New instruments

for dynamic Building-Construction:
computer as partner in construction

Cover illustration: the front page shows the church of Colmar, France. In medieval times, they were quite capable of constructing beautiful, elaborate buildings—without a computer in sight.

The back of this thesis shows a small part of the church tower of Thann, France.

New instruments
for dynamic Building-Construction: computer as partner in construction

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. dr. ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 15 januari 2007
om 15:00 uur
door

Reinout VAN REES

civiel ingenieur
geboren te De Bilt.

Dit proefschrift is goedgekeurd door de promotoren:

Prof.dr.ir. H.A.J. de Ridder
Prof.dr.ir. I.S. Sariyildiz

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. H.A.J. de Ridder	Technische Universiteit Delft, promotor
Prof.dr.ir. I.S. Sariyildiz	Technische Universiteit Delft, promotor
Prof.ir. F.P. Tolman	Technische Universiteit Delft
Prof.dr.ir. I. Horváth	Technische Universiteit Delft
Prof.dr.ir. B. de Vries	Technische Universiteit Eindhoven
Prof. C.J. Anumba, PhD	Loughborough University
Dr.ir. H.M. Böhms	TNO Bouw en Ondergrond

The research presented in this thesis has been sponsored by the EU and by stichting STABU.

Author email: `reinout@vanrees.org`
Dissertation available on-line at `http://vanrees.org/research/phd`
ISBN: `978-90-808007-3-1`

To Annie (for loving me) and to Rianne and Floris (for not deleting and eating my dissertation, respectively).

Military engineers build missiles.
Civil engineers build targets.

Preface

This thesis culminates some six years of PhD research at Delft University of Technology. The start was a deep plunge into product modelling at the start of my graduation project—by being thrown in head first by my professor, Frits Tolman: two weeks after starting my graduation project, the EU research project 'eConstruct' also started and I had to participate fully from the beginning.

I fitted right into eConstruct and immensely enjoyed working together in that international setting. After my graduation I continued right away as a PhD, working also part-time at STABU and thereby gaining valuable experience in a much more business-oriented setting.

The motivation for undertaking this research was the attractive combination of civil engineering and ICT, with an emphasis on the Internet for the latter. Within eConstruct there was every opportunity to push this combination hard, which we enthusiastically did. Working together closely with graduate student Noor Hellemans helped winding up the practical side of things (meaning the applications) at the end.

Thanks go to Delft University of Technology and STABU for funding my research. Special thanks to Frits Tolman for giving both guidance when needed and freedom whenever possible as my daily supervisor and initial promotor. Thanks to Sevil and Hennes for being my promotors.

Thanks to all the colleagues at the university and STABU, the graduate students I worked with, the fellow researchers in eConstruct and the members of my committee. Roommates Hans and Johan: thanks for everything. Jaco, Bert, Herman and Gerben for lots of

information and fun. Jeff, David, Celson, Kees, Peter and Michel. Joost, Wouter, Jasper, Noor. Marco, Bart, Annie and Maurits for reading lots of pages.

Annie, thanks for loving me and allowing me to work on this thesis.

Nieuwegein, 2006
Reinout van Rees

Getting him to write anything is usually like trying to pull teeth from a live sabre-tooth tiger while wearing a necklace of freshly killed lamb chops.

Summary

For many years Information and Communication Technology (ICT)-related research for Building-Construction (BC) process innovation has been researching communication technologies that would help the BC industry with the application of computers and computer networks. After 20 years, progress has been made, but real improvements in speed and consistency were lacking. The main reason is that our instruments are not able to understand our technical communications. Humans have to read drawings, extract input for their instruments, let those instruments do their jobs, translate computer output back into humanly understandable information and pass that on. All these transformations do not add value. On the contrary, they are error prone and add to the confusion.

With the arrival of XML/RDF and the Semantic Web a new opportunity arose to really improve human-computer and computer-computer communication. This might potentially be the missing technology the BC industry needed for a long time. What if we could communicate with our instruments in the same technical terms as we use between ourselves? Could ICT communication be elevated to that level? Would that mean that unneeded human activities can be cut out of the information chain and that humans no longer would have to function as translators between the project communications and the computer applications? Would that mean a reduction in communication errors and in cost of failure? Would that speed up the process? What would the consequences be and how should this new technology be implemented?

Could it be that the missing technology is the ability for instruments to participate in human technical communication?

The research was performed at Delft University of Technology and at STABU, the Dutch building Specification institute, so BC ICT in theory and in practice. The first two years took place within the EU research project 'eConstruct' which ensured a good embedding in international BC ICT research.

Current first generation ICT support mainly focuses on individual tasks and not on project-related tasks requiring information and knowledge sharing. (Computerised) Classification and Specification systems are widely accepted in the market, but do not offer enough information richness to adequately support information sharing in BC. Product Data Technology (PDT) solutions are either too centralised or effectively cater for the top 5% of the companies only, not for BC as a whole.

The research reported here was mainly aimed at the use of innovative Internet technologies on the intersection of two fields: (a) existing BC ICT, especially product modeling and information exchange; (b) the desired process innovation in existing BC practice.

Internet has created an unprecedented number of new possibilities. The question arose which of those would prove usable in BC ICT to help initiate BC process innovation. Also the question arose as to the manner in which to use and implement those possibilities.

After the first analysis chapters, the research questions were revisited and reformulated. How can state-of-the-art web technology possibly help to realise new, dynamic BC processes that adequately serve the Client's needs (provide value for money) and, at the same time, serve the needs of the BC companies (provide money for value)? How to use the Next Generation Internet (NG Internet) to provide both parties (and their application software) with the precise and Semantically rich on-line BC object definitions? How can software vendors take advantage of such a development? What is the possible role of traditional Specifications in newly innovated BC processes?

According to the solution concept, the value adding performance of the BC industry can be much increased by creating an open, freely accessible Building-Construction Ontology Web (bcoWeb), meaning: (a) Stimulating the development of domain specific BC Ontologies containing definitions of BC terms structured according to a GARM-

based model. BcoWeb's main strength is its focus on the interaction points within BC: the points where demand and supply must be matched. (b) Relating the different BC Ontologies in a national open source bcoWeb. Earlier standardisation efforts tried in vain to provide international standards for information exchange. One of the conclusions to be drawn from the European eConstruct project was that BC is mainly country-specific (inner doors are different in each country), so international standards development can only start after national standards have become available. (c) Stimulating the development of co-operative BC web services that 'understand' the BC terms and can be used as references to support communication between all the stakeholders—including the general public—where 'smart' computer applications and building documents are also seen as stakeholders.

BcoWeb has many advantages: it serves as a reference vocabulary available fur humans and computers; it is open source, so safe to build upon; it has rich Semantics, provides context-dependent definitions, supports multiple representations, so there are many possibilities for innovative and value-adding applications, amongst others by using web services.

During the implementation phase, a bcoWeb web-based collaborative editor was created which was used to fill bcoWeb. Furthermore, a catalog and an object tree application were made and a simple Specification generation application. The cases showed that such a bcoWeb approach provides the flexible, commonly accessible BC language required for the development of 'smart' computer applications that are able to behave as if they really understand the technical language(s) of BC.

BcoWeb and the applications surrounding it have a number of advantages over earlier mainly paper-based mechanisms. Computer applications can use the bcoWeb to share a common view on the project. Providing Internet-based access to the data that underlie the textual documents helps to turn those documents into 'smart' applications that are able, like human co-workers, to complete tasks in close co-operation with other co-workers and other smart computer applications, just like they understand the technical BC language. They can complete many tasks regarding information and knowledge

management without human interference—though not without human control—thus providing a promising new technology that seems able to drastically improve current BC processes. The primary information process can be best left to the information system, just like in the automotive industry, as that lowers the number of errors.

A contribution of this research: using the General AEC Reference Model (GARM) results in a kind of Ontology that is much more BC-oriented and that also has a direct economic connection with BC because it simulates the demand/supply process. A second contribution is the emphasis on the necessity of an open source license for the Ontologies that should underpin the information exchange of the whole of BC: that is no place for individual companies' or projects' restrictions. Third, this thesis' use of GARM also allows different views and definitions for the various disciplines. Definitions can be voluntary referenced.

An important contribution is the emphasis on practical applicability for the average BC company—quite unlike research that aims at the top 1% of the BC companies that can afford the big project databases and the most expensive CAD applications.

Specifications will expand their central role in the BC process if they use bcoWeb to become a valuable source of well-ordered and well-accessible information.

What is the recommendation? What should be done? BC should start national BC Ontology developments like demonstrated here with bcoWeb. The start can be small, just a set of Ontologies that together provide enough information richness to enable communication.

An important role is given to information and software providers. Without end-user software, the bcoWeb proposal is just an idea on paper.

If you're not sure what to do, make something.

Paul Graham

Contents in brief

Information overload isn't the problem. If it was, you'd walk into a library and die.

David Allen

Contents

A university is an aggregation of sovereignties connected by a common heating plant.

List of Figures

The most expensive part of the building is the mistakes.

Ken Follett, the pillars of the earth

1 Introduction

The aim of this chapter is to introduce the thesis and the motivation to start this PhD research. It also introduces the initial research questions present at the start of the research. The next chapters contain the analysis regarding these questions.

This chapter *introduces* the research topic, presents a *characterisation of BC*, provides the *initial research questions* and, lastly, gives an overview of the *content of the chapters* and some notes on *the typography used and the reading aids*.

1.1 Introduction

This PhD research is part of an ongoing research effort of the building processes research group [1] at Delft University of Technology in the Netherlands. The focus of the building processes research in Delft has been strongly on process innovation for many years. PDT, ISO-STEP, IAI-IFC, Object Trees, and other process related topics have been studied in relation to long-standing problems in the BC industry. At the time this research project was initiated, first results of

new Internet developments were made public. The World Wide Web Consortium (W3C) proposed eXtensible Mark-up Language (XML) as the successor of HyperText Mark-up Language (HTML). XML is an interesting Internet language that separates the content of a message from its mark-up (or presentation). The resulting web page strongly resembles a HTML page, but the fact that the end-result has been created by applying a separated set of mark-up rules to the content, makes it possible to allow computer applications direct access to the content, while ignoring the human-oriented mark-up rules accompanying the message. This XML feature also supports the application of *different* sets of mark-up rules to the same content. Content of messages can be transformed into many forms: different natural languages, different alphabets, graphical form, voice output, dedicated application formats and much more [2].

It slowly occurred to us that XML might potentially be the missing technology the BC industry needed for a long time. Could it be possible to elevate the ICT communication to the level of our human co-workers? What if we could communicate with our instruments in the same technical terms as we use between ourselves? Would that mean that unneeded human activities can be cut out of the information chain and that humans no longer would have to function as translators between the project communications and the computer applications [3]? Would that mean a reduction in communication errors and in cost of failure? Would that speed up the process? What would the consequences be and how should this new technology be implemented?

Could it be that the missing technology is the ability for instruments to participate in human technical communication?

This realisation was the primary reason to (1) formulate an MSc thesis research project for the author of this PhD study and (2) to write a winning proposal for a European project, called eConstruct [4].

The results of the author's Master's thesis[1] research [5] were partially adopted by the eConstruct project: eConstruct developed a mechanism that creates meaningful XML tags like `<ConcreteBeam>`

[1]Dutch: *afstudeerverslag*.

from data items in a list (a list of 'beam' types in the case of the example) [6].

After finalising his Master's thesis the author embarked on a PhD research project that wanted to look into the eFuture of BC with a special emphasis on the future role of traditional Specifications. From Specification to eSpecification, that was clear, but why, how and with which result was not clear at all.

The first two years were spend in the European eConstruct project [7], researching applicability of modern Internet technologies in the BC industry. W3C yearly develops new additions to Internet technology, RDF, XXX, YYY (sic) and lately OWL. And each new addition increases the expressiveness of the Internet. *How should BC profit from this development?*

Research on eSpecifications and their role in future BC processes soon showed that detailed BC Semantics should be made available on-line [8]. Specifications have a central place in the information flow in BC, providing a legal basis for contractual agreements. This requirement fitted well with the latest developments on the Internet.

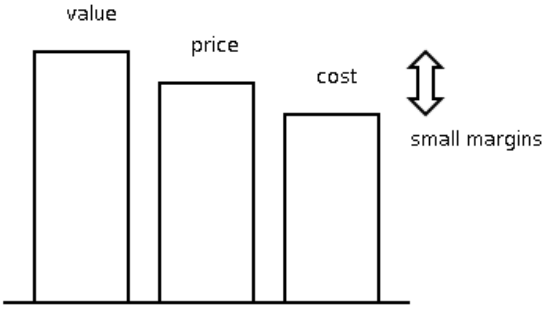
1.2 Characteristics of Building-Construction

The BC industry is under a lot of pressure: unsatisfied clients, building frauds, costs of failure, nothing very optimistic to say it mildly. Improving the value adding performance of BC is the challenge [9].

In BC, little value is added. There is, mostly, only a small difference between actual costs to build the product and value provided to the customer. Assuming that the price is set between the costs and the value, only little room exists for profit for the Virtual Enterprise that performs the project. Also, relatively little extra value is provided for the money spend by the customer. See figure 1.1 on the following page.

Why is it that BC is not the powerful industry that it could be? Many reasons have been mentioned, *i.e.* uniqueness of the building

Figure 1.1: *Value, price and cost: with small margins, the Client doesn't get much added value compared to the price and the Contractor doesn't get much profit compared to the cost.*



projects [10], uniqueness of the Virtual Enterprises, fragmentation of the BC Information Systems (ISs) [11], lack of strong market leaders besides the national and local governments, the fact that buildings and civil engineering products have to be build on-site, a strong focus on minimum price, and several more. The question is of course whether these reasons are cause or effect. Undoubtedly some external influences like bad weather or traffic jams are playing a negative role in the BC industry, but it is by no means clear that the value adding performance of the BC industry cannot be substantially improved.

In fact the hypothesis of this PhD research is that ICT (and especially the Internet) can help BC to perform like any high-tech industry. The question is not if, but how. And it is this question—*how can state of the art ICT help to improve the performance of the BC industry* - that is targeted here.

1.2.1 From construction to eConstruction

As said above, collaboration means information exchange or, more precisely, information and knowledge sharing. The following paragraphs discuss one-off prototype building and the desire by the customers to get more grip on the process and on the end result. Also a comparison with other industries is made, followed by a short discussion on contract characteristics.

BC is often characterised as a collection of information islands [12] (figure 1.2 on the next page). There are a lot of different parties: Clients, Contractors, sub-contractors, engineers, architects, (local) governments, specialists, the fire department. Also characteristic is a subdivision in disciplines: civil engineering, house building, electro-technical installations and so on. Everyone has their own systems, possibly even their own favourite separate Classification system[2]. This subdivision does not yet take into account the islands of information within actual companies. Are the designer, engineer and costing expert working with the same information? Above fragmentation prevents needed collaboration.

The needed collaboration means there is a huge amount of information that needs to be communicated. Technical data is needed by the sub-contractors. Safety checkers need to be able to verify the correctness of technical data. The project planning must be communicated to every worker, taking the form of work orders. Also suppliers need to know when to deliver which goods.

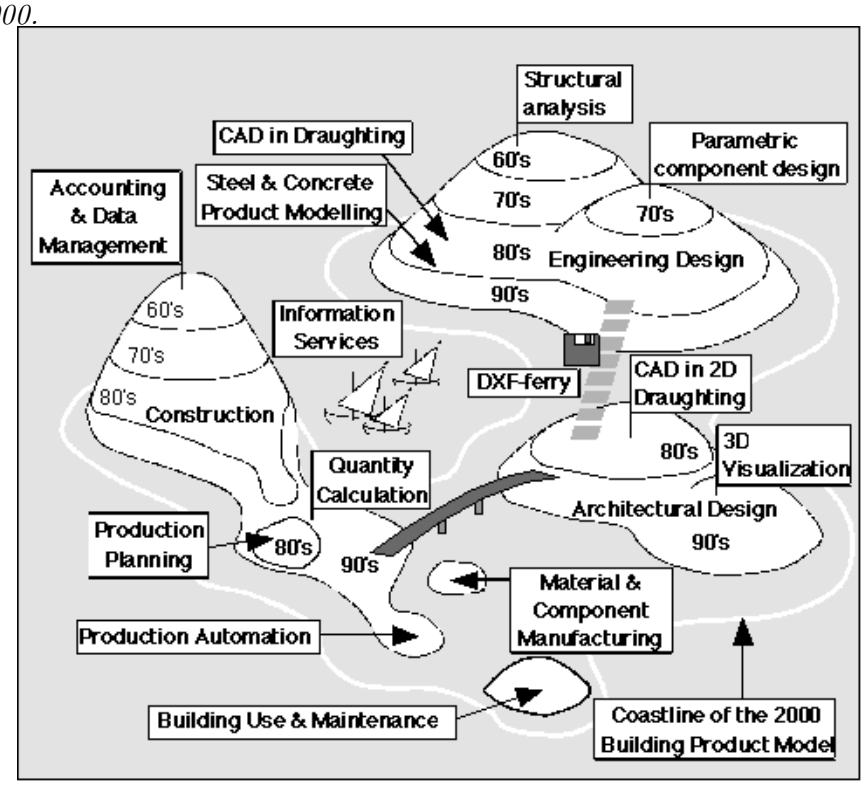
Regarding this information, current practice in the BC industry is mainly paper-based, though electronic communication is an alternative. If an electronic way of communication could reach the same level of trust and—also important—the same legal status, it could replace much of the paper-based communication, with the possible advantages of speed and accuracy associated with electronic communication.

A second characteristic: a lot of work basically consists of prototypes. Unless entire city extensions are build in one go, individual projects are mostly one-of-a-kind. Cooperation is in per-project consortia (Virtual Enterprises).

Third, there is a growing need for the Client to get more grip on the process. For large infrastructural works it is normal to greatly exceed the budget [13]. There are also quality issues: the Dutch house owners' association *Vereniging Eigen Huis* (VEH), for example, does a lot of post-building check-ups to find errors in brand new houses like

[2]In the Netherlands, architects use the *elementenmethode*. For Specifications the STABU system is used. The electro-technical industry uses ETIM. And for road and water construction there's CROW.

Figure 1.2: *Islands of automation illustration, originally by Matti Hannus [12]. This version is from Per Christiansson,* `http://it.civil.aau.dk/it/education/overview.html`. *It illustrates the various separate islands of automation while at the same time expressing the hope that the next level of technology can finally raise the land enough to merge the islands. Note that the figure slots that joyful moment for an arrival in 2000.*



missing doors, mis-aligned stairs. VEH finds, on average, 28 failures in any newly constructed house[14].

Following from the nature of the industry and the wishes of clients, *collaborative construction* is important. There is a great need to communicate meaningfully between islands of information, exchanging information and knowledge in order to reach an improved and more effective cooperation.

