the application components been implemented, and on what locations? What are the operational costs of this implementation?	With what infrastructural products have the infrastructural components been implemented, and on what locations? What are the operational costs of this implementation?
Transformational				
What are the differences between AS IS and TO BE for the business?	What are the differences between AS IS and TO BE for information supply?	What are the differences between AS IS and TO BE for data management?	What are the differences between AS IS and TO BE for the applications?	What are the differences between AS IS and TO BE for the infrastructure?
What is the change plan for the transformation of Business, Information Supply, Data Management, Applications and Infrastructure?				

The *contextual level* provides the *motivation*, while also more explicitly adding the notion of scope. The *conceptual level* corresponds to the (functional) *requirements*. The *logical* and *physical* levels together cover the *design* where no explicit distinction is made between a *normative* and an *instructive* design.

- The distinction between the *conceptual* and *logical* levels also corresponds to the distinction between the *black box* and *white box* perspective, as described in the *construction abstraction* dimension.
- The combination of the *conceptual* and *logical* levels, and the *physical* level roughly corresponds to the dimension of *implementation abstraction*, where the *conceptual* and *logical* level provide an implementation independent (i.e. the *essential level*) design (covering both a black-box and white-box perspective), while the *physical* level represents the *implementation level*. No specific distinction is made between a social and a technical implementation.
- *Transformation* contains the dimension *Design evolution*, adding to that the notion of the Business Case.

These findings are summarised in Table 2.

Table 2. The DAShboard's abstraction levels and the reasoning dimensions

DAShboard	Reasoning dimensions			
Abstraction level	Design motivation	Implementation abstraction	Construction abstraction	Design evolution
Contextual	Motivation	All	All	
Conceptual	Requirements	All	Black-box	
Logical	Design	Essential	White-box	
Physical	Design	Implementation	White-box	
Transformational				All

The second dimension in the DAShboard concerns the Aspect Areas —which corresponds to the reasoning dimension *System types*:

- *Business* deals with the creation of new facts in reality, or observing the state of reality; e.g. building a bridge (material new fact), closing a deal (immaterial new fact) or monitoring / judging the state of the road;
- *Information* deals with the creation of information needed by the business for their situational awareness, derived from original or derived facts, including the data management needed for that; e.g. creating information on built bridges per period and region, derived from original facts for each built bridge, on which data have been managed. RWS has split this Aspect Area in two sub-Aspect Areas, namely *Information delivery* and Data management;
- *Applications* provide automated information systems, including their mutual and human interfaces, to support or execute part of Business, Information delivery and Data Management;
- *Technology Infrastructure* provides the ICT infrastructure — such as computing systems, network technology and database management systems — to make the applications work.

In terms of the *design horizon* reasoning dimension, the DAShboard could be used at any of the levels identified. In the SVM case, it was mainly used at the *tactical* and *operational* levels.

The DAShboard provides a systematic analysis of business processes, from their contextual aspects, via their business aspects to their implementation using people and technological means, covering both the present and future states. The general flow of reasoning followed by RWS when using the DAShboard [45], is as follows:

1. One starts by considering the context (i.e. the contextual level). What is the strategy. What are the goals of the organisation? What is the desired direction of the transformation?
2. One then continues with the business aspects. What are the business services offered, and what are the processes needed to deliver these?
3. Then one continues with the identification of the information needed to support these processes. What information is needed? What data needs to be gathered and stored to provide this information?
4. The next step is to asses to what extend the data processing and information supply is / can / should be computerised.
5. Finally, one assess which organisation units and people are responsible for the execution, management, and maintenance of the processes and their IT support, as well as the costs involved.

Answering the questions listed in the DAShboard, also leads to a multiple ambition levels for the transformation of the organisation. In other words, a series of future scenarios with increasing ambitions. DAS also enables a cost / benefits analysis of all aspects of the organisation. This provides RWS with the ability to study the impact of design decisions beforehand, and use as a means to reduce risks when actually transforming the organisation.