Regarding this communication need: the Internet provides a lot of possibilities for exchanging data and for structuring data. How to capture and use construction Semantics using NG Internet technology? What is the relation to existing approaches like Classification and PDT, does the web approach help to overcome their shortcomings?

1.3 Initial research questions

As explained above, the goal of this research is to help BC become a knowledge-rich industry with well controlled processes and guaranteed output quality. This overall goal has been translated into a number of research questions.

The first research question targeted in this PhD thesis is the question how new instruments can help BC to increase its value adding performance. Formulated like that, the question is still too broad for one humble PhD student to answer. Chapter 2 will be used to clearly identify the problem and to scope the rather broad research question down.

The second research question is to investigate the process that brings new instruments to the market, *i.e.* which type of standardisation is required (if any) and how should the software vendor community react? The question comes from the observation that the time-to-market of research and development results in this area should be as short as possible.

The third initial research question is: 'What is the role of the traditional Specification in the future?' This question has to be answered in the context of the solution concept that targets the two

other questions.

1.4 Thesis overview and methodology

Analysis The first part of the thesis analyses the current situation and the available technologies.

Chapter 1 is the *introduction*, it introduces the BC industry, poses initial research questions and provides an overview of the thesis.

Chapter 2, *problem analysis*, analyses the current situation in the BC industry and improves one of the initial research questions.

Chapter 3, *analysis of current information and knowledge sharing in the Building-Construction industry*, analyses the history of methods for information and knowledge structuring and suggest research opportunities for improvement. The goal is to analyse what works well currently, what lacks and what needs improvement.

Chapter 4, *Analysis of new applicable Next Generation Internet technologies*, analyses the applicability of the NG Internet as a vehicle to improve the information and knowledge sharing capacity of BC. The goal is to extract the technologies that can be most useful in improving BC's value-adding performance.

Research questions Chapter 5, *revisiting the research questions*, revisits the initial research questions. The goal is to take the analysis of the previous chapters and prepare it for the solution concepts chapter.

Solution and implementation Chapters 6–8 propose a solution concept and implement and test it.

Chapter 6 presents the *solution concepts* proposed to help solve the problem.

Chapter 7, *technical prototype implementation*, discusses aspects of the technical implementation of the instruments. The goal

is to provide the technical background of the prototype applications that were built to demonstrate and test the solution concepts.

Chapter 8, *case studies*, evaluates the solution concept by means of two case studies where the instruments discussed in the previous chapter have been applied.

Conclusions Chapter 9, *conclusions and recommendations*, presents the conclusions of the research and gives recommendations for further research and development work.

1.5 Note on typography and reading aids

Words written with a Capital Letter or all in CAPS are explained in the list of acronyms and definitions on page 215. Acronyms are normally written in full the first time they occur in a chapter, like for instance Building-Construction Ontology Web (bcoWeb).

Words in *italics* are in a non-English language, like *Samarbetskommittén för Byggnadsfrågor* (SfB). Italics is also used to provide emphasis on *certain* words.

Words in 'quotes' are terms from an Ontology, like 'overpass' or 'traffic class'. This is used when it is necessary to distinguish between, for instance, an overpass in general and the object 'overpass' from an Ontology.

There are two additional reading aids. The index on page 229 allows you to look up items of interest in a more search-oriented way. Likewise, the author index on page 230 allows you to look up references per author.

The references for a chapter are grouped together at the end of each chapter. This way, it is easier to look up references than in one single long list at the end of the document. Also, readers of individual chapters derive more utility of a by-chapter list. Aforementioned author index still allows a thesis-wide search.

Every chapter starts with two short paragraphs, the first stating the aim of the chapter, the second summing up the chapter's content.

The first introductory section always places the chapter in the context of the rest of the thesis.

Bibliography

[1] Website of the section 'building processes' at Delft University of Technology, 2004. Available on-line at `http://www.bouwprocessen.citg.tudelft.nl/`.

[2] Tim Bray, Jean Paoli, C Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (XML) 1.0 (third edition), February 2004. Available on-line at `http://www.w3.org/TR/2004/REC-xml-20040204/`.

[3] Wim Gielingh. *Improving the performance of construction by the acquisition, organization and use of knowledge*. PhD thesis, Delft University of Technology, October 2005. Available on-line at `http://repository.tudelft.nl/file/89968/071532`.

[4] Jeff Stephens and Frits Tolman. eConstruct – now you're talking our language, 1999. Available on-line at `http://xml.coverpages.org/econstructPR01.html`.

[5] Reinout van Rees. ceXML - an XML vocabulary for building and civil engineering. Master's thesis, Delft University of Technology, August 2000. Available on-line at `http://vanrees.org/research/econstruct/afstudeerverslag`.

[6] Reinout van Rees. ceXML - an XML vocabulary for building and civil engineering (chapter 6). Master's thesis, Delft University of Technology, August 2000. Chapter 6 available on-line at `http://vanrees.org/research/econstruct/afstudeerverslag/messagevocabulary.html`.

[7] Website eConstruct project, 2000-2002. Available on-line at `http://www.econstruct.org/`.

[8] Reinout van Rees and Frits Tolman. Semantic web technologies applied to building specifications. In *Conference proceedings - CIB world building congres Toronto*. CIB, 2004. Available on-line at `http://vanrees.org/research/papers/`.

[9] Hennes de Ridder, Roland van der Klauw, and Ruben Vrijhoef. Het nieuwe bouwen in Nederland. Technical Report 2002-BPI-028, TNO bouw, December 2002.

[10] Peter Willems. *Conceptual modelling of structure and shape of complex civil engineering projects*. PhD thesis, Delft University of Technology, 1998.

[11] Sander van Nederveen. *Object Trees - improving Electronic communication between participants of different disciplines in large-scale construction projects*. PhD thesis, Delft University of Technology, 2000. Available on-line at `http://repository.tudelft.nl/file/117347/098324`.

[12] Matti Hannus. Islands of automation in construction. In Žiga Turk, editor, *Construction on the information highway*, number 198 in CIB publication, page 20. University of Ljubljana, 1996.

[13] Bent Flyvbjerg, Mette K. Skamris Holm, and Søren L. Buhl. What causes cost overrun in transport infrastructure projects? *Transport Reviews*, 24(1):3–18, January 2004. Available on-line at `http://flyvbjerg.plan.aau.dk/COSTCAUSESASPUBLISHED.pdf`.

[14] Website *Bewonersvereniging* Terwijde, 2006. Information page about VEH, quoting 28 errors per house in 2002 (which was a top year). Available on-line at `http://www.terwijdeweb.nl/veh/index.html`.

The path is clear
Though no eyes can see
The course laid down long before.
And so with gods and men
The sheep remain inside their pen,
Though many times they've seen the way to leave.

Genesis, firth of fifth

2 Problem analysis

Successful BC process innovation results from a complex transformation process that involves organisational, technical, legal and cultural changes. Too many, and too complex for a single PhD study to target. This chapter scopes down the initial research questions that closely relate to BC's effort to increase its performance. The scoping process is done from an ICT point of view and with the aim of improving BC's value adding capacity.

The *introduction* puts this chapter into perspective. The next section deals with an *observation on business-level innovations* required to improve the BC industry, followed by a short analysis of *current ICT usage in SMEs*. The reformulation of an initial research question is presented in the *conclusion* of this chapter.

2.1 Introduction

From an ICT viewpoint, BC is lagging behind. A European officer of the Esprit R&D programme, whose name for obvious reasons will not be disclosed, once joked that there are high-tech industries, medium-

Figure 2.1: *A bridge Building-Construction can be proud of: Firth of Forth rail bridge, Scotland. Photo: Michael Laing*

tech industries and low-tech industries, and there is Building-Construction.

Though BC can be proud of most of its water works, bridges (see figure 2.1) and buildings, from an ICT point of view, the European officer's remark is not wholly untrue. ICT could do wonders in process innovation [1] and improving value adding: BC can become a medium- or even high-tech industry, also from an ICT point of view.

2.2 Observing Building-Construction

There are many areas for process and system innovation: improved flexibility, co-operation, process control, value-based steering, and knowledge utilisation. The next sections elaborate these points.

2.2.1 Improving flexibility

One of the aims of the BC industry could be to change the current static nature of the process into a more *dynamic* structure that is much more tolerant for change, and does not require early agreement about all the details that cannot be over-seen beforehand [2]. The

static nature of the current process can also be described as *sequential* and *monotonous*. The various phases are executed sequentially. In a dynamic process, phases interact and overlap much more and earlier decisions can be changed with full knowledge of the consequences.

Improving the flexibility of the BC industry will require many developments. Most are out of scope of this research. One important requirement is that dynamic processes require tight control over the information and knowledge logistics. Only if everybody involved has access to the right information and the right knowledge on the right time, can increased flexibility be implemented. Another requirement is that consequences of changes are made transparent and controlled, following mutually agreed rules.

2.2.2 Improving co-operation

An important weakness of the BC industry is the limited collaboration between the demand side and the supply side to get the demand sorted out correctly [3]. For example, a fixed Specification is made—only then does the Contractor get a look at it, often too late for suggestions to apply different construction products and methods.

But not only co-operation between demand and supply is weak, also co-operation between project partners is weak [1] [4]. Perhaps it is not the *intention*, but the *ability* to co-operate which is insufficient. It seems that everybody nowadays understands that unsatisfied clients are bad for the business, but what can you do? There is always the stress to make money, always too little time left.

Of course there are many reasons why the situation is as it appears to be now. Increasing BC's ability to co-operate is closely related to BC's ability to communicate and to the way process management is able to translate project goals into efforts, risks and money.

In the spirit of this research the solution is sought in the increase of ICT to help us out. True, the current crop of application software and computer networks is not able to communicate (see the next chapter, 3, for a thorough analysis), but maybe that can change.

2.2.3 Improving process control

To run BC projects as tight industrial processes with well integrated supply chains, *i.e.* getting things done on time, with the agreed quality or value and for the agreed price is a major challenge—even more if the process itself becomes more flexible. Such level of control, comparable with that of the automotive industry [5] [6], is only possible if computers take over much of the primary control role of the human actors. Again this seems a matter of increased information and knowledge sharing.

In a more dynamic building process, a new generation of building documents is needed to support the dynamic character of the processes—instead of dragging them down through inertia.

Dynamic processes are more knowledge-intensive, increasing the requirements for usability and accessibility of that knowledge. The operational building process is facilitated, but also restricted, by the available infrastructure. Infrastructure in this context is the possibility to store, share and use information and knowledge between partners in the building process.

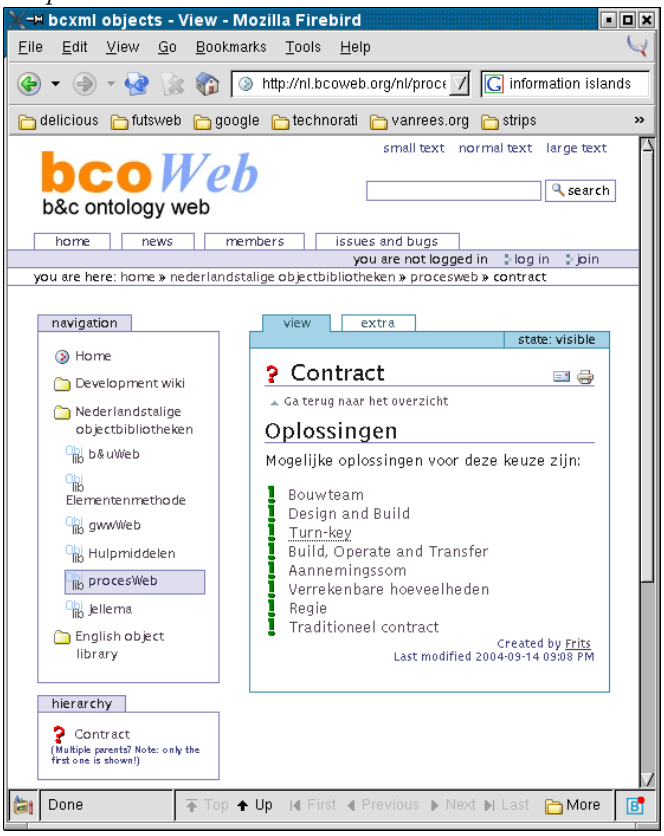
Most progress has been made in the field of contracts. Design-build, design-build-operate, performance contracts, etcetera (see figure 2.2 on the next page). In the wake of these contract types, changes in Specification practice could follow. This way, research attaches itself to a forward movement of building practice, but in turn fuels that same forward movement.

2.2.4 Improving value-based steering

Improving the value adding performance of the industry is vital for satisfied clients and a healthy [1] industry. The Client's influence can be increased by giving him influence on value received and cost incurred. The other way around, BC can add real value for the Client with its knowledge and innovative solutions if that knowledge and those solutions can returned to the Client as feedback. This is a dynamic process with value-based feedback.

Dynamic steering on value is fully explained in [7]. Apart from the social, legal and organisational aspects, dynamic steering on value has

Figure 2.2: *Some contract types (Dutch) in the bcoWeb Ontology, part of chapter 7's implementation. The list is not exhaustive.*



to be implemented in practice. ICT support might be instrumental, which will be further analysed below.

The need for ICT support follows from the dynamic character of the envisioned processes. Dynamic means that there is a continuing process of adjusting, re-assessing the value, further adjusting. ICT support can be used to calculate the value of proposed solutions and to calculate the change in value when changes are required. In such a system, resulting value forms the feedback mechanism for design or construction decisions. For a dynamic system to work well, the feedback mechanism needs to provide accurate and rapid response.

Relying on a mostly human-based feedback means a slow and error

prone feedback loop. Implications of changes have to be understood and checked manually. Parts of the calculations can probably be done by computers, but communicating various implications amongst partners and calculating a final net value of the proposed change are all human processes again.

An alternative, that allows for a much more rapid feedback loop, is to use ICT as much as possible. For a large class of possible changes, calculations can probably even be done automatically, including sensitivity analysis. The results could be communicated electronically between various stakeholders. Such a degree of electronic cooperation is not common in BC nowadays, but it is technically feasible and applied in other sectors of industry. Automatic calculation of the effect of changes on value to the Client and electronic communication of the calculation results can critically shorten the feedback loop.

In a dynamic system, shortening and strengthening the feedback loop is instrumental in improving the efficiency of the whole system. This does not diminish the need for solving the social and juridical sides of the problem, but establishes ICT implementation as a subproblem of like magnitude.

2.2.5 Improving knowledge utilisation

Increasing the value adding process of the BC industry means putting more emphasis on knowledge-intensive processes [3]. Many results of R&D efforts are never used in practise. Likewise, the knowledge learned in previous projects needs to be re-used in current and future projects. An important and direct result is the need for retaining (the knowledge of) knowledgeable and experienced employees.

Knowledge utilisation can also be improved by storing knowledge in an information system, re-usable for a new project. Historically, Classification systems have been used to both store and structure books and documentation in order to make them searchable when the need arose. This does not, however, provide a very powerful and detail-rich way to link the ordered knowledge to problems and objects in a new project.

What is needed to improve knowledge utilisation is a better way

of storing a wide variety of knowledge and allowing it to be referenced flexibly from newer projects. Secondly, information and data collected in current projects needs to be available for subsequent analysis and for inspection by later projects.

2.3 Current ICT usage in Small and Medium Enterprises

When analysing BC, it is interesting to look at the ICT usage in Small and Medium Enterprises (SMEs). The total size of the largest BC companies is way smaller than the combined size of all the SMEs[1]. For analysis of BC as a whole, analysis of the SMEs is therefore more significant, especially when looking at the ICT usage: the shiny applications that can be used by the few biggest companies tend to obscure (or over-illuminate) the generally dim picture.

For the small companies, one can assume a basic level of ICT usage. An Internet connection is normally available; a recent year-long action in the Netherlands tried to raise the number of small construction companies that had an ADSL connection from 50% to 100%, with reasonable success. Word processing and spreadsheets are in place. For the large number of < 5 person local constructors, the level of automation will be equal to that of the proverbial 15 year old nephew (without meaning this in any negative way).

For a higher level of automation, it is illustrative to look at the applications on display at BC ICT shows [12]. Most common are: Computer Aided Design (CAD) packages; administrative software like order management, customer databases, financial packages; document

[1]For the Netherlands, [8] gives a total number of 65000 individual companies, 500 of which are part of the 12 big consortia. That is less than 1%. Dutch industry organisation *Bouwend Nederland* cites SMEs are 90% of the membership [9]. Near the end of [10], a secretary of a Dutch software company federation [11] says that a technology like IFC is something used by the top 1% of the construction companies. They have their own solutions, he and the other software vendors are delivering to the other 99% percent of the market. Also [1] mentions that only 1% of the companies employ more than 100 people.

management (including project extranets[2]); calculation packages for for instance strength calculation. The bigger the company, the more one can expect to see such packages. Of course, even one-man architectural offices will have CAD software; document management systems on the other hand need reasonably large companies before there is enough return on investment.

There are software packages for SMEs that are advertised as complete solutions for the entire BC process, mostly with a CAD system at the core. Within one such system, a level of cooperation and information re-use is feasible. Once you cross organisational boundaries—which happens often, as BC projects are normally done together with other companies—there is only a small chance of encountering the same system on the other side of the organisational boundary. Which means that such integrated packages are not a solution for improved information exchange in BC as a whole. Often, the limited technical basis of the Classification and Specification systems make it very hard to couple such data (see the beginning of chapter 3).

Concluding, the availability of computers, basic word processing and Internet connectivity is not a problem. The further level of automation leaves much to be desired, especially when looking at the meaningful exchange of information. Part of this is due, however, to the underlying technologies.

2.4 Conclusions and improved initial research question

BC process innovation should focus on a substantial increase of the information and knowledge sharing capacity of the industry at large. Unlike industries such as automotive and shipbuilding, BC projects are usually temporary joint ventures of companies (Virtual Enterprises) that realise a unique project on a unique construction site. Information and knowledge developed is in general not systematically

[2]The Dutch project extranets that I investigated resembled web-based document management systems with a forum and discussion possibility and some workflow.

stored for re-use in later projects. Goal of this research is to investigate a possible approach that can help to increase BC's capacity to (electronically) communicate and co-operate.

It seems likely that increasing BC's computer assisted information and knowledge sharing capacity can help to change the current *static* nature of the process into a more *dynamic* structure that is much more tolerant for change and does not require early agreement about all the details that cannot really be over-seen beforehand. In a dynamic system, shortening and strengthening the feedback loop is instrumental for improving the efficiency of the whole system. This does not diminish the need for solving the social and juridical sides of the problem, but establishes ICT implementation as a sub-problem of like magnitude.

2.4.1 Improved research question

The first research question formulated in chapter 1 was the question how new instruments can help the BC industry to increase its value adding performance.

To scope the question further, the research is limited to new instruments that help to improve ICT-supported information and knowledge sharing between all stakeholders, especially in the various chains. The hypothesis is that improved information and knowledge logistics reduces the costs of failure and increases the value of the output of supply and knowledge chains.

Bibliography

[1] Edwin Dado. *ICT-enabled communication and co-operation in large-scale on-site construction projects*. PhD thesis, Delft University of Technology, October 2002.

[2] Hennes de Ridder. *Design & construct of complex civil engineering systems - a new approach to organization and contracts*. Delft University Press, 1994.

[3] Hans Schevers. *Demand Support by Virtual Experts - Supporting the Client during the Inception Phase of a Building and Construction Project*. PhD thesis, Delft University of Technology, June 2004.

[4] Sander van Nederveen. *Object Trees - improving Electronic communication between participants of different disciplines in large-scale construction projects*. PhD thesis, Delft University of Technology, 2000. Available on-line at `http://repository.tudelft.nl/file/117347/098324`.

[5] James P. Womack, Daniel T. Jones, and Daniel Roos. *The machine that changed the world*. Scribner, October 1990.

[6] Wim Gielingh. *Improving the performance of construction by the acquisition, organization and use of knowledge*. PhD thesis, Delft University of Technology, October 2005. Available on-line at `http://repository.tudelft.nl/file/89968/071532`.

[7] Hennes de Ridder. Process and system innovation in the building industry. In *Conference proceedings - CIB world building congres Toronto*. CIB, 2004.

[8] Wikipedia article on 'bouw' (dutch), 2006. Available on-line at `http://nl.wikipedia.org/wiki/Bouw`.

[9] Website of dutch industry organisation 'Bouwend Nederland', 2006. Available on-line at `http://www.bouwendnederland.nl/`, specific article at `http://tinyurl.com/vm2nq`.

[10] Jasper Feenstra. LexiCon expert panel discussion, October 2004. Available on-line at `http://vanrees.org/weblog/1098802937`.

[11] Website of Forum systeemhuizen, 2006. Website of a Dutch software company federation. Available on-line at `http://www.forumbouw.nl/`.

[12] Author's observation of the applications on display at BC ICT trade shows, 2002, 2003, 2004. Trade show website available on-line at `http://www.ictbouw.nl/`.

Hij, die haar leidt
en in de waarheid stelt,
heeft zijn bestek met wijsheid uitgemeten.

Gezang 304

3

Analysis of current information and knowledge sharing in the Building-Construction industry

Following the conclusions of the previous chapter, this chapter looks into existing and on-going developments regarding information and knowledge structuring methods in BC, primarily from an 'ICT-enabled process innovation' point of view. The goal is to analyse what works well currently, what lacks and what needs improvement.

The *introduction* places this chapter both in historical perspective and in this dissertation's structure. *Specification and Classification* starts with the oldest and most widespread way of information handling. *PDT: information structuring* analyses past and current efforts at structuring information, followed by an *evaluation of improved information sharing*. The *conclusions* provide the input for the next chapters.

3.1 Introduction

Ordering of BC information and knowledge has been subject of research and development for a long time. Somewhere in the sixties the Swedish SfB Classification and coding system started its advance. In the early eighties PDT research efforts tried to find a way to use computers for information and knowledge sharing. In the late eighties the International council for research and innovation in building and construction (CIB) organised a conference that brought Classification and PDT together. From then onwards, researchers have tried to find solutions that improve the information and knowledge sharing capacity of BC, unfortunately still largely without success. This chapter analyses the developments and searches for reasons why BC could not apply technologies used successfully in other sectors of industry.

Over the years, many BC initiatives to improve information and knowledge handling have been started; not too many however have ended in commercial success. In the next sections a number of BC related development efforts will be described in some detail. The selection of development projects is somewhat biased towards the situation in the Netherlands. The underlying expectation is that other countries will show comparable efforts with more or less similar conclusions.

3.2 Specification and Classification

For many years already Specifications and technical drawings, the Tender Documents, form the core of a project Information System (IS). The basis of Specifications is formed by systems for Classification and Coding. Though ICT has been successfully applied in the area of technical drawings (Autocad c.s.) and Specifications, these first generation instruments only play a minor role in inter- and intra- project communication, *i.e.* even if an electronic document is exchanged, humans have to transfer the information by hand into the next application.

The previous chapter concluded that there is a need in BC for improved electronic co-operation and communication. Also, there has

to be a way of communication that is tolerant of changes: supporting a more dynamic BC.

In this section, first Classifications will be analysed, being the basis for most information structuring done in practice nowadays. Second, Specification systems, as an important user of Classifications and as a system that often has a central place in current BC. Third and last, two examples are analysed.

3.2.1 Classification and Coding systems

In the BC industry, Classification systems saw increased adoption in the fifties. Most prominent example is the Swedish *Samarbetskommittén för Byggnadsfrågor* (SfB) Classification and Coding system, which, in one form or another, has been in use ever since. It has received CIB endorsement and is used in many countries. Recently, the formalisms behind the SfB family of Classifications were standardised in the International Standardisation Organisation (ISO) 12006-2 standard [1].

A more rigorous variant of the original SfB, Co-ordinated Building Communication (CBC), was created in Denmark by Bjørn Bindslev [2], mainly with computer-based support in mind. It used very strict boundaries between classes for that aim. This fits in well with the Merriam-Webster dictionary definition 'a systematic arrangement in groups or categories according to established criteria'.

Classification is a grouping of entities according to some external criteria. The grouping will be quite natural, as it is mostly made from a specific viewpoint. Classification is basically a set of boxes (with labels) to sort things into and can be used as a user-friendly view on other kinds of information storages [3].

Goal of a Classification is to allow information, documents and drawings to be ordered [4]. The level of granularity is quite low, an example is the use of Classifications in naming layers of CAD drawings. Doors and windows can be described in layer `30` for instance; a more granular subdivision is not normally feasible in drawing practice.

For improved co-operation and more knowledge utilisation, this granularity level seems low. Alternatively, if a Classification can be

seen as a user-friendly way of ordering information, it can also be used as a user-friendly way of presenting information stored in more elaborate systems.

Looking at the design problem in the design stage from only one viewpoint is not enough [5]. A wall can be part of the main load-bearing construction; provide sound and heat insulation; provide safety; serve as a space separator—all at the same time. Looking at the problem from just one side is too restricted. When ordering information from just one viewpoint, a similar problem exists: one Classification is not enough for all information needs.

Classification systems are in wide-spread use and seem to fulfill an important information-ordering role in the BC industry. One single Classification system is, however, not enough for the fine-grained identification and ordering of information needed. Handling more than one Classification at the same time (to get a better combined 'resolution') is a lot of work and probably not feasible to do consistently. An ICT approach is not, however, restricted in this way and is able to present multiple Classification views[1] on the same information, using Classification as a user-friendly view on the data.

3.2.2 Specifications

A building Specification[2] (Dutch: *bestekstekst*) is usually understood as a central document in a building process. It, traditionally, functions between the design phase and the actual construction phase. It is a contract document, part of the Tender Documents, detailing the agreements made between the Client and the Contractor.

Content of specifications

Essential properties of the Specification text are the formal description, (references to) conditions and regulations, a Classification structure and references to Specification drawings [6].

[1] A representation.

[2] Technically, a Specification consists of both specification *drawings* and the specification *text*. In this dissertation Specification is used as a shorthand for 'specification text', as is common.

Formal description The formal description is build-up from a list of Specification items. Historically, one Specification item often deals with something that has to be budgeted. For such an item for instance the required end-result, required quality and source material is described.

Conditions Conditions (or regulations) give extra information on top of plain technical data (like fire resistance = 30 min). Conditions can be technical or administrative and standard or additional, explained below.

The standard (technical or administrative) conditions are typically valid for every building project. Standard administrative texts make sure that contract-wise a lot of commonly used safeguards are included. The correct legal terminology is used to invoke protection under certain laws. This way, what needs to be said is said simply by including it automatically in the Specification. Typically, the standard conditions are available pre-printed in book form and simply included with the rest of the Specification.

Additional administrative conditions describe the administrative conditions that are specific to this project. Delivery time, payment agreements, steering of the project. Additional technical conditions describe things like delivery of samples for the Client to agree upon.

References to Specification drawings Normally, the references to accompanying Specification drawings are not elaborate. The 'doors on the ground floor' are described. Also you can describe a set of doors, mentioning 'placement according to drawing'. These references are all textual.

Use of Classification in Specification

One common way of subdividing the textual Specification is subdivision into parts called chapters. Traditionally chapters often correspond with branches of the industry or types of work. All paint work

is in one chapter, groundwork in another and doors & windows in a third. This makes it easier to provide a cost estimation by allowing different experts to estimate their part. This kind of subdivision is common in the housing and utility construction sector, traditionally subdivided in specific crafts.

On the down side, much information gets scattered all over the place when there are Specification items that impact more than one kind of work. For instance, an inner wall with masonry, a door and wall finishing might end up in three different chapters [7].

A second common way of subdivision is by following normal execution patterns. Reason for this is that detailed cost estimations (in the ground/water/road sector) are normally made that way. A good match between cost estimation and Specification text is desired.

The Classification structure used by the Specification is sometimes also used to structure other information. Links, made that way, are however on chapter/section/subsection level, not really on the level of actual Specification units.

Types of Specification

There are—roughly—two kinds of Specifications: result Specifications and functional Specifications. Result Specifications specify in detail what the desired end-result is. Functional Specifications specify functional requirements on certain parts. A functional Specification allows the Contractor more decisions.

A mix between result Specifications and functional Specifications is possible. In the Dutch building Specification system it is, for example, normal to see functional requirements for the installation side of a building (heating, ventilation), with the rest of the Specification being a very detailed result description.

Functional Specifications can also be seen as dynamic documents. An initial high-level functional Specification is made and, throughout the design process, more detail is worked out. In this process, Specification items are often coupled with initial cost estimates that become more and more reliable with the increase in detail.

Challenges for specifications

Traditional Specifications are static. Changes do occur, but usually meet resistance, which hinders a dynamic connection between the Client's wishes and the eventual finalised project. Contractors are often selected on lowest price and often look for inevitable errors or omissions in the Tender Documents (and more specifically the Specification) for sources of additional work. This contract document—in reaction—is nailed down as much as possible. The concept becomes everybody's boss instead of being an effective agent for collaboration. [8]

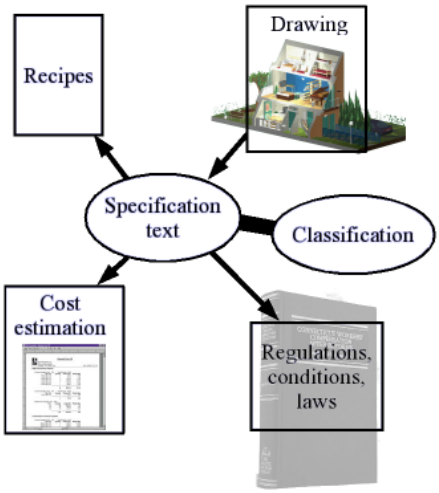
Premature nailing-down of what amounts to uncertainties costs a lot of money. Traditionally, a finished design is distilled into a Specification drawing and Specification text. The design often is changed to a bigger or lesser degree during construction. But especially the Specification text contains a lot of product descriptions that aren't certainties, as realities encountered during the actual construction phase often make changes necessary. This information is prematurely set in stone, leading to friction and reduced flexibility in the resulting construction phase; despite the fact that missing out on flexibility can mean missing out on reduced costs and that friction can mean increased costs.

The role of Specifications in the future BC industry can go two ways. The importance of the Specification as a separate document can become lower, as a good product model should be able to export a lot of information normally contained in a Specification, though probably not all. On the other hand, a Specification can be seen as a set of agreements between two parties. When the process becomes much more dynamic, the number of agreements (as opposed to the number of 'plain' building data) is bound to go up, strengthening Specification's position. This needs more detailed research, part of which is done in the next chapter.

Conclusions on specifications

Specifications have a quite central place in the building process, with many information links to other documents which are owned by many

Figure 3.1: *Specification text, connected. Cost estimation and recipes are examples of applications that want to connect to the Specification text.*



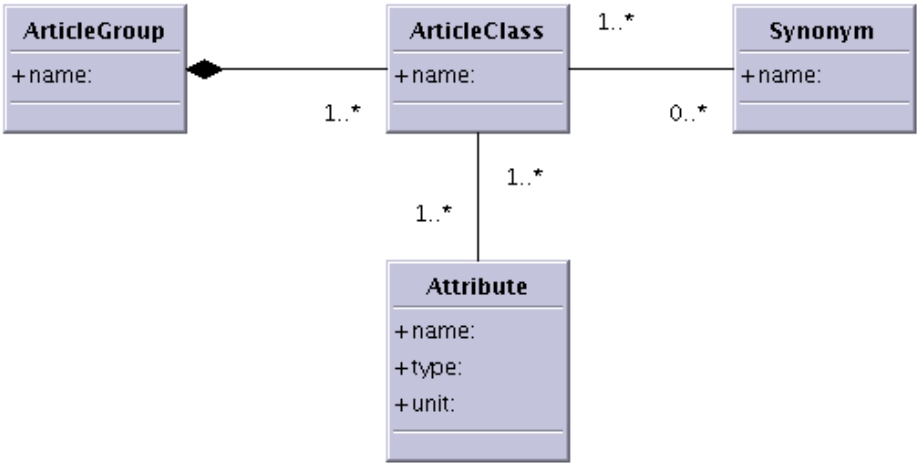
different parties (figure 3.1). This makes it an ideal target for research on improved instrumentation, as Specifications naturally include a lot of interaction between various sources of information. Links exist to: Classification systems, (Specification) drawings, laws, regulations, costing applications, etcetera.

In a traditional construction process, where a Specification is a one-off textual document, reasonable simple textual Specifications are adequate. In a dynamic process, though, the Specification is, initially at least, much more a work-in-progress. This dramatically raises the cost of having to extract the Specification's information manually, should the Specification still be purely text-based.

3.2.3 Two Dutch examples

In this section, two Dutch systems are discussed. The first is ElectroTechnical and Installation Model (ETIM), a Classification system from the electro-technical and mechanical sectors. The second is STABU's building Specification system. The goal is to improve the analysis by observing two real-world examples.

Figure 3.2: *Simplified UML diagram of the ETIM data model. ETIM's data is collected in article groups that consist of article classes. An article class is something like 'earthed single electrical socket'. Each article can have synonyms and a number of attributes.*



ETIM: electro-technical and mechanical classification

As an example of a current Classification-based system, ETIM is developed by the electro-technical and mechanical contractors organisation *Ondernemersorganisatie installatiebranche en de technische detailhandel* (UNETO/VNI) from the Netherlands, but now also internationally. ETIM uses a very simple structure to hold items with synonyms, an explanation, a few attributes (like 'voltage') plus units, but without parts. See figure 3.2.

This simple structure sees very widespread use in the electro-technical and installation sector, mostly as a basis for catalogs and catalog lookups. This usage level attests to the usefulness and acceptability of the Classification system.

Interesting to note is that one of the attributes for the items is always 'price'. A common conception is that this is not something that will be accepted by the supply side of the market, mostly due to 'items not being comparable on price' or because 'price is something that has to be negotiated'. Initially, no vendor included the price in

his catalog entries. Until a few 'broke rank', after which it became common to include a price within a few months [9].

The Classification—all items, including attributes—is freely available, at least in the international version[3] [10]. This makes it easy for software vendors to support this Classification.

There are some drawbacks. ETIM does not support easy information sharing and integration outside the current use case of catalog lookups. The current implementation does not (on a technical level) keep track of changes in attributes of items between different versions of the Classification, which means that the biannually Classification update leads to a manual update of many catalogs [9]. Also there is, for instance, no link to the Dutch Specification system. A second restriction is that the Classification only supports single components. As long as it is one item that you can buy, the item fits in: an electrical socket or a light switch. Complex items that consist of sub-components (like a heating system) do not fit in [9].

STABU: specification system

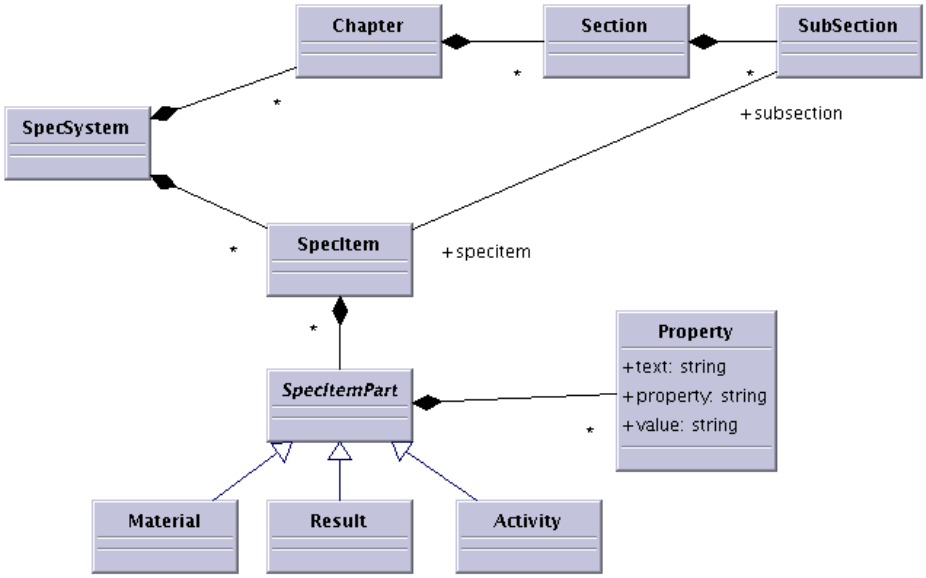
As an example of a Specification system, the STABU system is discussed. This originally paper-based system was given a renewed basis in 1991 with a fully database-based implementation. It successfully penetrated the Dutch Building industry and remains in use till this date.

The fact that an almost 15 year old computer-based system still has a 100% market share testifies to the quality of the original design. Implemented as a relational database, it consists on the one hand of a chapter structure, ordered by kind of work (masonry, painting). On the other hand it consists of Specification items, which are a combination of materials, activities and result descriptions. Specification items have properties (attributes) attached to them. See figure 3.3 on the facing page.

In the Dutch building industry, STABU's Specification system is widely accepted. There are numerous catalogs that offer their catalog items as STABU Specification items. The introduction of the STABU

[3]This does not include the actual product data (the catalogs), though.

Figure 3.3: *Simplified UML diagram of the STABU Specification system structure. The Specification system consists of two parts. At the top is the chapter/section/subsection Classification hierarchy. The second part is a collection of actual Specification items, for instance 'wooden door assembly'. Such a textual Specification item consists of multiple lines of information that specify materials, results or activities.*



system brought semantic integration a step closer. An important reason for acceptance, probably, is the inherent simpleness of the system. As a second reason, STABU's market position is helped by the law-backed requirement for certain government bodies to use a standard Specification system for their building activities.