RWS is using the DAShboard as a reasoning framework to systematically detect the impact of intended changes. In earlier versions, the cells of the framework were filled with Artefacts, such as "actor model", "use cases", "object model", "business function model". The names of those Artefacts were not clear for managerial purposes, which raised questions such as "*what do I need use cases for?*". Therefore the managerial questions itself were projected in the framework and made specific for the project at hand, enabling conscious choices about which questions to answer and which questions to let go – at least at this stage of decision making. As an example, Table 1 shows the DAShboard, made specific for the case "Berthing places". Generally speaking, "jumping squares" or "moving one diagonal step" in the DAShboard is discouraged; it means "jumping to solutions", which threatens the traceability of the work.

5 Applying the Framework to the Rijkswaterstaat Case

In this section we will discuss *some* of the results that were produced when using the DAShboard in the SVM case of Rijkswaterstaat. We have split the discussion of the results along the DAShboard's abstraction levels, also following the process used by Rijkswaterstaat.

5.1 Contextual

As discussed in [46], two core problems triggered the transformation programme at RWS:

- Berthing place users are unsatisfied about the existing situation (quantity and locations).
- RWS experiences capacity problems on their berthing place due to increasing recreational use, as well as ships avoiding paid alternatives.

Influenceable (by RWS) root causes that have been discerned are:

- Objectification of causes for dissatisfaction and necessity for additional capacity.
- No integrated national policy for management of berthing places (involving RWS and partners)

Berthing places need to be requested and allocated well in time. One emerging task in the SVM-responsibility, therefore, is the use and allocation of berthing places. An important question was also what level / type of management for berthing places is needed. Ideally, this could be solved by introducing a broker between organisations offering berthing places and ship owners needing berthing places. To this end, Rijkswaterstaat considered several solutions for the apparent scarcity of berthing places. One of the solutions could e.g. include a *Shared Service Centre* to accommodate the brokering and allocation tasks. This however would be a rather costly solution.

The original goal for applying the DAShboard for the management of berthing places was therefore:

> *Formulate several solution alternatives for the improvement of the management of berthing places, by Rijkswaterstaat in collaboration with other organisations, and clarify the consequences on the primary processes, information provisioning, data provisioning and IT support.*

A first result in this case study was the context diagram as shown in Figure 1 and the goal tree shown in Figure 2 These diagrams provide an answer to the DAShboard question *What are the goals of BpM? Who are the internal and external stakeholders for BpM and what are their interests and requirements? What types of Berthing places can be discerned, and how are they used?* The context diagram in Figure 1 provides an overview of the key stakeholders of shipping traffic management, while the goal tree included in Figure 2 zooms in on the goals of Rijkswaterstaat pertaining to safe and efficient shipping traffic, and the role of berthing places.

Based on the goals of stakeholders, and the possible impact of different designs of the enterprise, several architecture principles were formulated that address concerns raised by these goals:

- Rijkswaterstaat orchestrates the information flow in the traffic management chain.
- Data is acquired, stored and managed at one location only.
- Information systems facilitate coworkers in their roles and responsibilities.
- The presence of a berthing place is not allowed to decrease efficiency and safety of shipping traffic.