In order to use the STABU system, a license has to be bought. Likewise, software vendors have to buy a license before they receive the development kit and are allowed to access the system's database.

As with the aforementioned ETIM system, technical restrictions hamper the emergence of ubiquitous information exchange in BC based on this Specification system. On a technical level, like ETIM, attributes of the Specification items are not identifiable. The biannual update therefore means a manual update of a lot of specification items and catalog items. Result is that software vendors are practically unable to link information in a Specification with for instance a costing application.

STABU's Specification system is geared towards generating a paper Specification document. Many other Specification systems are even more explicitly aiming at paper documents as they are a collection of macros for Microsoft Word [11]. These paper documents documents are just representations of the real Specification contents. A system that exposes the real content, instead of a representation, will prove more useful and valuable.

Conclusions on the examples

Both examples, ETIM and STABU, have a quite simple internal structure, which helps in the adoption of the system. In both cases, the implementation prevents—on a technical level—easy sharing of information within the BC process. If this technical restriction could be lifted, it would help both systems to integrate more fully in BC's information landscape.

A closing comment has to be that both systems discussed are database-driven systems. This means that they have a good basis for communication when compared with a lot of other systems. A lot of Specification systems, for instance, are based on word processors

(mostly Microsoft Word) [11] and lack, therefore, any Semantics that can be reused by other applications.

3.3 PDT: information structuring

This section analyses Product Data Technology (PDT) initiatives, starting with ISO-STEP, followed by the STEP split-off IFC. Next are future ISO-standard 12006-3, with two example projects, and lastly reference models.

3.3.1 ISO-STEP

Within BC, the International Standardisation Organisation (ISO)'s STandard for the Exchange of Product data (STEP) has not met much success [12]. This in contrast with other industries like automotive and the oil/gas industry. Two BC ISO-STEP initiatives are discussed below, but some general comments can already be made.

ISO-STEP is a big standardisation-by-committee effort. A lot of effort is put into coordination between the various parts of STEP, making sure that the same definition of 'person' or 'address' is used. The same is true for shape related models, like 2D drawing models and topology and geometry models. Besides generally applicable models, STEP also contains sector specific models, but not for BC. Why not?

An ISO-STEP standardisation process needs strong participants and enough funding. BC is conspicuous for lack of companies with a large market share, being mostly dominated by SMEs. Also, the profit margins do not allow for much research and standardisation subsidies when there is no reasonably direct return on investment.

There is a difference in variety of objects in the various industries. Cars in the automotive industry are largely one product family with a typical set of components. BC by contrast makes houses, stadiums, power plants, bridges: all distinct product families, but of course realised with a limited but common set of materials and basic components. Still it seems that the uniqueness of the products forms a strong negative influence on the information standardisation process. Also the fact that houses, components and materials are different in

each country (due to local circumstances like regulations, weather and such) makes it hard to develop standards for international usage.

AP221/EPISTLE

An early example of a successful STEP development is the AP221/ EPISTLE/POSC-CAESAR project developed for the process industry in the late nineties using ISO-STEP technology. The EPISTLE Reference Data Library (ERDL) is a managed collection of process plant life-cycle data that represents information about classes or individuals which are common to many process plants or of interest to many users. The ERDL as published contains about 10500 classes [13] representing core class specialisations for physical objects and properties which could provide a significant reference source to doing projects.

The ERDL data library was made using Excel and Access tables, which was a practical choice at the time (when Internet was still obscure), but now has resulted in a data collection with little or no appeal. The reason for mentioning ERDL in the first place is to give credit to the developers that were advanced in their time, but could not do the job right because a suitable technology was lacking.

BCCM: building-construction core model

The Building-Construction Core Model (BCCM) was probably the most important part of the STEP effort in BC. A core model is a model that describes common objects like beam, column and floor in such a way that more detailed objects can be described by specialisation, *i.e.* a hollow concrete floor can be modelled as a subtype of floor [14].

The first version of the BCCM was a result of the European ATLAS project [15] and subsequent development of ISO-STEP Part 106. Development of a large model full of BC Semantics did not fit in well with the heavyweight ISO standardisation process and progress was slow. This lead to the start of the IFC project discussed in the next section. Following IFC's start, development of BCCM has ceased. [4]

An observation that can be made is that heavyweight standardisation processes are not suited for the development of a standard library for BC. Perhaps a standardisation process is not the right fit

in principle for a development effort. A standard development efforts should standardise what is available and what has been proven in practice (*de facto*), it should not be created as a *de jure* standard, hoping for acceptance-because-it-is-a-standard.

3.3.2 IFC

The International Alliance for Interoperability (IAI) [16] started the development of the Industry Foundation Classes (IFC) [17] on the results of the aforementioned Building-Construction Core Model (BCCM). Since then IFC is developed in rapid cycles.

IFC is a modeling effort of the ISO-STEP type. They do not operate within the ISO standardisation effort, though, as that was deemed too slow. It was started by Computer Aided Design (CAD) vendors, intending a much more rapid progress. Progress has indeed occurred and work on STEP standards like BCCM has ceased.

As it was started by software vendors, software support for the standard is available. It is mostly restricted, however, to the big CAD packages and a handful of model checkers etc. Result is that only a very small part of BC uses IFC in real practice, it is mostly restricted to the top 1% biggest companies [18]. This makes it uninteresting for the majority of software vendors and thus restricts IFC in its adoption by the market.

The information ordering allowed by IFC is restricted by the amount of BC Semantics ('door', 'windowsill', 'fire resistance') contained inside the IFC model. Currently a workgroup (XM7) is researching the possibility to split IFC into a core IFC and a separate library containing the Semantics: construction objects and properties currently built into the IFC model itself. The 12006-3 standard, discussed in the next section, is one of the candidates for storing the Semantics. Since halfway 2006 this work is underway using the name 'IFD library for buildingSMART' [19].

Something that speaks for the attractiveness of IFC is the fact that researchers doing roadworks engineering have used IFC for their (essentially unsupported) data: for roads [20] and for concrete bridges [21]. The road- and waterworks industry is badly under-provided by

information standards, worse than the building industry. IFC proved to be useful and extendable.

The IAI at last did understand the need for exploitable results. That has been the reason for the funded, controlled development environment outside the standardisation arena where no financing is involved. Despite this understanding and despite efforts of several CAD/CAE vendors, IFC adoption and commercialisation is still on a very low level ($< 1\%$). The reasons are mainly the lack of added value that follows from IFC conformance. Individual CAD-systems like ArchiCad, ADT and others internally have more power than IFC, so why should users or user groups settle for less?

IFC will receive a boost, as recently a big USA governmental building organisation made IFC usage mandatory for their future construction projects (from 2006 onwards) [22].

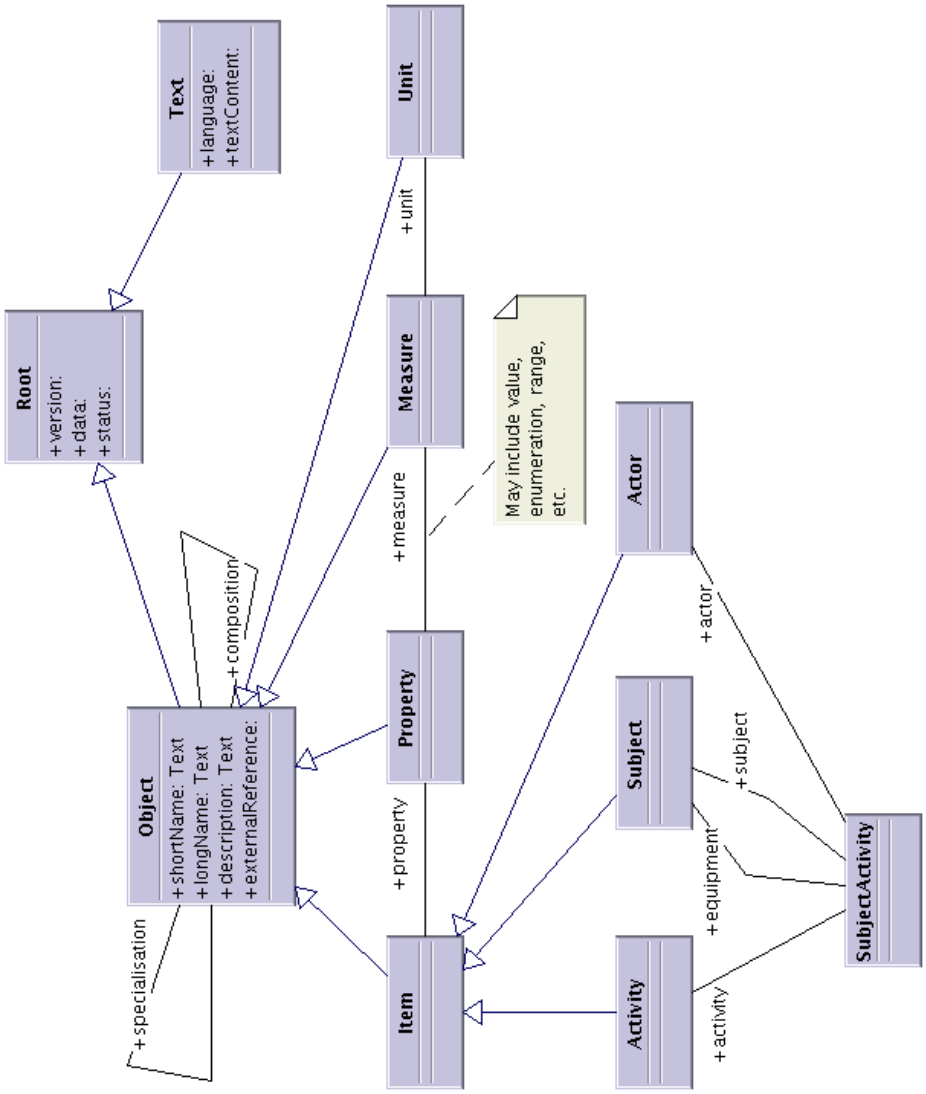
3.3.3 12006-3 taxonomy development

Since mid 1990s, work is underway to create an 12006-3 ISO standard for object libraries [23]. This concerns the data standard, the actual contents of the object library are outside of the ISO process. The standard is developed using STEP technologies, most notably the Express data format specification and the ExpressG diagram format. Currently (halfway 2006) the standard is in the Draft International Standard (DIS) stage. Figure 3.4 on the next page provides a simplified overview.

The object library format allows for objects, properties, units and so on. All items—including properties and units—are uniquely and rigorously identified with an ID, which is a real improvement over the Classification and Specification systems described earlier in this chapter. Every item can have names in multiple languages, likewise multilingual textual descriptions [24]. This provides a basis for international object library development, when desired.

Two mechanisms are available for structuring information: inheritance (*is-a*) and composition (*has-a*). With the inheritance mechanism, an inheritance tree can be made with more abstract items on top, branching out to more specific items further down. For instance,

Figure 3.4: *Simplified UML diagram of ISO 12006-3's structure. The top of the diagram shows the common elements of each object: names, description, possibility of specialisation and composition. Main items of the model are activities, subjects and actors, all of which can have properties that are measured in a certain unit. 'SubjectActivity' is an example of the possibility to tie items together. Not clearly shown in this simplified diagram is the possibility to translate the names and descriptions.*



a 'door' *is-a* 'access construction' or a 'highway' *is-a* 'road'. With composition, typical components can be attached to an object. For instance, a 'house' *has-a* 'roof', a 'foundation'.

12006-3 is a very focused effort, providing only a way to build object libraries for BC. This is contrary to, for instance, the IFC project, where Semantics (the object library's content) are mixed in with the rest of the standard. It is good to keep BC Semantics separate [25] from, in this case, shape modeling. This is evidenced by IFC's start of the XM7 project which aims at separating Semantics out of the IFC model, possibly by using 12006-3. XM7 has recently been superseded by International Framework for Dictionaries (IFD) [26]).

Though a focus purely on the object library level is a good thing, the accompanying data level (where the library objects are used) has received virtually no attention. Only the eConstruct project—discussed in the next chapter—really used objects in catalog items, objects taken from an 12006-3 object library (STABU's LexiCon) [27].
This lack of data-level usage is a two-fold risk. First there is little feedback from actual data on the object library; if there is a continuous effort to create, for instance, catalogs based on the object library, then this gives insight into weak and strong points of the current objects contained in the library. Second, it is harder to show the effectiveness of such an object library, since it cannot be demonstrated. To safeguard against this weakness, it is advisable to do at least a prototypical walk-through all the way from the object library to actual catalogs or Specifications [28].

Though being much more specific and providing *much* more computer-usable information than common Classification and Specification efforts, there are a few weaknesses relating to an improved information and knowledge handing in BC. First, the reference mechanism by which external information sources (like regulations) can be referenced is text-only. It is to be acknowledged that there is not a lot of rich information with which to link in a more semantic way, so there was probably not really an alternative for this textual solution. Second, there is no built-in way to connect two 12006-3-based object libraries, apart from using aforementioned textual reference mechanism. This means a built-in assumption that one

single 12006-3 object library will be used for BC, or at least for a large part of it. Third, properties can only have normal values, they cannot have another object as their value. This restricts the richness of the object-to-object relations to the build-in inheritance and composition relation.

Next, two 12006-3 examples from the Netherlands will be analysed.

CROW object library

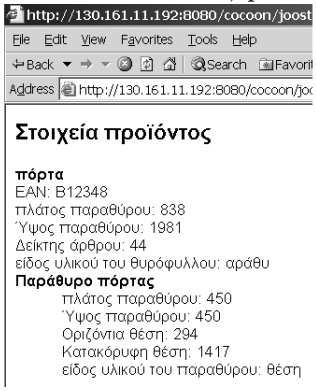
CROW publishes the Dutch Specification system for civil engineering works [29]. CROW is participating in the development of structures and guidelines for object libraries and is working closely together with STABU in these matters.

CROW is developing its own object library for works of civil engineering constructions (the CROW object library, 'CROWOB' in Dutch). Lately CROW changed its top-down, manual approach in filling the object library for works of civil engineering constructions, supported by the LexiCon Explorer, into a—on the one hand—more bottom-up, semi-automated and—on the other hand—more practice driven approach. The top-down approach has shown too little progress in the past few years. It did not result in a usable hierarchy and each discussion provided a new kind of hierarchy without a clear advantage over the previous one and without providing clarity on the lower levels containing the actual data. Also the manual approach resulted in re-thinking lots of definitions that were already available, albeit possibly less consistent and in a less structured way. Overall, this stood in the way of getting any substantial content at all.

In the first step along this new approach CROW has transformed their Dutch 'nomenclature for roads and traffic' into the first initial version of the CROW object library (version 0.1, beginning of 2005). This resulted so far in approximately 2500 object definitions and many more relations, with the capability of an XML export according to the 12006-3 format.

The next step, started in 2005, is the evolution to version 0.2. One element is to work from projects from construction practice. Another

Figure 3.5: *eConstruct Greek content, provided by the LexiCon.*



element is that the next version(s) will also contain the relations, amongst others, with CROW's Specification system.

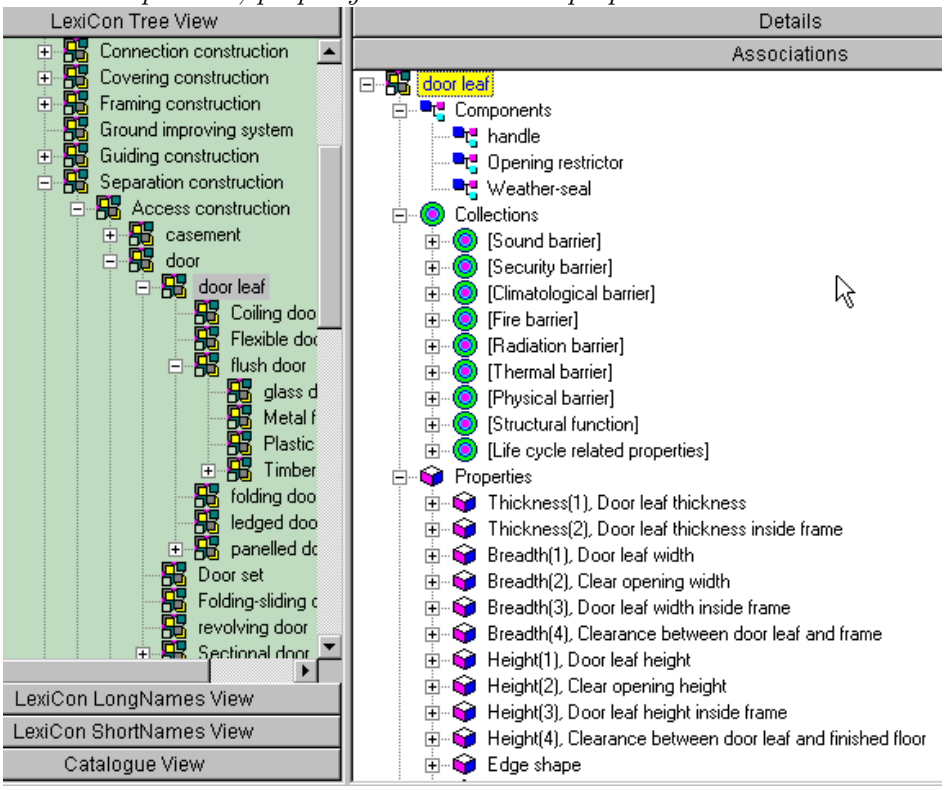
Conclusion: acceptability of a top-down hierarchy is apparently low when its advantages cannot be clearly demonstrated. The bottom-up approach tries to build on existing data, making it more semantically rich. The speed with which this bottom-up approach could be executed has much to commend it, though the results have not proven themselves in practice yet.

STABU LexiCon

STABU's LexiCon has always been directly linked with the core 12006-3 development through its main developer: Kees Woestenenk. It has therefore all 12006-3's advantages like multilingual possibilities and strict modeling. The LexiCon is consequently bi-lingual (English and Dutch), in the sense that those two languages are kept updated and synchronised. Lexicon's content during the eConstruct project was translated in no less than six languages [30], see figure 3.5 for a Greek example.

In the most recent version only the inheritance mechanism is used [31]; the other object structuring possibility, composition, is reserved for future commercial enhancements. LexiCon therefore has as its backbone a huge single-inheritance hierarchy, part of which is shown in figure 3.6 on the next page. As everything has to fit in that one

Figure 3.6: *STABU LexiCon explorer interface. The left hand pane shows part of the inheritance hierarchy, the right hand pane shows the selected item's components, property collections and properties.*



hierarchy, a sizable number of objects are by necessity abstract. A stated goal of the single inheritance hierarchy is improved consistency.

For the LexiCon, 12006-3's lack of data format was addressed in the eConstruct project (2000-2002, discussed in the next chapter). Delft University of Technology and TNO developed the Building-Construction XML (bcXML) data format, which was subsequently used for a convincing demonstration. Catalogs and IFC drawings were linked together, with the LexiCon Taxonomy as the Semantics used by the bcXML data.

There was no follow-up on the bcXML possibility and attention was shifted to updating the hierarchy, which resulted in multiple re-

visions. These revisions have not yet obtained market acceptance, one of the reasons being that there are no demonstratable results [18] [32].

The recent LexiCon versions do not offer the possibility to attach properties like 'fire resistance' and 'door leaf height' to the objects, neither do they indicate parts. The LexiCon is seen as a basis upon which you can build libraries [18]. The creation of commercial object libraries, which include attached properties and parts, can then be done on top of the base LexiCon.

Improved information handling in BC means a more streamlined exchange of objects and properties. Keeping the (linked) properties out of the core and entrusting that to multiple commercial object libraries does seem problematic, especially since 12006-3 does not include a mechanism to connect multiple object libraries.

3.3.4 Reference models

Other research in the past looked into a method that could produce product models for certain domains that required multiple sub models. An example has been the EPISTLE development of the international process plant industry. Basically the idea is to create a common meta model for a number of overlapping detailed models.

A well known reference model is the GARM [33] [34]. The purpose of the GARM was to develop dedicated type models for specific product classes (office buildings, highways, bridges) that all have a similar structure.

The approach advocated in the GARM is particularly suited to the problems of BC. The GARM supports: the distinction between a functional perspective and a technical perspective; top down and bottom up problem solving; and a layered approach.

Many researchers have applied GARM principles to create a reference model with properties that were required. Luiten [35] for example created the Building Product Model (BPM) that could support design for construction.

Until recently, the influence of the reference modelling approach has been very limited. It seems that the technology of the past and

the traditional standardisation process lacked sufficient power for successful implementation of this approach.

3.3.5 Conclusions on PDT

In a decennium the BC PDT researchers have not advanced much. Only the IAI has something tangible. The development of ISO-STEP standards for BC is abandoned. The promise of semantic integration could not be realised. The Classification system people infiltrated in the PDT arena but synergy did not follow.

Tolman analysed [12] why the PDT approach of ISO-STEP did not (yet) deliver BC what it promised. Basically (1) the expressiveness of the standard has been too small, (2) the time to market of the standard and its related tools has been too long and (3) the standard lacks openness (cannot be developed without heavy involvement of ISO-people and ISO ownership). While developing and implementing the standard, the technology changes and changes.

The IAI with its IFC scores somewhat better; a fraction of the current BC projects uses some IFC technology. What has been missing in ISO-STEP and what still is lacking in the IFC is Semantics, *i.e.* definitions of the 300k+ objects of interest playing a role in BC, including properties, units, relations, constraints. As to openness: the IFC and its submodels are controlled by the IAI, which is not really much different from ISO. It is possible to develop your own extensions, though.

3.4 Evaluation regarding improved information sharing

This section takes the analysis of the previous two sections and evaluates the suitability for improving information sharing. Any shortcomings are translated into research opportunities.

3.4.1 Evaluation of suitability of classification and PDT efforts

The Classification and Specification efforts are widely accepted in the market. Their market adoption is an indicator that having a simple structure greatly helps market adoption.

Except purely textual, there are no links to other information sources. You can refer textually to 'the doors on the fifth floor' in a Specification, but you cannot link directly to an item in a drawing. This largely prohibits information sharing in any non-textual way. Likewise, information contained in a Specification or Classification cannot be used by other applications, as—apart from numbered chapter headings—no computer-identifiable information is available.

From all PDT efforts discussed above, only IFC was adopted by the market. Mostly because it was started by CAD software vendors, which therefore implemented it in their products. The use of IFC in the market is, however, low. One reason is that only the more elaborate CAD packages include IFC capabilities and that not everybody uses the total functionality offered. A second reason is that only a small portion of the BC companies use the elaborate project databases needed to get the most out of IFC.

One reason the progress is slow is because of committee-based standard development; a new standard is *developed* in a committee, instead of standardising existing best practices. STEP showed that this does not work well for BC. IFC is a bit better in this regard. The amount and diversity of objects in BC makes it probably unworkable to define those objects inside a formal standardisation process as part of the rest of the model. IFC discovered this and is looking at ways to keep the object definitions (the Semantics) external.

The 12006-3 development provides a way to make object libraries. The idea to store object definitions in a separate object library is probably good. 12006-3, however, has some weaknesses that limit its use for improving information sharing in BC. There is no way to link two object libraries, which means that object definitions have to be concentrated in, preferably, *one* single object library. As the most important object structuring mechanism is inheritance, a huge

tree structure is the result, in which all the various objects have to fit. Also, there is no exploitation follow-up: no data format or application that uses the object library's definitions, so no feedback from real data and no demonstratable use. This has, for example, prompted CROW to take a more pragmatic approach and stop with the top-down tree structure that was permanently in state of revision and that could not show its effectiveness.

As a last point, we can conclude that extensibility is good. IFC can be extended and therefore allowed itself to see experimental use for road construction, an area not originally intended.

3.4.2 Research opportunities for improvements

In addition to the functionality offered by the technologies discussed in this chapter, further advancements are needed for improved information sharing in BC.

Improved information linking IFC and 12006-3 both allow computers to identify the individual items contained in them: you can identify individual objects and attributes. Ideally, there would be a generic way in which you can actually point from one information item to another. For example, from a Specification item to the corresponding item in an IFC model. Research is needed on how to connect—in a generic way—various information items.

Generic object library support in data formats Having a separate object library[4] is a good idea, but its definitions should be usable in a generic way in a variety of data formats. Research is needed on how to connect—in a simple way—object libraries and data.

Generic accessibility of information Something not really covered by any of the technologies discussed: how to access the information. IFC tends to prefer a central, Express-based project

[4]In the rest of the thesis, the term Ontology will be used. That is a more correct term than 'object library' and more in line with Internet terminology.

database. This is not something that scales to small projects and it is not a method that is applicable in general. Research is needed on how to make information generally available in a way that scales also to small projects.

Improved market acceptance Apart from the two Classification/ Specification examples, widespread market acceptance of technologies that provide improved information sharing is problematic. Research is needed on how to improve the chances of market adoption of information sharing technologies.

3.5 Conclusions

The mainly paper-based Information System (IS) of the BC industry is not able to provide the information and knowledge sharing capability requested.

Also current first generation ICT support fails. Classification and Specification systems are widely accepted in the market, but do not offer enough information richness to adequately support information sharing in BC.

The next generation instruments should promote transparent ways of storing information; provide for re-use of information and knowledge; support co-operation between applications; and have improved market acceptance.

Some of the principles explored by the research on reference modelling might be worthwhile exploring.

Bibliography

[1] ISO 12006-2:2001, November 2001. Available on-line at `http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=35333`.

[2] Bjørn Bindslev. *Paradigma - a product data model*. CBC systems, 1995.

[3] Reinout van Rees. Clarity in the usage of the terms ontology, taxonomy and classification. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at `http://vanrees.org/research/papers/Cib78ConferencePaper2003`.

[4] Sander van Nederveen. *Object Trees - improving Electronic communication between participants of different disciplines in large-scale construction projects*. PhD thesis, Delft University of Technology, 2000. Available on-line at `http://repository.tudelft.nl/file/117347/098324`.

[5] Charles M. Eastman. Modeling of buildings: evolution and concepts. *Automation in construction*, 1(2):99–109, September 1992.

[6] J.O. Zijlstra. *Bestekken in de Grond-, Water- en Wegenbouw*. C.R.O.W., 2e druk, 2e oplage edition, 1995.

[7] Personal communication of the author at STABU, 2004.

[8] Hennes de Ridder, Roland van der Klauw, and Ruben Vrijhoef. Het nieuwe bouwen in Nederland. Technical Report 2002-BPI-028, TNO bouw, December 2002.

[9] Personal communication of the author during a visit to UNETO/VNI, 2003.

[10] Datenmodell ETIM Klassifikationssystem version 2.0, 2002. Available on-line at `http://www.etim.de/`.

[11] John Gelder. ICIS product survey comparison, November 2004. Available on-line at `http://www.icis.org/siteadmin/rtdocs/images/8.doc`.

[12] Frits Tolman. Product modeling standards for the building and construction industry: past, present and future. *Automation in Construction*, 8(3):227–235, February 1999.

[13] EPISTLE Reference Data Library overview, November 2001. Available on-line at http://www.btinternet.com/~chris.angus/epistle/specifications/erdl.html.

[14] Edwin Dado. *ICT-enabled communication and co-operation in large-scale on-site construction projects*. PhD thesis, Delft University of Technology, October 2002.

[15] Patrice Poyet, Guillaume Besse, Eric Brisson, Philippe Debras, Alain Zarli, and Jean Luc Monceyron. STEP software architectures for the integration of knowledge based system. In *Conference proceedings*. CIB-W78, 1955.

[16] IAI international website, 2004. Available on-line at http://www.iai-international.org/.

[17] Lachmi Khemlani. The IFC building model: A look under the hood, March 2004. AECbytes newsletter, available on-line at http://www.aecbytes.com/feature/IFCmodel.htm.

[18] Jasper Feenstra. LexiCon expert panel discussion, October 2004. Available on-line at http://vanrees.org/weblog/1098802937.

[19] IFD library for buildingSMART website, 2006. Available on-line at http://www.ifd-library.com/.

[20] Tamer El-Diraby. e-infrastructure: An interoperable GIS system for infrastructure decision making. In Žiga Turk and Raimar Scherer, editors, *Conference proceedings - E-work and e-business in AEC*. ECPPM, Balkema, 2002.

[21] Nobuyoshi Yabuki and Tomoaki Shitani. An IFC-based product model for RC or PC slab bridges. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at http://itc.scix.net/data/works/robots/w78-2003-463.htm.

[22] IAI North America. Planning IFC building info model, GSA contracts with NIBS/IAI, September 2004. Available on-line at http://www.iai-na.org/news/092204.php.

[23] ISO/DIS 12006-3.2, 2005. Available on-line at http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=38706.

[24] Kees Woestenenk. The LexiCon. In Žiga Turk and Raimar Scherer, editors, *Conference proceedings - E-work and e-business in AEC*. ECPPM, Balkema, 2002.

[25] Nicola Guarino. Formal ontology and information systems. In Nicola Guarino, editor, *Formal ontology in information systems*. IOS Press, 1998. Available on-line at http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/FOIS98.pdf.

[26] Expressway newsletter, 2006. Available on-line at http://www.epmtech.jotne.com/newsletter/ew106/oneworld.html.

[27] Reinout van Rees, Frits Tolman, and Rèza Beheshti. How bcxml handles construction semantics. In *Conference proceedings - Distributed knowledge in building*. CIB-W78, 2002. Available on-line at http://vanrees.org/research/papers/Cib78ConferencePaper2002.

[28] Reinout van Rees, Wouter van Vegchel, and Frits Tolman. Practical use of the semantic web: lessons learned and opportunities found. In Attila Dikbaş and Raimar Scherer, editors, *eWork and eBusiness in architecture, engineering and construction*, pages 329–336. ECPPM, Balkema, September 2004.

[29] Website CROW. Available on-line at http://www.crow.nl/.

[30] Frits Tolman, Michel Böhms, Celson Lima, Reinout van Rees, Joost Fleuren, and Jeff Stephens. eConstruct: expectations, solutions and results. *ITcon*, Special issue on European projects, 2002. Available on-line at http://itcon.org/2001/13.

[31] Personal observation of the author at STABU of the development of the lexicon in 2003 and 2004., 2004. At the moment (late 2006), the latest inheritance hierarchy can be found on-line at http://www.stabu-lexicon.com/.

[32] Jasper Feenstra. STABU LexiCon - diffusion and adaptation of a new information standard. Master's thesis, Twente University, 2004. Available on-line at `http://vanrees.org/research/lexiconinfo/TheLexiCon.pdf/view`.

[33] Wim Gielingh. General AEC reference model. Technical Report BI-88-150, TNO, 1988.

[34] Wim Gielingh. General AEC reference model (GARM) an aid for the integration of application specific product definition models. In Per Christiansson, editor, *Conceptual modelling of buildings*, W74+W78 workshop, Lund, Sweden, October 1988. CIB. Available on-line at `http://itc.scix.net/data/works/robots/w78-1988-165.htm`.

[35] Bart Luiten, Thomas Froese, Bo-Christer Björk, Rachel Cooper, Richard Junge, Kari Karstila, and Robert M. Oxman. An information reference model for architecture, engineering, and construction. In *Conference proceedings*, w78 conferences. CIB, 1993. Available on-line at `http://itc.scix.net/data/works/robots/w78-1993-2-391.htm`.

I can be googled, therefore I am.

4

Analysis of new applicable Next Generation Internet technologies

This chapter analyses the applicability of the NG Internet for BC and analyses the technologies involved. The goal is to extract the technologies that can be most useful in improving BC's value-adding performance.

After the chapter's *introduction*, the Internet as *generic communication medium* is analysed. Next, the *basis for computer-understandable data* and computer-involving web services, followed by the application thereof in the *EU eConstruct project*. Internet-based formality is provided by the *generic vocabulary: OWL*, analysed next, followed by thoughts on *development and open source*. To close off the chapter, the *suitability of NG Internet for BC* is analysed, followed by the *conclusions*.

4.1 Introduction

The previous chapters concluded that the information and knowledge sharing capacity of the BC industry lags behind (see section 2.4 on page 20), and that the current ICT support is insufficient to improve this capacity (see section 3.5 on page 48). The next generation instruments should promote improved information linking, generic Ontology support in data formats, generic accessibility of information and improved market acceptance.

In recent years, the Internet has enabled an almost unprecedented increase in available information, also it has seen great market acceptance. Following the success of the human-readable world wide web, data sharing and knowledge structuring standards have sprung up around the Internet, prompting the use of the term 'Next Generation Internet (NG Internet)'. This NG Internet seems an attractive research target to analyse on its applicability for BC.

A large part of this chapter is structured around the notion that communication requires a *communication medium*, a *grammar* and a *vocabulary*. All three, looking at the (next generation) Internet and BC, are analysed in turn; with an analysis of eConstruct, which used many of the technologies, as a benchmark.

With actual market acceptance being important, an analysis is made of the development style favoured on the Internet and the open source licensing that has gained much prominence.

The first section starts with the Internet's basic technologies.

4.2 Generic communication medium: the Internet

The two base characteristics of the Internet[1], as analysed here, are its generic access method (`http://...`) and generic addressing mechanism (`vanrees.org/research/`). Their analysis is followed by an

[1]The term 'the web' could be more accurate, as that indicates the most recent additions to the more than 20 years old Internet: HTTP, HTML, XML. The Internet also includes the TCP/IP protocol, FTP, email, etcetera.

analysis of the original textual nature of the web (HTML). Resource Description Framework (RDF) is the last technology analysed in this section: the most recent addition that enables even richer communication.

4.2.1 Identifying and addressing everything: URL

Internet addresses are URLs, for example `http://www.tudelft.nl` and `http://en.bcoweb.org/bridges/overpass`. The first identifies the homepage of Delft University of Technology, the second is a definition for 'overpass' in an Ontology. `http://objecttree.org/delft-hospital/1st-floor/door/42` could indicate door #42 on the first floor of a hospital in Delft, inside an object tree web application. Every one of these addresses is unique: an Internet address effectively functions as Globally Unique ID (GUID).

In practice, address space is unlimited. There is no technical limitation on what can be uniquely identified.[2]

When addresses are unlimited, every relevant BC object can receive its own address. This might sound like a lot of work, but for applications—who will do it most often—it is no problem: computers are very patient. With no technical restrictions, attention can be shifted entirely to the needs of BC.

There is no technical reason not to identify every single relevant object if doing it is possible. If the software employed stores the individual objects, it has a built-in mechanism to identify these objects; this can be converted to an Internet address.

There can be non-technical reasons, however. What is the needed level of detail? Not every nut and bolt is interesting, perhaps the detail level should be 'all the doors on the first floor'. But, when an individual door should be replaced, that specific door should be identifiable.

To illustrate the conceptual idea, lets take an IFC file. Normally,

[2]The only thing that is theoretically scarce is the domain name (like `vanrees.org`). Everything that comes after the `http://vanrees.org` is totally free, though. In practice, every current project of 10 houses or more already has its own web page, if only to interest prospective buyers.

the IFC data *as a whole* would be accessed through a single STEP database. Using the Internet, `http://example.org/project2` could identify the entire IFC file. `http://example.org/project2/floor/2` would be a data file with every object on the second floor. `http://example.org/project2/objects/door` could mean all the doors in the building. `http://example.org/project2/objects/door/5`, lastly, could be the individual door with id `5`.

The increase in computer-supported communication possibilities is striking when comparing a BC industry using above method to the current BC that uses a purely text-based Specification, for example, that does not allow fine-grained object identification and access.

The nature of the Internet is—amongst others—the lack of centralised organisation: it is an endpoint-to-endpoint medium [1]. In the building industry, endpoints can be either *actors* or *documents*. This build-in focus on endpoints is a good fit with the building industry, as it consists of a lot of small companies. Every company or project can be as much an endpoint as another.

4.2.2 Access method: HTTP

When you visit a web page (like `http://www.tudelft.nl/`), your web browser internally uses the standardised HTTP access method to request that page. With this single request method, every web page on the Internet can be retrieved.

There are four standard access methods: retrieve info, delete info, update info and add new info. These four methods are typically enough for all applications. Retrieving or deleting information is typically straightforward, inserting or updating is more involved.

HyperText Transfer Protocol (HTTP) defines basically four operations on Uniform Resource Locators (URLs). Of those four normally only `GET` is used. If you request a certain page in Firefox or Internet Explorer you technically send a `GET` request to the web server hosting that page. Like SQL's `SELECT`, `UPDATE`, `INSERT` and `DELETE` on database rows, HTTP offers `GET`, `PUT`, `POST` and `DELETE` on URLs [2]. `GET` a door catalogue item, `PUT` a different door in your project's data.

Using HTTP's four access methods for accessing BC data also adds

the following benefits:

- HTTP authentication (username/password protection).
- `https://` secure connections (also used for Internet banking).
- HTTP caching mechanisms and load balancing.

HTTP's four access methods are used for so-called web services [3], explained more fully in the next section. The currently much-hyped Simple Object Access Protocol (SOAP) approach is seen as an alternative to HTTP. With SOAP you use every application's own access methods, typically easier for the software vendors. The drawback is that this can be done in a large number of ways[3]. The interaction complexity of this alternative rises with the square of the amount of applications, $O(n^2)$ [4] [5].

When using the standard Internet access method, the complexity of retrieving and deleting information is constant for the amount of applications. The interaction complexity of updating or adding information rises linearly with the number of applications, $O(n)$.

The BC industry is heavily fragmented and the last thing it needs is a large number of custom-made interfaces instead of a completely standardised data access and modification mechanism as offered by HTTP.

With the large variety in applications used in BC, the advantages of using the standard Internet access method are clear—it is therefore recommended for use in the concept solution.