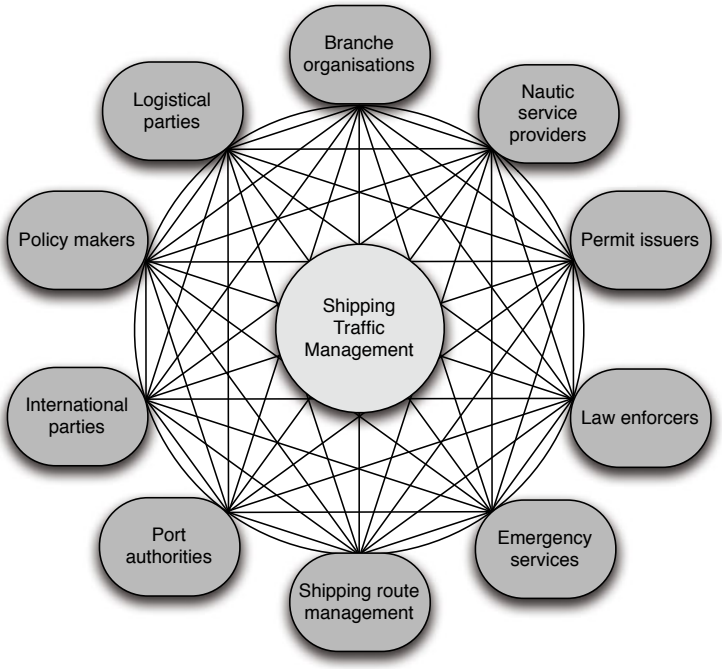

Fig. 1. Context of Shipping Traffic Management

In addition, since the Rijkswaterstaat agency is formally part of the Dutch government, all architecture principles included in the NORA, the Netherlands' Government Reference Architecture [50], apply as well.

5.2 Conceptual

The *function* perspective on the *Shipping Traffic Management* domain, i.e. the *black box* perspective, is provided in Figure 3. This diagram basically is a mind map which lists the functions provided by *Shipping Traffic Management* to its clients, which also shows the satisfaction of stakeholders in the current situation. For example, 40% of the captains of freight ships is satisfied with the number of berthing places available at locks.

5.3 Logical

The core of the logical level (the business aspect) is shown in Figure 4. The depicted construction model corresponds to the *white-box* perspective on core transactions of this domain. The notation used in Figure 4 is the notation for construction models as suggested in the DEMO method from [37]. Core to these models are the actors and the transactions in which they are involved. For example, the transaction labelled T01 leads to the result *berthing place reserved*. It is initiated by actor CA01, the *Berthing place user*, while the request will be met by actor A21, the *Berthing place reservation maker*.