4.2.3 Data protection

When looking at a building project, you can distinguish four kinds of information in two dimensions. The first dimension is whether the information is specific to the project or not. The second dimension is whether the information is specific to a certain company or not. [6] Figure 4.1 shows the four intersections of the two dimensions.

[3]Instead of HTTP's `GET`, you'd have `getDoc()`, `get_document()`, `retrieve()` and so on: a different method for every application.

Figure 4.1: *Information can be specific or not specific to projects and companies. This table shows some examples for all four combinations.*

	Project specific	**Not project specific**
Company specific	Internal cost statements, proprietary designs	Internal regulations, recipes, past projects database
Not company-specific	Client's brief, building specification	Building codes

Building codes are an example of information that is neither project-specific nor company-specific.

Project-specific is the project information. The not-company-specific part of this includes the client's brief, government plans regarding this project, the building Specification, etc.

Company-specific is the proprietary information. The not-project-specific part includes internal regulations, internal recipes-for-work, a database of past projects and so on.

The proprietary information and the project information partly overlap. This is an area where problems loom. Part of the proprietary information could be very handy for the project and therefore for the project partners. But do you want to give them that valuable information for free? Do you want to share it? It might be a competitive advantage in later projects, which you give away by sharing.

Also, part of the project information will be necessary for the company: the specification drawings, the specification text. Essential for coupling with the internal information like work planning. But the project information has to be available to all partners in the project. This therefore means that the information has to be shared, so it either has to be kept in one location or different copies have to be kept synchronised.

The essential property here is that the information has to be

shared between the quadrants. The information should at least be partly accessible.

Data can be protected, on a technical level, using standard Internet means. Data encryption (using the `https://` protocol variant) prevents data snooping. User/password protection safeguards the data.

4.2.4 Information presentation: HTML

The HyperText Mark-up Language (HTML) was invented by researcher Tim Berners-Lee for allowing researchers to easily share their notes and articles [7]. Thus it allowed for headings (in various levels), links (to other documents), quotations, bold, italic, code examples. Later additions brought images, ways of indicating fonts and colors and so on.

HTML's text-oriented nature meant a limited emphasis on syntax and relative greater emphasis on presentation. The syntax available is 'heading', 'paragraph', 'quotation'. Presentation is 'bold', 'italic', 'image'. A text viewpoint is assumed when calling 'heading' syntax: from a data viewpoint, a 'paragraph' with a Specification item is indistinguishable from one with a catalog item. Mixing syntax and presentation makes HTML unusable for data transfer, as opposed to its great use as a textual medium.

The hypertext part of HTML's acronym indicates the familiar links you can click on in HTML documents. Hypertext means you can point towards another text from a starting document. HTML was not the first technology to attempt hypertext; it was the first technology, though, to gain such widespread acceptance. The two main reasons mentioned most often are the simpleness of the format and the one-way linking mechanism.

The simpleness of the format allowed you to do a 'view source' and to try to come up with your own document, see figure 4.2 on the next page. The one-way linking mechanism was a departure from then-current hypertext systems in the sense that the other systems had bi-directional links. The bi-directional links meant internal consistency: no broken links. The drawback, negated by HTML's

Figure 4.2: *Short HTML code example.*

```
<html>
  <head>
    <title>html example</title>
  </head>
  <body>
    <h1>Main heading</h1>
    <p>A paragraph with some text in it,
       part of which is <b>bold</b>.
    </p>
    <h2>Subheading</h2>
    <p>More text...</p>
  </body>
</html>
```

use of one-directional links, is that an elaborate coordination mechanism is needed. Allowing everybody to link to every document (one-directionally, without coordination) is clearly the simplest linking mechanism. The rise of the Internet has shown that simpleness's advantages won out over more elaborate, consistent systems [8].

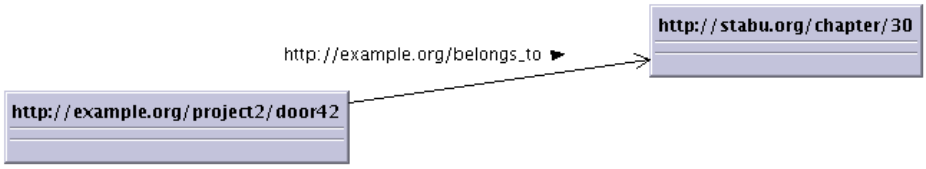
Based on HTML's success, a more data-oriented standard was created: XML, explained in section 4.3.1 on the facing page. Like HTML, XML was a simplification of then-current technologies, done with the aim of securing much greater adoption.

4.2.5 Connected information: RDF

The NG Internet has a generic linking mechanism build-in, the technology is called RDF [9]. If BC would standardise on this linking mechanism, a lot of opportunities would open up.

The linking mechanism is the simplest mechanism possible, as it connects one object with another object using a named relation. Both the objects and the relation are identified using Internet addresses (as unique identifiers). The result is that every object that is identified

Figure 4.3: *RDF example: two objects linked using a named relation. Both the source, target and link are identified with URLs. This means that also the link itself can be the target or source of another link. Source, target and link together form a so-called 'triple'.*



using an Internet address can be linked to another object. As the link is also identified, different relations are possible.

For example, an object in an IFC file (`http://example.org/project2/door42`) can be linked (`http://example.org/belongs-to`) to `http://stabu.org/chapter/30`, meaning that door #42 belongs in STABU's Classification, chapter 30. For a depiction, see figure 4.3.

The data format characteristics of RDF are analysed in section 4.3.2 on page 63.

4.3 Grammar: involving the computer meaningfully

This section analyses two data structuring technologies and a way of interacting with that data. The first technology is XML. The second is the data structuring aspect of RDF, of which the linking mechanism has already been analysed above. At the end, web services are discussed as a means of accessing the information.

4.3.1 Data format: XML

With XML [10], documents are no longer inaccessible for computers. An XML file is a text file, containing information on a certain topic (like 'I sell wooden doors 2 meters in height'), tagged with sort-of nametags that explain what is meant with the text. To say that 'height' in above example is a property, you would do it in an XML

file like this: `<property>height</property>`. The mechanism is analogous to HTML, analysed earlier.

Text is the only truly portable data, binary formats being inherently more difficult to use[4]. XML files, containing all the information you need, can be directly inputted into an application and can be also quite easily formatted for the human eye, either as a web page or in print. So, XML covers both the human and the computer side of information exchange. XML contains the syntax, mark-up can be applied to it: separation of mark-up and syntax.

The syntax of XML documents is defined in a schema like XML Schema Definition (XSD) [11], Document Type Definition (DTD) [10] or Relax NG (RNG) [12]. Such a schema allows applications to know what data to expect.

XML was designed by a workgroup that cleaned up and simplified the existing Standard Generalized Markup Language (SGML) standard [13]. Hypertext was a good idea in general: the small, simple HTML made it attractive and widely-used. Likewise, SGML was a good information structuring idea: the smaller and simpler XML made it acceptable and widely-used. Tools are ubiquitous: parsers, writers, editors.

It is helpful to compare working with XML to working with STEP's Express-based data, as STEP also tries to make computer-readable data available. On the technical level, STEP data is more elaborate and powerful than plain XML. On the acceptance and usage level, XML is way ahead of STEP data standards. The simplicity of XML is probably a big factor, but also the availability of tools. Every computer language nowadays allows you to use XML. When dealing with STEP, the choice of libraries is restricted; free libraries are hard to find, a drawback not encountered with XML. (STEP and XML have different origins and goals, so in that sense, the comparison is not completely fair).

[4]Binary data compared to textual data: ever tried importing a Word document using a plain text editor? Note that the comparison is between textual and binary data formats, not between plain `*.txt` files and `*.doc` files.

4.3.2 Data format: RDF

As it has been shown in the introduction of the previous section, RDF links information items to other information items with a named relation. The source, target and relation are all identified with a URL[5]. Such a three-part combination of source, target and relation is called a 'triple', see figure 4.3 on page 61. With a set of 'triples', everything that be expressed with XML can be expressed by RDF, too. A set of triples is the simplest way to encode information.

An advantage is that using triples is a very flexible method. The flexibility comes from the built-in RDF schema language [14], which allows you to state, for instance, subclass relations. Take as an example a current property called 'is connected to'. In the future, more fine-grained Semantics are needed, so properties like 'is bolted to' and 'is screwed together with' are created. Existing software continues to work just fine, provided 'is bolted to' and 'is screwed together with' are defined as subproperties of 'is connected to'.

The flexibility comes at the cost of loss of predictability of the physical data format. An RDF file contains a set of triples and the order in which the triples are specified (or nested) does not matter. Though RDF is most often stored in RDF's XML representation format, the many different ways in which you can store the same information makes the use of standard XML tools impractical. RDF tools are needed, which are less widespread than XML; RDF tools are available for most computer languages, but they are not as widespread as those for XML.

RDF provides a lot of flexibility, but at the price of some standard XML tools. The alternatives will have to be weighed for individual cases. A solution that gains some prominence in RDF/XML practice is to provide an RDF mapping [15] for XML-based formats [16], trying to get the best of both worlds. An advantage observed in those efforts is that the more formally and mathematically defined Semantics of base RDF force you to think through the model more rigorously.

[5]Technically Uniform Resource Identifier (URI), but the distinction is small enough to use the more familiar term URL.

4.3.3 Accessing the information: web services

Web services are Internet-accessible applications that can be used directly by other applications; web services use the Internet and XML [3]. Web Services enable the bridging of data islands [17], allowing applications an active role on the Internet. For an introduction, see [18].

As analysed in the beginning of this chapter, using 'plain' HTTP access instead of SOAP is advocated. With HTTP, 'webifying' is a useful term to indicate the work needed. Webifying data means that every piece of useful data should have a URL. There is no reason to restrict the number of URLs [19]. The success of the World Wide Web is entirely based on assigning a URL to every web page and image and enabling links between them. Every object on the Internet is accessible directly, be it `http://www.cnn.com` or a page describing the lodging facilities for 2002's ECPPM conference.

Webifying a door catalogue, for example, doesn't mean creating a human-readable page containing pictures and some text listing the available types. Instead it means having your catalogue available at `http://company.co.uk/doorcatalog` and the parts describing the various individual doors at `http://company.co.uk/doorcatalog/door1`. This makes it possible to link to a specific door in your catalogue from the project where they want to use your door.

With XML and RDF, we now have the means to access and to provide *data* using HTTP, instead of just text-oriented HTML pages. For applications that want to use web services to enable their integration in what is sometimes called the 'Internet operating system' [20], what is needed is to choose XML/RDF models to support and to design URLs for their information.

Choosing XML/RDF models Both for information being received by the application and for information provided by the application, a format must be chosen or designed. For this, either XML or RDF should be used, being the preferred standard data formats. For some applications, for instance ifcXML might be a good choice. For others, a self-made Specification format.

Such a self-made format should be well-documented and publicly available to enable interoperability.

Designing URLs This has been covered in this chapter's first section. Take care to make every useful item in the data available using an URL. As a minimum, other applications are then able to download that item's information. More advanced, this opens up the way to delete, append or change the item—from another application.

4.4 Using XML and the Internet: eConstruct

This section analyses the European Union (EU) research project 'eConstruct' [21] that used Internet and XML as central project ingredients, therefore providing a good way to get BC input on those technologies. The section starts off with an introduction on the project, followed by the central vocabulary: the LexiCon. Then the bcXML data format is analysed, with web services next. The section closes with a discussion on the conclusions that can be drawn from eConstruct's experience.

This section represents a large portion of the work that was done in the first two years of this research. It can be considered a case study that evaluates the first part of this chapter.

4.4.1 Introduction

The aim of eConstruct project was to develop a vocabulary called bcXML that would 'know' what is meant by user terminology. A computer would therefore understand what is meant by terms such as 'wall', 'floor', or 'roof' in conjunction with concepts such as 'cleaning' and 'painting', making web-based communication and interrogation possible. In addition, concepts used in bcXML like 'door' and 'height' would be available and translatable into different national languages, Classification and coding systems.

Initially bcXML only provided a limited set of terms, but these were be capable of supporting real end-user cases. The first applications using bcXML focused on the 'shopping phase' of eCommerce, allowing such questions as 'Where can I buy a new boiler and what will it cost me?' to be answered. It was the intention that the second generation of bcXML applications would be more influential and would concentrate more on the underlying goal of the project: to help increase the competitiveness of the European BC industry by supporting business-to-business transactions.

eConstruct as a two-year project did not itself increase the competitiveness of Europe's BC, but it showed the way. The final demonstration was convincing, in this sense that a full scenario was played through: existing catalogs were converted to a Semantics-based model and stored in an on-line catalog server; the taxonomyserver allowed multi-lingual browsing and searching using the Semantics (see figure 4.4 on the facing page); an IFC building model could, using the Taxonomy server's Semantics, query a catalog for matching items and insert the matching item (a door in this case) back into the IFC model. [22]

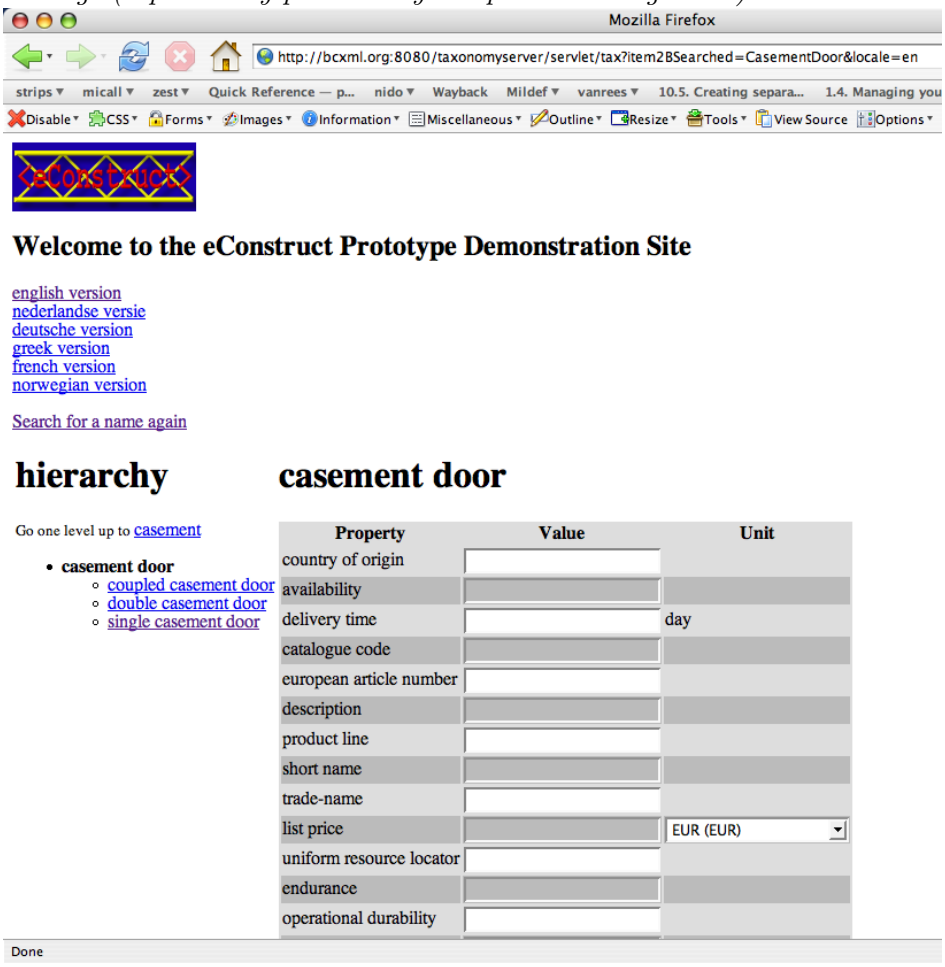
eConstruct results have seen use in other projects, the most recent being an EU project for an electronic marketplace for equipment with the aim of allowing collaborative use of equipment in order to prevent (costly) downtime [23].

This section analyses the eConstruct development and result, to provide feedback on the Internet and XML technologies used. First, the vocabulary used is discussed including the XML format behind it; second the bcXML data format that mimics RDF; third the web services used.

4.4.2 Vocabulary: the LexiCon taxonomy

From the start of the eConstruct project, STABU's 12006-3 development 'LexiCon', called a Taxonomy, was intended as the basis for communication. The most attractive features were: (1) the rigorous formality in specifying objects, properties, all with unique IDs, in the PDT tradition that sets it apart from the earlier Classification efforts;

Figure 4.4: *eConstruct's taxonomyserver, a web application made by the author. It shows part of the inheritance hierarchy, it links to versions in different languages and it allows you to enter values in order to search in catalogs (a possibility provided by a separate catalogserver).*



(2) the build-in multilinguality: attractive for a European research project with partners from six countries.

LexiCon contents were made available to the various parts of the eConstruct system by means of the *bcXML Taxonomy format* [24], after a conversion from LexiCon's own XML export format. A separate format was used because of a slightly different emphasis, but also because the non-LexiCon researchers wanted to research XML's possibilities more deeply. The bcXML Taxonomy format was also more explicitly focused on later use of the vocabulary by actual data. Being both XML formats, tool support was excellent, which helped in the implementation.

There are two versions of the bcXML Taxonomy format. The first version was quite elaborate and kept growing in size because of cross-breeding with the then-developing ifcXML format [25]. Another reason for the size was that it catered for both Taxonomy needs and data (catalogs, project information) needs. The second version was a much simpler model: complexity was halved. Halfway through the project, it replaced the initial elaborate one [26]. This decision helped the project along, as development of the prototype started to gain steam immediately. For an overview of the model, see figure 4.6 on page 70. This simple Taxonomy format was used a couple of times by other projects after the end of the eConstruct project [23] [27] ; it was used most directly in the EU projects e-cognos [28] [29] and funsiec [30]. The simpler model only focused on the Taxonomy level (the data format is analysed in the next section). A code example of the bcXML Taxonomy format can be seen in figure 4.5 on the facing page.