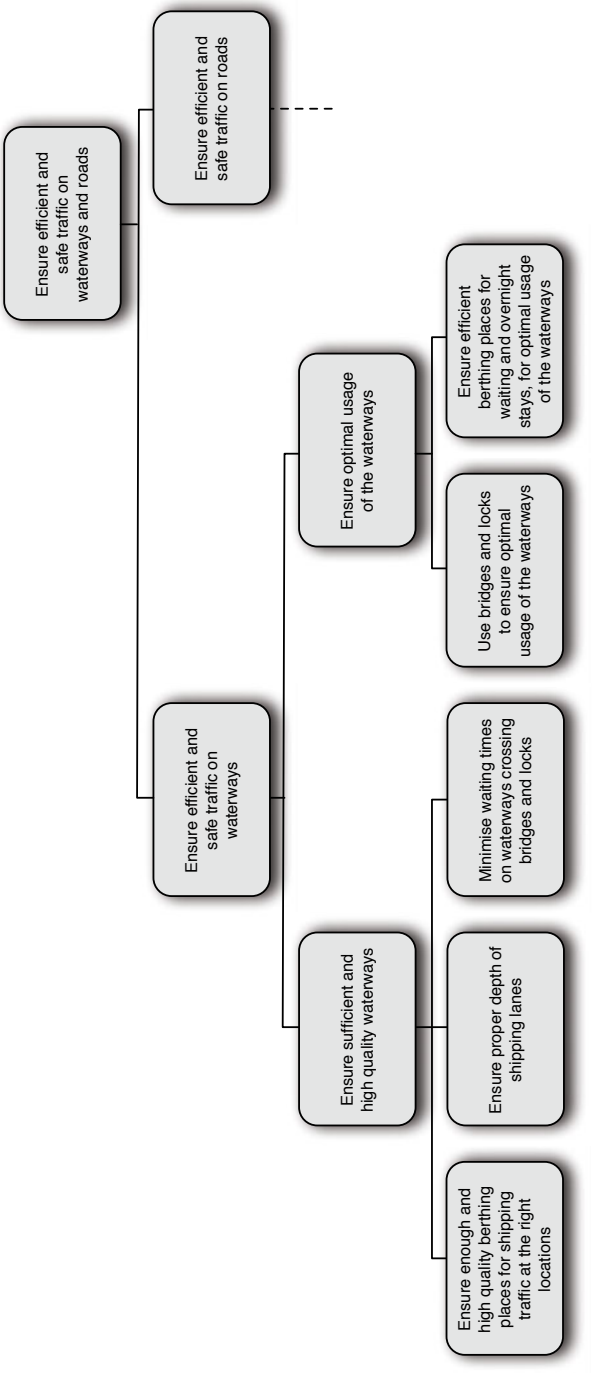

Fig. 2. Goal tree for (part of) the Rijkswaterstaat case

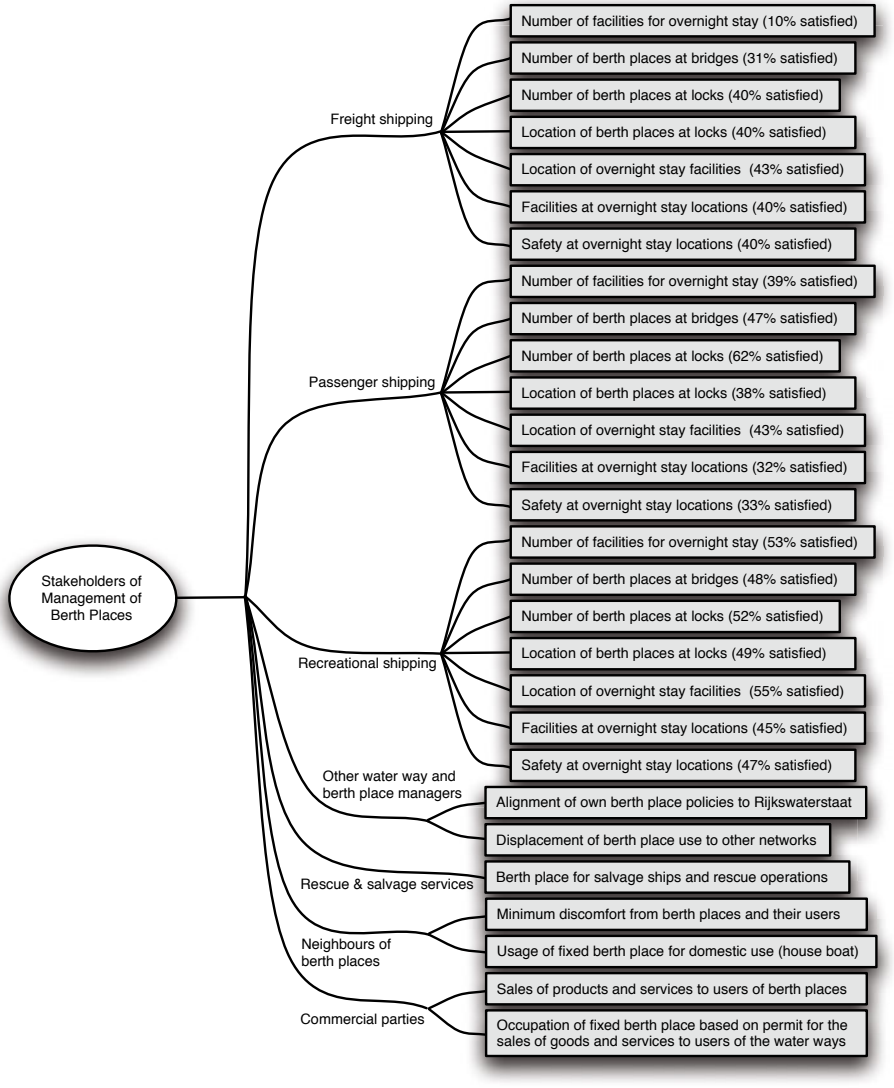

Fig. 3. Function design of the Rijkswaterstaat case

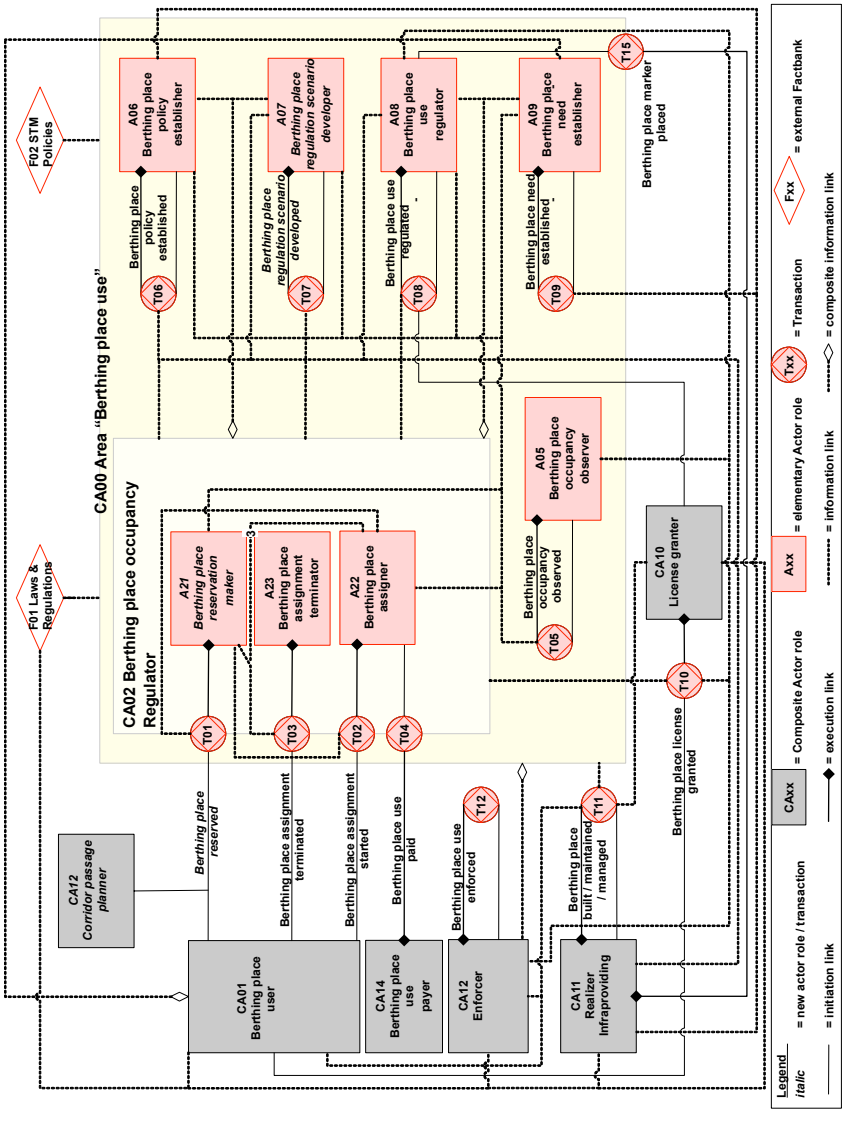

Fig. 4. Example logical level model from the Rijkswaterstaat case

5.4 Physical

In the Rijkswaterstaat case, several applications have been identified that (in the existing situation) support shipping traffic management. Some of these include:

IVS90 – The IVS90 (Information and Tracking System for Shipping Traffic 1990, in Dutch: Informatie- en Volgsysteem voor de Scheepvaart 1990) application, manages the data with regards to the network of waterways and ships making use of these waterways. This includes information pertaining to the ships, their cargo, and planned route. Local waterway administrations also record actual waterway information in this system, including information on berth places. This information is needed by traffic management.

IVS90 is acknowledged to be a mission critical application for the Netherlands as a whole.

BC2000 – The BC2000 (Message Centre 2000, in Dutch: BerichtenCentrum 2000) application, manages and providers nautical and hydrological information with regards to the major rivers within the Netherlands and Europe.

The local offices at bridges and locks, provide key information for shipping (such as "trafic jams", water quality and icing alarms), as well as up-to-date meteorological and hydrological information. This takes place by means of reporting to BC2000, and registration in IVS90, as well as connections to local systems (with attached sensors).

6 Conclusion

The primary focus of this paper was on the discussion of a real-life enterprise engineering case study in the form of a large program at Rijkswaterstaat, an executive agency of the Dutch Ministry of Transport, Public Works and Water Management. In discussing this case study, we traced the line of reasoning used in the case study in terms of a set of earlier defined reasoning dimensions. Central to the Rijkswaterstaat case was the use of the co-called DAShboard, which we also related to the identified reasoning dimensions.