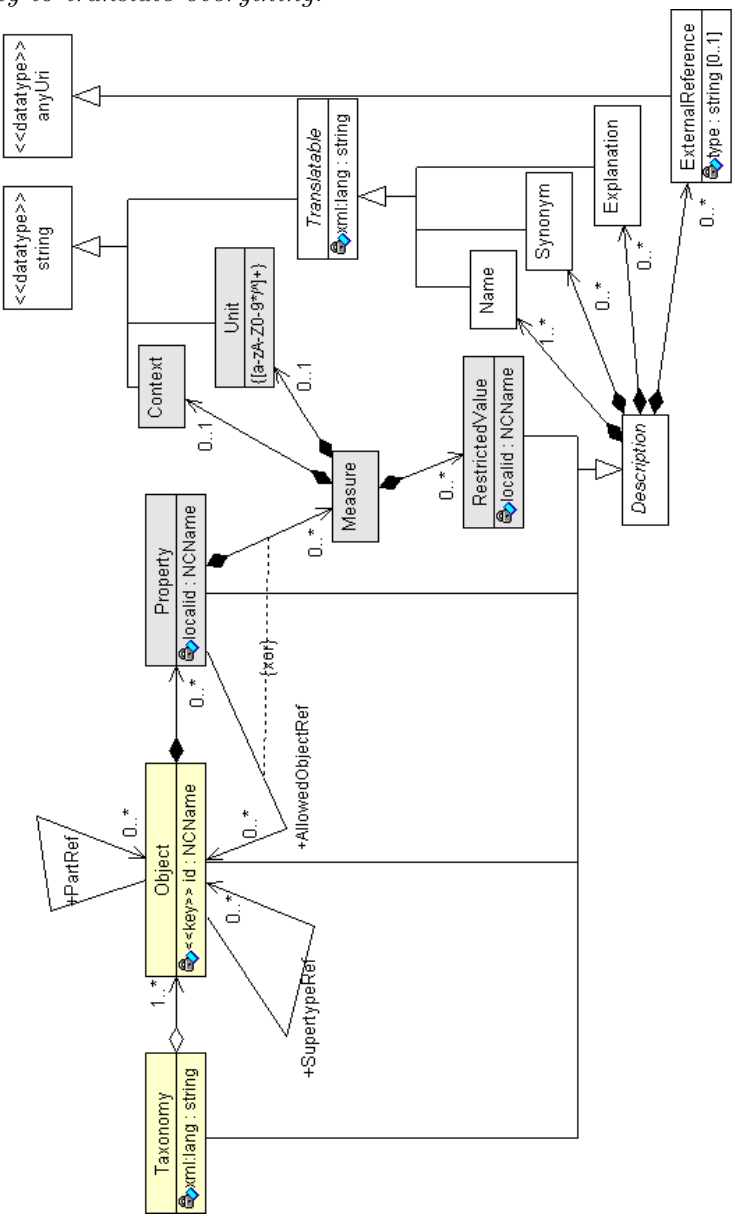
Adding Taxonomy data to the LexiCon did not proceed swiftly: in the end, effectively only data on doors was available for use in the prototype. The 'LexiCon Explorer' tool, see figure 3.6 on page 43, was a stand-alone application, so collaborative work was not possible, making this a bottleneck[6]. On a more fundamental level it turned out to be difficult to integrate the data needs of the practical proto-

[6]The current LexiCon Explorer version is based on a central mysql database and allows distributed development within one company, so the problem is being addressed.

Figure 4.5: *bcXML Taxonomy XML code example.*

```
<Taxonomy xmlns="http://www.bcXML.org/2003/bcXML"
          xml:lang="en">
  <Name xml:lang="en">bcbuildingdefinitions</Name>
  <Explanation xml:lang="en">
    Taxonomy exported from the LexiCon.
    Default language is English.
  </Explanation>
  <Object id="WoodenDoorleaf">
    <Name xml:lang="en">wooden door leaf</Name>
    <Property localid="doorleafHeight">
      <!-- ...... -->
    </Property>
    <Property localid="madeOf">
      <AllowedObjectRef>Wood</AllowedObjectRef>
    </Property>
    <PartRef>Letterbox</PartRef>
  </Object>
  <Object ....>
    ....
  </Object>
  <!-- etcetera -->
</Taxonomy>
```

Figure 4.6: *UML diagram of the most recent version of the bcXML Taxonomy format. It concentrates on objects with properties and on the possibility to translate everything.*



type with the careful creation of one big single-inheritance Taxonomy spanning BC as a whole.

The multi-linguality provided much added value to the final demonstration: showing the same content in multiple applications in multiple languages (English, Greek, Dutch, German, French, Norwegian) makes for an attractive demonstration. For the eConstruct system, the LexiCon's multi-linguality was a real asset.

The national differences, though, also provided difficulties. In different countries, the regulations and norms differ: fire resistance in the Netherlands is measured in minutes according to some norm (fire resistance = 30 min), fire resistance in Great Britain is measured according to a textual class (fire resistance = 'class 30 minutes'). This means that the property *fire resistance* can be measured in at least two ways, when it is not an entirely different property altogether. As a second example, the first floor in a British building is on ground level—in the Netherlands it is one floor up. And so on.

The idea of using a separate vocabulary proved advantageous, the multi-linguality was also a plus. The national differences, however, provided difficulties. Using XML for LexiCon's export and for bcXML's Taxonomy format worked well and allowed simple implementation.

4.4.3 Data format: bcXML

eConstruct's system used two formats: one for the Taxonomy, described above, and one for the actual data, like catalogs. The data format was designed to be a simple, readable format, called bcXML (the Taxonomy format is called bcXML Taxonomy format).

bcXML was a direct application of a result of the author's Master's thesis [31], where the idea of having a separate Taxonomy and a separate data format generated from that Taxonomy was pioneered [32]. The acceptance of a simple, readable data format had the effect of speeding up the eConstruct project's prototype development. For catalog data (the main use case for eConstruct), bcXML proved good enough [27].

A conversion generated the data format from the Taxonomy data.

Figure 4.7: *Example of bcXML data: a door with height* $= 2.10m$ *and width* $= 0.80m$*, followed by the relevant part of the bcXML Taxonomy where the bcXML data example is based on.*

```
<Door>
  <height unit="m" value="2.10" />
  <width unit="m" value="0.80" />
</Door>

...............

<Object id="Door">
  <Name xml:lang="en">door</Name>
  <Property localid="height">
    <Unit>m</Unit>
  </Property>
  <Property localid="width">
    <Unit>m</Unit>
  </Property>
</Object>
```

On the one hand the Taxonomy (using the bcXML Taxonomy format) has an object named 'door' with a property named 'fire-resistance' measured in unit 'meter'. From this information, a data format is generated that allows `<Door>` elements with, as subelements, the properties like `<fire-resistance>` with the units added as attributes like `<fire-resistance unit="m" value="....">`. A separate Taxonomy format and data format is elegant and allows for simple, focused formats. The data format can be seen in figure 4.7 on the facing page.

On a technical level, data formats from different Taxonomies were given their own namespace, so that different Taxonomies with objects with the same ID would not overlap on the data level. In practice, only one Ontology was used, though simple additional Taxonomies to assist visualisation were attempted on a prototype level. The functionality to combine data from various Taxonomies in one data file was, however, not fully developed and only used in a small visualisation prototype (see the 2D and 3D examples in figure 4.8 on the next page; a point that should receive attention in further efforts.

Within the eConstruct project, a variety of bcXML visualisations were explored [33], see figure 4.8 on the following page for examples. Most prototype applications used the Taxonomy directly to query for the name (and the description) in the correct language. Another development created simple visualising (eXtensible Stylesheet Language for Transformations (XSLT)) stylesheets from the Taxonomy (figure 4.9 on the next page), these stylesheets could then be used for textual or visual formatting of bcXML data—using standard XML tools.

Overall, the simpleness of the data format was a plus. The format is human-readable, which is good when it is one of the first implementations using such a new technology. In hindsight, the mechanism used resembles that of a simplified RDF notation, which delivers also readable element names. What did not work well was the mixing of various Taxonomies, something that was not really researched. For this a look at RDF could provide advantageous, as RDF has build-in formal means to mix Taxonomies (Ontologies).

With the bcXML Taxonomy and the bcXML data format now described, attention in the following subsection goes to the web services

Figure 4.8: *bcXML visualisation: multilingual text, SVG 2D and VRML 3D. The text and 2D images are from Joost Fleuren.*

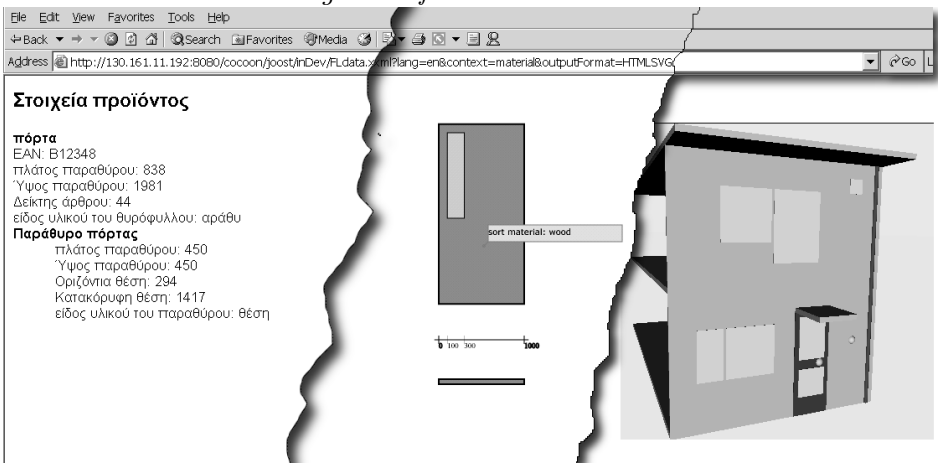
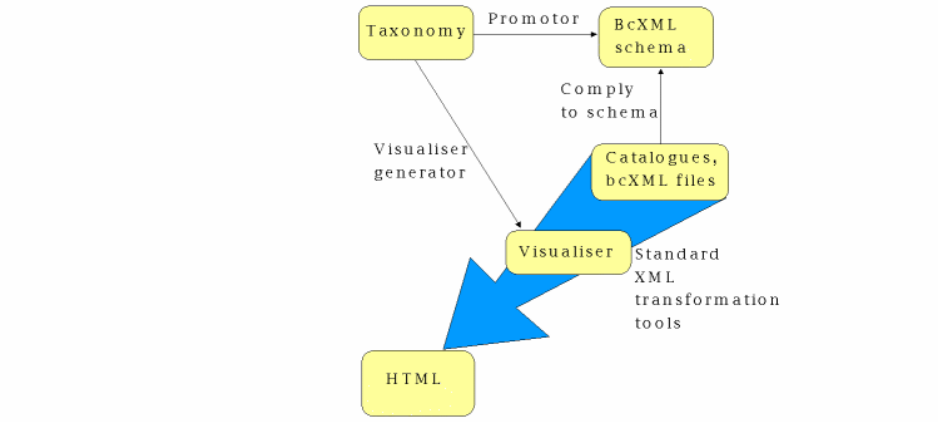


Figure 4.9: *bcXML stylesheet-based visualisation architecture. The Taxonomy holds all the translation and description information for objects and properties. The catalogs and other bcXML data files conform to the bcXML schema that has been generated from that Taxonomy format. So a visualiser XSLT can be generated from the same Taxonomy that knows how to handle bcXML data files and how to transform them into HTML.*



that used them.

4.4.4 bcXML web services

In eConstruct, two web services were created: the Taxonomy server and the catalog server. The advertised way to connect to the servers was by SOAP.

The catalog server allowed you to retrieve the list of catalogs and individual catalogs. Also, you could search for matching items in the catalogs, upon sending a bcXML-formatted query. Also, new catalogs could be uploaded.

Interesting is the original idea for the catalog servers—that wasn't implemented due to time constraints—to be able to tier the servers. 'Higher level' catalog servers could aggregate multiple 'lower level' catalogs or catalog servers, to provide a single entry point for catalog queries. This would enable individual bigger companies to aggregate their preferred suppliers in one company-internal catalog server, so that it would be easy to order from those preferred suppliers, probably including a pre-calculated lower price because of the higher volumes. Or aggregators could provide an additional service by doing some quality filtering on the catalogs they aggregate, removing untrustworthy suppliers.

The Taxonomy server allowed textual search (in the various languages) for matching items. Individual Taxonomy items with all their information could be retrieved. The same actions could be done via HTTP using the HTML interface, with the HTML being automatically generated from the Taxonomy server's XML results (using XSLT), so it was natural to also provide the possibility to download these XML results directly. This direct HTTP method proved much simpler than SOAP and was therefore used a lot for testing. In fact, the SOAP interface was a small wrapper around the HTTP interface.

In the final project demo, both web services were used from within various programs, running in various locations. Each of them was able to use the same data, which made for a convincing demo. Having data available in centrally accessible locations was a plus, it seemed to offer a definitive advantage over the otherwise-needed distribution of data

and subsequent problems of keeping it up to date. A question mark could be placed at the use of SOAP instead of HTTP+XML. SOAP did not offer anything extra, as exemplified by the Taxonomy server. SOAP did however necessitate the installation of extra software and it opened up some potential security holes.

4.4.5 Evaluation

Looking at the eConstruct experience, some conclusions seem warranted. Using a separate vocabulary (Taxonomy, Ontology) is advantageous. Multi-linguality is attractive, though national differences make the creation of an international vocabulary difficult. Building one big single-inheritance Taxonomy of all construction terms seems an impossible task when looking at eConstruct: the amount of terms produced after two years was disappointing. A context mechanism (for instance a country dependent definition) seems the most reasonable approach.

Using XML worked well and allowed implementation using generally available means. Simple, focused formats helped implementation greatly; eConstruct's success comes mostly from the rapid implementation that was possible (and that was done!) after the choice for simple formats was made.

For future efforts, a solution that can comfortably use multiple Taxonomies at the same time is advisable, as that was missing in eConstruct's system.

eConstruct's experience with web services was positive, the eConstruct-provided data was always up-to-date and accessible from multiple applications. In the eConstruct system, SOAP did not offer anything extra over HTTP+XML, so the latter alternative seems better suited as it is simpler.

4.5 Generic vocabulary: OWL

The goal of this chapter is to analyse NG Internet technologies for BC. This section analyses the Internet's proposal for a generic vocabulary: Ontology Web Language (OWL). After an introduction, the

three OWL variants are analysed. Using OWL by means of references instead of building Semantics into an application is analysed next, followed by OWL's reasoning possibilities. Web services, followed by a summarising discussion, close the section.

4.5.1 Introduction

XML allows a generic way to deal with syntax; RDF makes it possible to identify and relate data items and also provides a data syntax. This by itself is not enough, what is needed is to make the Semantics of data items explicit. An Ontology can be used to define classes and to provide a list of properties associated with those classes.

OWL is the language used to define Ontologies [34]. OWL is the successor to the DAML+OIL combination of languages. DARPA Agent Mark-up Language (DAML) and Ontology Inference Language (OIL) were the two best and most accepted efforts at an Ontology language to result from the Ontology research community [35]. DAML is of North American descent while OIL is the European development. The two languages had a lot in common, so a combination was agreed upon under the name of DAML+OIL. OWL is a slight clean-up of DAML+OIL and represents the best-practice from the Ontology research community and as such is an important input to BC Ontology efforts.

With RDF and OWL, the web has the potential to become an actual worldwide open database where both structure and content can be directly stored, exchanged and processed by software applications. RDF and OWL are much more powerful than XML. XML data formats have a *syntax* defined by Document Type Definition (DTD)s or XML Schema Definition (XSD) or Relax NG (RNG): these just provide structure. RDF and OWL provide the *Semantics*, going one necessary step further than plain XML.

RDF and OWL involve a lot of critical mass from multiple industry branches, especially from the (open) software tool support side, both development and Computer Aided Software Engineering (CASE) tools and infrastructure tools. Powerful free OWL editors are available, like protégé [36], which is a huge difference with for instance the STEP

ecosystem in BC, which lacks such generally available tools[7]. Generic availability helps with bringing new instruments to market, see the initial research questions in section 1.3 on page 7.

An Ontology's goal is to provide a common, referenceable set of concepts for use in communication. It is quite common to use multiple Ontologies, each providing concepts for a particular domain, together forming a rich vocabulary for communication. The term Ontology is discussed more fully in [37].

4.5.2 OWL variants

OWL comes in three variants, each with their own trade-off; they are introduced briefly, in order of rising complexity. All OWL variants build on RDF's schema language. RDF schema allows the specification of (sub)classes and (sub)properties and attachment of properties to one class. Also included are facilities for multilingual names, descriptions and simple values. The OWL variants all add to this RDF schema functionality.

OWL Lite OWL lite is a simple format for not-too-elaborate Classifications, Taxonomies and Ontologies. To RDF schema, it adds (in)equality so you can say that two classes in different Ontologies are (not) equal. Properties can be described more detailed, like saying that 'has part' is the inverse of 'is part of'. 'Contains' can be made transitive so that if A contains B and B contains C, A can be deduced to contain C.

A very useful addition to RDF schema is improved association of properties with classes: properties can now be applied to multiple classes, so that you can use a generic 'fire resistance' property on multiple classes without having to resort to one generic supertype for those classes. Lastly, limited cardinality[8] can be added; the only allowed values are 0 and 1.

[7] A core developer of a BC STEP-related standard needed somebody else's help to update the standard, as the company wouldn't afford the ExpressG editing tool.

[8] Cardinality means how much something can occur. A 'door' with a cardinality of two till four means that there are at least two doors and four doors maximum.

Important for good modeling practise are the versioning possibilities like 'backwards compatible with', 'incompatible with' and 'deprecated class/property'.

OWL DL OWL Description Logics (DL) provides all OWL possibilities (but with one restriction). It adds arbitrary cardinality. Properties can be pre-filled with data values or enumerations. Classes can be complex combinations and/or intersections of other classes instead of OWL lite's simple classes.

Description logics is the name of the formalism behind OWL's reasoning/deduction possibilities. OWL DL has one restriction—lifted by OWL Full described below—in order to guarantee deductability of all conclusions that can be calculated from an OWL file [38]. A deduction is for example that if a certain wall is next to a certain door and that door is on the first floor, that the wall is automatically also on the first floor (if the property of 'is next to' is defined in that way).

OWL Full OWL Full lifts the important restriction of OWL DL: instances of classes can be classes themselves.

A BC-related example is that of the Ontology class 'door'. A catalog item might be an instance of that class, so that you have several doors in that catalog. With OWL DL, you cannot say that an actual door in a project is an instance of such a door from the catalog, as the catalog door needs to be a class for that. In order for catalog items to be instantiatable, with OWL DL the catalog's door should be a subclass of the Ontology's door.

In practice this means that you are forced to stick to subclassing (instead of instantiating) for as long as possible, which does not seem to fit in too well with BC, as there are some distinct levels (Ontology, catalog, much-used items in projects, the actual items).

For BC, OWL Full's lifting of the subclass/instance restriction posed by OWL lite and OWL DL seems to decide the issue in favour

of OWL Full. The drawback is that the full reasoning capabilities of OWL cannot be used. In practice, this is only a problem when one completely deduced picture of all of BC (or even of one complete construction project) is desired, an exercise that probably fails much earlier because of the richness and variety of BC objects and viewpoints and the to-be-expected associated incompatibilities regarding Ontologies. Even with OWL Full, powerful restricted reasoning on an object-to-object basis is possible. For the foreseeable future, this restricted reasoning already is a big improvement.

The lack of enumerations in OWL Lite seems to rule it out altogether for BC Ontology usage.