The application of the reasoning dimensions as discussed in Section 2 to the Rijkswaterstaat case, already raised some interesting questions about the dimensions themselves and their scoping. For example:

1. How to link reasoning on classes of systems to reasoning of specific systems. In particular in terms of the link between requirements and architecture principles, in the context of the different levels of the *design horizon*.
2. How does the construction abstraction dimension (*black-box*, *white-box* perspectives) and the design motivation dimension relate? Are they orthogonal, or are the black-box and white-box perspectives just two sub perspectives on the design?

We also regard the discussion of the RWS case study in terms of the reasoning dimensions as a first step in better clarifying and understanding the lines of reasoning followed in real-world cases, understanding the lines of reasoning as suggested by existing enterprise architecture / engineering methods, and gather insight into the modelling languages used to represent the different models and specifications used to represents results in the line of reasoning. As further research activities we therefore see:

1. Use the reasoning map from Section 2 for further a-posteriori documentation of the line of reasoning followed in real-world cases.
2. Study the lines of reasoning suggested by enterprise architecture / engineering methods such as TOGAF [30], ArchiMate [36], GERAM [58], DEMO [37], the Integrated Architecture Framework (IAF) [29], et cetera.
3. Document the use of modelling languages to represent the results in different steps of the line of reasoning. Based on the reasoning map from Section 2, one might expect different languages to be more / less suitable to represent different aspects. In this sense we intend to map "languages" such as i* [47], KAOS [48], SBVR [52], DEMO [37], e3Value [54], BPMN [35], UML [34], ArchiMate [53], et cetera on the reasoning map, based on proven usability from the case studies, as well as the intended purpose of the language.

Acknowledgements

We want to acknowledge Cor Venema, director of Rijkswaterstaat's ScheepvaartVerkeersCentrum (SVC), for his sponsorship for the case "*Ligplaatsenbeheer*", and for generously making available the materials of this project. Also we want to thank all Rijkswaterstaat- and Capgemini experts and team members involved, for their wonderful cooperation and valuable input.

References

1. Tapscott, D.: Digital Economy – Promise and peril in the age of networked intelligence. McGraw-Hill, New York (1996) ISBN-10: 0070633428
2. Hagel III, J., Armstrong, A.: Net Gain – Expanding markets through virtual communities. Harvard Business School Press, Boston (1997)
3. Horan, T.: Digital Places – Building our city of bits. The Urban Land Institute (ULI), Washington (2000) ISBN-10: 0874208459
4. Mulholland, A., Thomas, C., Kurchina, P., Woods, D.: Mashup Corporations - The End of Business as Usual. Evolved Technologist Press, New York (2006) ISBN-13: 9780978921804
5. Hagel III, J., Singer, M.: Unbundling the Corporation. Harvard Business Review (1999)
6. Malone, T.: Making the Decision to Decentralize. Harvard Business School – Working Knowledge for Business Leaders (2004)
7. Galbraith, J.: Designing the Global Corporation. Jossey-Bass, San Fransisco (2000) ISBN-13: 9780787952754
8. Tapscott, D., Ticoll, D., Lowy, A.: Digital Capital: Harnessing the Power of Business Webs. Harvard Business Press, Boston (2000) ISBN-13: 9781578511938
9. Friedman, T.: The World is Flat: A Brief History of the Twenty-first Century. Farrar, Straus and Giroux, New York (2005) ISBN-10: 0374292884
10. Umar, A.: IT infrastructure to enable next generation enterprises. Information Systems Frontiers 7, 217–256 (2005)
11. Gordijn, J., Petit, M., Wieringa, R.: Understanding business strategies of networked value constellations using goal- and value modeling. In: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE 2006), Washington, DC, pp. 126–135. IEEE Computer Society, Los Alamitos (2006) ISBN-10: 0769525555

12. Veldhuijzen van Zanten, G., Hoppenbrouwers, S., Proper, H.: System Development as a Rational Communicative Process. Journal of Systemics, Cybernetics and Informatics 2, 47–51 (2004)
13. The Engineers' Council for Professional Development. Science 94, 456 (1941)
14. Wupper, H.: Design as the Discovery of a Mathematical Theorem: Technical Report CSI–R9729, Radboud University Nijmegen (1997)
15. Arnold, B., Op 't Land, M., Dietz, J.: Effects of an architectural approach to the implementation of shared service centers. In: Second International Workshop on Enterprise, Applications and Services in the Finance Industry (FinanceCom 2005), Regensburg, Germany (2005)
16. Op't Land, M.: Applying Architecture and Ontology to the Splitting and Allying of Enterprises: Problem Definition and Research Approach. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1419–1428. Springer, Heidelberg (2006)
17. Op 't Land, M.: Enterprise architecture, praktische sleutel tot bedrijfsbesturing – case rijkswaterstaat (2007)
18. Op't Land, M., Proper, H.: Impact of Principles on Enterprise Engineering. In: Österle, H., Schelp, J., Winter, R. (eds.) Proceedings of the 15th European Conference on Information Systems, pp. 1965–1976. University of St. Gallen, St. Gallen (2007)
19. Op't Land, M., Dietz, J.: Enterprise ontology based splitting and contracting of organizations. In: Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008), Fortaleza, Ceará, Brazil (2008)
20. Op't Land, M.: Applying Architecture and Ontology in the Splitting and Allying of Enterprises. PhD thesis (2008)
21. Daft, R.: Organization Theory and Design, 9th edn. South-Western College Pub., San Diego (2006) ISBN-10: 0324405421
22. Yu, E., Mylopoulos, J.: Understanding 'why' in software process modelling, analysis, and design. In: Proceedings of the 16th international conference on Software engineering, pp. 159–168. IEEE, Los Alamitos (1994) ISBN-10: 081865855X
23. van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: Proc. RE 2001: 5th Intl. Symp. Req. Eng. (2001)
24. Regev, G., Wegmann, A.: Where do goals come from: the underlying principles of goal-oriented requirements engineering. In: Proc. of the 13th IEEE International Conference on Requirements Engineering (RE 2005), Paris, France (August 2005)
25. Rifaut, A., Dubois, E.: Using Goal-Oriented Requirements Engineering for Improving the Quality of ISO/IEC 15504 based Compliance Assessment Frameworks. In: Proceedings of the IEEE International Conference On Requirements Engineering (RE 2008), Barcelona, Spain. IEEE Press, Los Alamitos (2008)
26. Tapscott, D., Caston, A.: Paradigm Shift – The New Promise of Information Technology. McGraw-Hill, New York (1993) ASIN 0070628572
27. Wagter, R., Berg, M.v.d., Luijpers, J., Steenbergen, M.v.: Dynamic Enterprise Architecture: How to Make It Work. Wiley, New York (2005) ISBN-10: 0471682721
28. Op't Land, M., Proper, H., Waage, M., Cloo, J., Steghuis, C.: Enterprise Architecture – Creating Value by Informed Governance. Springer, Berlin (2008) ISBN-13: 9783540852315
29. Capgemini: Enterprise, Business and IT Architecture and the Integrated Architecture Framework. White paper, Utrecht, The Netherlands (2007)
30. The Open Group – TOGAF Version 9. Van Haren Publishing, Zaltbommel, The Netherlands (2009) ISBN-13: 9789087532307
31. Dietz, J.: Architecture – Building strategy into design. Netherlands Architecture Forum, Academic Service – SDU, The Hague, The Netherlands (2008), `http://www.naf.nl`, ISBN-13: 9789012580861

32. Proper, H., Greefhorst, D.: The Roles of Principles in Enterprise Architecture. In: Proceedings of the 5th workshop on Trends in Enterprise Architecture Research, Delft, The Netherlands. LNBIP, vol. 70, pp. 57–70. Springer, Berlin (2010)
33. Meriam–Webster: Meriam–Webster Online, Collegiate Dictionary (2003)
34. OMG: UML 2.0 Superstructure Specification – Final Adopted Specification. Technical Report ptc/03–08–02, OMG (2003)
35. Object Management Group: Business process modeling notation, v1.1. OMG Available Specification OMG Document Number: formal/2008-01-17, Object Management Group (2008)
36. Iacob, M.E., Jonkers, H., Lankhorst, M., Proper, H.: ArchiMate 1.0 Specification. The Open Group (2009) ISBN-13: 9789087535025
37. Dietz, J.: Enterprise Ontology – Theory and Methodology. Springer, Berlin (2006) ISBN-10: 9783540291695
38. ISO: Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base (1987) ISO/TR 9007:1987
39. Batini, C., Ceri, S., Navathe, S.: Conceptual Database Design – An Entity–Relationship Approach. Benjamin Cummings, Redwood City (1992)
40. Halpin, T., Morgan, T.: Information Modeling and Relational Databases, 2nd edn. Data Management Systems. Morgan Kaufman, San Francisco (2008) ISBN-13: 9780123735683
41. Zachman, J.: A framework for information systems architecture. IBM Systems Journal 26 (1987)
42. Object Management Group: MDA Guide v1.0.1. Technical Report omg/2003-06-01, Object Management Group (2003)
43. Wout, J.v., Waage, M., Hartman, H., Stahlecker, M., Hofman, A.: The Integrated Architecture Framework Explained. Springer, Berlin (2010) ISBN-13: 9783642115172
44. Op't Land, M.: Enterprise Engineering and Enterprise Governance. In: Dutch National Architecture Congres, LAC 2009 (2009) (Presentation, in Dutch)
45. Rijkswaterstaat: DA'S Nieuws. Newsletter Domain Architecture Shipping Traffic Management (2009)
46. Venema, C.: Domeinarchitectuur SVM; Met DAS aan het werk in de casus Ligplaatsenbeheer. In: Dutch National Architecture Congres, LAC 2009 (2009) (Presentation, in Dutch)
47. Yu, E., Mylopoulos, J.: Using goals, rules, and methods to support reasoning in business process reengineering. International Journal of Intelligent Systems in Accounting, Finance and Management 5, 1–13 (1996), Special issue on Artificial Intelligence in Business Process Reengineering
48. Matulevicius, R., Heymans, P., Opdahl, A.: Ontological Analysis of KAOS Using Separation of Reference. In: Krogstie, J., Halpin, T., Proper, H. (eds.) Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2006), held in conjunctiun with the 18th Conference on Advanced Information Systems (CAiSE 2006), Luxembourg, Luxembourg, pp. 395–406. Namur University Press, Namur (2006)
49. BMM Team: Business Motivation Model (BMM) Specification. Technical Report dtc/06–08–03, Object Management Group, Needham, Massachusetts (2006)
50. ICTU: Nederlandse Overheid Referentie Architectuur 2.0 – Samenhang en samenwerking binnen de elektronische overheid. (2007) (in Dutch), `http://www.ictu.nl`
51. Bommel, P.v., Buitenhuis, P., Hoppenbrouwers, S., Proper, H.: Architecture Principles – A Regulative Perspective on Enterprise Architecture. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Enterprise Modelling and Information Systems Architectures (EMISA 2007), Bonn, Germany, Gesellschaft fur Informatik. Lecture Notes in Informatics, vol. 119, pp. 47–60 (2007)
52. SBVR Team: Semantics of Business Vocabulary and Rules (SBVR). Technical Report dtc/06–03–02, Object Management Group, Needham, Massachusetts (2006)

53. Lankhorst, M., et al.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin (2005) ISBN-10: 3540243712
54. Gordijn, J., Akkermans, H.: Value based requirements engineering: Exploring innovative e-commerce ideas. Requirements Engineering Journal 8, 114–134 (2003)
55. Hevner, A., March, S., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28, 75–106 (2004)
56. Lankhorst, M., Proper, H., Jonkers, H.: The Architecture of the ArchiMate Language. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, H., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterpise, Business-Process and Information Systems Modeling – 10th International Workshop, BPMDS 2009 and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands. Lecture Notes in Business Information Processing, vol. 29, pp. 367–380. Springer, Berlin (June 2009) ISBN-13 9783642018619
57. Op't Land, M.: Principles and architecture frameworks. Technical report, Radboud University Nijmegen, The Netherlands, Educational material of University-based Master Architecture in the Digital World (2005)
58. IFIP-IFAC Task Force: GERAM: Generalised Enterprise Reference Architecture and Methodology (1999), Version 1.6.3, Published as Annex to ISO WD15704