4.5.3 References instead of build-in

Ontologies all by themselves serve no purpose: they must be used by actual data. By linking information items to Ontology items, the information is enriched and made usable to other applications that support the Ontology.

OWL uses the generic linking mechanism RDF to connect information items to Ontologies. For this, labeled relations can be used. some often-used relations are a build-in, like the 'is a type of' relation (instantiation). You can say that item 42 in an IFC file 'is a type of' `http://ifc.org/definitions/Door`.

A useful viewpoint is to treat Ontology Semantics as something that otherwise needs to be hard-wired into your application. Instead treat Ontology Semantics as something that you reference. Semantics are something you can, voluntary, reference from within your data; Semantics are not something you are forced to hard-wire into your data. By referencing Semantics you get a lot of benefits, but it should always be seen as a loose coupling.

It is normally a useful strategy to keep as much data as possible out of applications and to store it as meta-data in external files. This allows the data (including the assumptions behind the data) on the one hand and the application on the other hand to change at their own individual paces. The flexibility given by this loose coupling can be invaluable when data needs to be enhanced: changing the

application is much more involved than changing data files, as you need programmers in addition to domain experts.

A case can be made for being formal with regard to the data that is kept externally: [39] formalises this loose coupling as *Ontology-driven information system development*, keeping the structural data out of the application in the form of a formal Ontology.

4.5.4 OWL reasoning

Besides being a standardised web format for Ontologies, OWL also allows a certain amount of reasoning. The Semantics of the OWL model elements are well-defined, which is a necessary prerequisite for reasoning. When a certain property (say 'door height') is defined as being only allowed on a certain class (say 'door leaf'), any object that has that property can be deducted as being part of that class. When needed, this allows you to define class A as being the same as class B, but excluding those objects that have property C.

In practice, you can for instance exclude objects from being a member of the class 'load bearing walls' if they are not connected to another supporting load bearing structural element, like a foundation. This way you can detect possible failures on the condition, of course, that the Ontologies and supporting applications are detailed enough.

[40] advocates an integrated approach by saying that ICT (Internet) should be an integral part of company and project knowledge management systems .

4.5.5 Semantic web services

The *semantic web* is a term often used for the collection of technologies analysed here, taking the normal web one step further. On the web, you can access lots of documents with a URL and link from document to another document anywhere; on the semantic web, you can point from one object to another object anywhere in some file: the semantic web extends the URL mechanism to a GUID mechanism [41]. Build into the semantic web is the idea of Ontologies, providing also a special format for Ontologies: OWL.

A combination of web services and semantic web makes the web services' advantages bigger, as it makes the data that goes over the wire even more web-centric. For BC, this merges two technologies that both enable a better and more dynamic information exchange: generalised, reasonably simple data *structuring* plus simple and ubiquitous data *exchange*. The combination makes it even more attractive for BC.

4.5.6 Discussion

Comparing the functionality of OWL with other languages like 12006-3 or eConstruct's bcXML Taxonomy format shows that OWL is really a generic, small, but very flexible core model. The W3C people wanted to be able to formally define OWL in a very precise way and therefore left out meta-concepts that were hard to define as a language construct. The industry-independent approach ensures future possible interoperability and re-use in and between different industry sectors.

An interesting feature of OWL is that missing functionality of basic OWL can be added by using OWL itself to describe the extensions. A recent effort of TNO in the Netherlands resulted in a number of extension Ontologies containing, for example, the 12006-3 based archetypes like 'activities' and 'actors'. Other examples of what they added in a test setting are units and complex properties, IFC-like placement and composition.

A strong point of OWL is that its Semantics are *formally* defined, using Description Logics (DL). This puts it ahead of BC initiatives like IFC and 12006-3, which do not have the formal logic behind them. 12006-2, more or less a formalisation of the SfB system, has formal logic that defines the core concepts, as, in their opinion, in order to achieve clarity in communication and technical development, a common conceptual framework is needed [42]. This can perhaps be extended from the Ontology's content to the Ontology technology itself.

Attesting to RDF's and OWL's thought-out model is that some of the concepts explicitly modelled in 12006-3 and bcXML are treated

in a much more comprehensive way. URLs are used for unique identification, which provides both an elegant Globally Unique ID (GUID) and a mechanism for extension, because other Ontologies can expand upon the concepts defined by those URLs without a need for those expansions to be included in the original file. Also multilingual labels and descriptions are build into RDF using a standardised XML method.

4.6 Improving market acceptability with open source

The key question then seems to be: how to develop a standard that is semantically rich, can be useful and even commercially attractive within a short period of time (less than one year) and allows the owners of the information and knowledge to keep control and ownership. This is important for bringing new instruments to the market (section 1.3 on page 7).

The suggestion put forward in this research is fourfold. The cost of use should be minimal; there should be as much application support as possible; development should be lightweight; the development should be supply/demand-oriented. These four points are elaborated below.

Low cost of use All other things being equal, a lower price increases demand. The suggestion for this concept solution is to make the usage free of charge. Free of charge does not equate to low quality, as is exemplified by wikipedia, the free, on-line encyclopedia [43].

Much application support Application support will depend on individual economic decisions on the part of the software vendors. What can be done, though, is to do away with all barriers for the software vendors. So: no mandatory membership of an organisation, no fees payable before you're allowed to access the Ontology, no risk by tying yourself to one platform vendor. This all can be achieved by making the Ontology *open source*. This makes the Ontology free in the sense of 'gratis' and free in the

sense of freedom to use, freedom to change, freedom to redistribute. One restriction that might be a good idea is to demand that the redistributed or changed Ontology contents be freely available under the same conditions, this as a safeguard against an embrace-and-extend attack by another market party.

With the same-condition restriction, the GNU Free Documentation License (FDL) is recommended as the most common [44], otherwise the popular Creative Commons (CC) license[9] [45].

When a purely commercial Ontology approach is chosen, adoption will depend almost solely on political decisions in BC. Such a central position that covers most of BC's information exchange can only be granted by a large majority of parties, all in agreement on granting that central position to one commercial entity.

Lightweight development Heavyweight committee-based development is slow. Some of this work is rightfully hard and should take a long time. The suggestion for the solution concept, though, is to develop the Ontology as multiple small pieces that can be made by just a few interested individuals, without having to resort to formal organisations and meetings. The NG Internet possibilities allow you to tie the various independently developed parts together in a web-based network.

Supply/demand oriented To achieve immediate relevance and interest in the BC industry, a new development has to attach itself directly to the money-making process. If an Ontology can be build on basis of a supply/demand distinction, it can start lubricating the market mechanism in the BC industry. At the least it can provide a few innovating new ways to connect supply and demand.

Concluding, the proposal is for an open source web-based object library as a very interesting new effort at creating a useful basis for web-based communication in BC.

[9]Note that this thesis is itself available under a CC license, see details on page 233.

4.7 Suitability of NG Internet for Building-Construction

The NG Internet (semantic web, web 2.0) can provide an opportunity to help the BC industry become a knowledge-rich (high-tech) industry with well controlled processes and guaranteed output quality.

In traditional BC, little value is added by each partner in the chains. There is, mostly, only a small difference between actual costs to build the product and value provided to the customer. Assuming that the price sits in between the costs and the value, only little room exists for profit for the Virtual Enterprise that performs the project. Also, relative little extra value is provided for the money spend by the customer.

State-of-the-art web technology can be an instrument to help realise new, dynamic BC processes that give Clients more value for money and construction companies more money for the value provided.

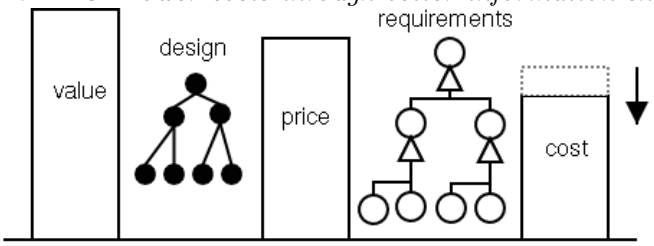
There are three basic avenues that can be pursued. Cost incurred can be lowered, value provided can be increased and, third, interaction between the cost and the value can be streamlined.

The NG Internet dramatically increases possibilities for information and knowledge sharing. Knowledge sharing is at the very least as important as information sharing for the desired knowledge economy. Information is *what* and *how*, knowledge is *how* and *why*.

With NG Internet possibilities arise firstly for back-and-forth information exchange, allowing the process to become more dynamic. There is improved upstream and downstream information and knowledge exchange, not just a transfer of a fixed document at a fixed point in the process's time after which no exchange is possible or feasible.

Secondly, increased exchange and availability of information also opens up more opportunities for enriching the information with valuable knowledge. Knowledge enrichment is now done by humans in a manual process. When more communication takes place in computer-understandable ways, in combination with a much more fine-grained communication, every one of those communications is an opportunity to (automatically) attach knowledge. For example, continuous check-

Figure 4.10: *Lower costs through better information sharing.*



ing of the height of the building-being-designed with local regulations; checking suppliers with the in-house observed-quality database when comparing offers; doing an maximum escape route length calculation [46] after each office building design change. This way, new knowledge possibilities are integrated into the BC process that simply are not currently feasible on that level.

These three core solution concepts, *improved information sharing*, *dynamic interaction* and *knowledge integration* are discussed in the next sections. They are coupled with the three improvement avenues, *lowering costs*, *increasing value* and *streamlining interaction*.

4.7.1 Improved information sharing

An example. A Client revises his opinion and wants to have an extra 10 square meter of paved surface in his garden. The Contractor that contracted the refurbishment of the house changes the project data. The gardener, when buying pavement tiles next week, has this small project change automatically reflected in his item quantities, printed out from the inventory section of the small-company-management web site that he's using to run his company.

Second example. The last-minute political wish to upgrade the plans for the new locks for sea-going vessels to allow a maximum ship length of 180 meters is entered into the project's data. The Waterworks Department's cost estimator application is re-run. The cost estimator downloads the revised project data and calculates the new estimate.

Examples like the two above are often far from current reality.

The gardener in the first example could just as easily have heard about the Client's changed wishes when arriving to put the pavement in place: with a truck with not enough paving tiles to add the extra 10 square meters. The options are to come back next day with more tiles or to go to get them right now (both costing additional money) or for the Client to do without the extra paved surface (meaning disappointment and less value). All because a simple telephone or fax from the Contractor slipped through the cracks.

Internet is available practically everywhere. On the physical level there is therefore no problem to access information in another location. The NG Internet provides means to make human-readable information available as computer-readable data. In many scenarios, computers could do menial tasks automatically, if only they are given the data to work on. Some of these tasks will free humans to do real value adding work, but other tasks will be tasks that are not even done today, like checking and checking and checking again for errors stored in the company's database of often-recurring errors.

When looking at the level of *Client value* and *construction industry cost*, a value adding component is present in NG Internet fueled improved information sharing. But the low-hanging fruit is probably in reducing the costs by eliminating construction or management failures (see figure 4.10 on the preceding page).

When looking at the level of *Contractor value* and *supplier costs*, some cost effectiveness can be achieved on the supplier side by, for instance, automatic order processing instead of dealing with faxed orders. The Contractor, however, is enabled to handle a much larger palette of products when compared to searching paper or web catalogs by hand. This larger palette means he has a better chance at finding the products best suited to the task, getting more value.

4.7.2 Knowledge integration

With computer-understandable information available and accessible, new market opportunities occur. A very real value adding opportunity is to add value by providing knowledge. Providing knowledge in addition to or instead of 'stacking bricks'. Two examples.

A company specialised in heating installations has acquired valuable experience with installing installations in old buildings. They work co-operatively with a main Contractor, and have access to the project's building model data (the object-based 3D CAD model). The company is able to design a heating installation that is fitted to the old building: it not only heats the building, but actually actively helps combat such an old building's inherent draft and humidity problems in such a way that it saves serious money on measures otherwise necessary. Early and concurrent access to the building information allows the company to transform their knowledge into added value for the Contractor.

For their internal needs, the defence department created a database of the kinds of repairs to their buildings that recur often. They also identified problem indicators, such as buildings built around 1960 in humid areas, or even simple things like flat roofs that need to be re-done every ten years. They built an application that could check for these problem indicators in the sub-department's building databases by downloading the information from the web. If a pro-active action on a building was needed, a partly pre-filled project website was generated for the sub-department. This pro-active, knowledge-based approach saved the defence department a lot of money with regard to the annual total repair costs.

For examples like this to work, a generic way to recognise objects in other data is needed. In the above example, the application had to look for 'flat roofs' amongst other things. If the building's data would contain a 'tarred flat roof', the scenario could only work if there was an easy way to say that a 'tarred flat roof' is also a 'flat roof'. The NG Internet has the build-in possibility to recognise this.

Also illustrated here is the need to identify the separate objects that make up one project. It is not enough to say 'this is building C's drawing', it must be possible to ask for the information for 'all the doors on the third floor'.

The possibilities shown here tell us that the NG Internet allows the combating (sic) of costs with knowledge. It also opens up a wider market for knowledge. This will make BC as a whole a much more knowledge-intensive industry. See figure 4.11 on the facing page.

Figure 4.11: *Knowledge integration: more value through better, smarter solutions; lower costs through smarter and more efficient working.*

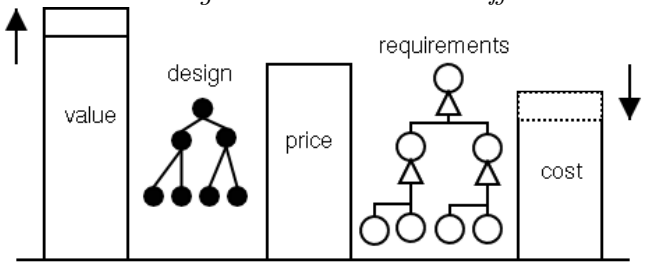
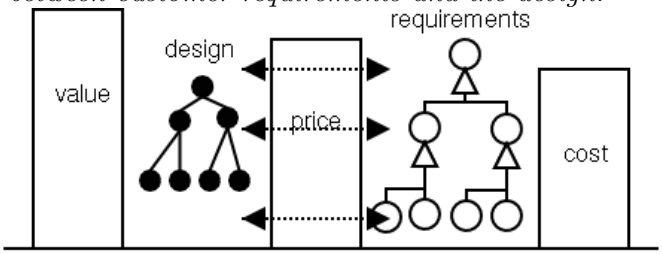


Figure 4.12: *The Next Generation Internet offers a much more dynamic interaction between customer requirements and the design.*



4.7.3 Dynamic interaction

With much improved information accessibility and sharing, the possibility arises to streamline the interaction between value and cost. In the interaction between an Client and a Contractor, the value received by the Client can be higher, as increased interaction means that what is build matches better with his wishes.

The interaction offers the Contractor the opportunity to work more effectively, as there is a bigger chance that his expertise and experience is used to more effect.

When looking at more dynamic interaction between Contractors and suppliers, a back-and-forth (see figure 4.12) between design and available products can lead to changes that provide more value for a small increase in costs, or even lowered costs. Likewise, in these interactions attractive offerings might well weigh up against a small decrease in value. This, of course, in this example needs interaction

between the Client and the consortium.

4.8 Conclusions

The Internet has enabled an impressive increase in communication possibilities.

Increasing the value adding performance of BC necessitates an increase in communication and information and knowledge exchange and processing. The most powerful communication medium available is the Internet, so BC knowledge and information should use the Internet for maximum impact on BC's processes.

Overlooking the BC development efforts regarding the Internet, while noticing that developments in this area go extremely fast and what is true today, might be less true tomorrow, the following may be concluded:

Information availability The Internet is near-universally available and accepted. So much so, that for BC information to be really sharable, it has to be available over the Internet.

Usable data The NG Internet provides ways to make data understandable and usable for both humans and computers. Currently, many mechanical and boring tasks must be performed by the human participants in the process—purely because the data isn't structured enough for computers to jump in. Also, many innovative, time-saving and common-errors-preventing technologies are handicapped just by lack of usable data to operate on. NG Internet technologies are widely and cheaply available, which makes it more interesting for BC than specialistic solutions. On a more specific level, this translates to the following three *technical* conclusions:

First, BC information must be made accessible and identifiable using standard Internet means (HTTP, XML, RDF). This increases the accessibility and generic addressability of BC information. Information can become really part of the process.

Second, to link various information items together, RDF provides a generic reference and linking mechanism that merits detailed research on its applicability in a BC setting.

Third, an emerging theme that can be observed in BC ICT research is retained explicitly: the separate storage of Semantics in an Ontology. OWL provides a generic mechanism to create Ontologies. It seems to offer a more generic basis for Ontologies than 12006-3, the current most-advanced technology in BC, though it has to re-use some concepts also in 12006-3.

Involving applications Web services, providing either information stored in applications or results from calculations or services, are a good way to make even more information and knowledge available and accessible.

National BC Semantics BC Semantics has not received the attention it required. One reason has been the difference between objects in BC in different countries. The European eConstruct project clearly showed that objects like 'inner doors' are different in each country. It seems therefore inevitable that detailed modelling efforts in an international setting failed and that successful developments should be context dependent (for instance, they should start on a national scale).

The proposal is to use the NG Internet's W3C standards as much as possible. The existing developments in BC are retained, when useful. No existing effort is needlessly re-done. Existing developments should, however, be brought into line with the NG Internet.

Bibliography

[1] Chris Locke, Rick Levine, Doc Searls, and David Weinberger. *The cluetrain manifesto*, chapter 5. Perseus publishing, January 2001. Chapter available on-line at `http://www.cluetrain.org/book/hyperorg.html`.

[2] Paul Prescod. Hypertext transfer protocol - http/1.1, rfc2616, 1999. Available on-line at http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html.

[3] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. Available on-line at http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[4] Steve Vinoski. Putting the "web" into web services - web services interaction models, part 2. *IEEE Internet Computing*, pages 90–92, July/August 2002. Available on-line at http://www.iona.com/hyplan/vinoski/pdfs/IEEE-Web_Services_Interaction_Models_Part_2.pdf.

[5] Stefan Marr. Restful web services, 2005. Available on-line at http://www.stefan-marr.de/artikel/restful-web-services/paper.html.

[6] J.O. Zijlstra. *Bestekken in de Grond-, Water- en Wegenbouw*. C.R.O.W., 2e druk, 2e oplage edition, 1995.

[7] W3C. Hypertext markup language (HTML) home page. Available on-line at http://www.w3.org/MarkUp/.

[8] Tim Bray. Technology prediction success matrix–12: 80/20 point, January 2004. Available on-line at http://www.tbray.org/ongoing/When/200x/2004/01/14/TPSM-8020.

[9] W3C. Resource description framework (RDF), 2004. Available on-line at http://www.w3.org/RDF/.

[10] Tim Bray, Jean Paoli, C Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (XML) 1.0 (third edition), February 2004. Available on-line at http://www.w3.org/TR/2004/REC-xml-20040204/.

[11] W3C. XML schema, 2000. Available on-line at http://www.w3.org/XML/Schema.

[12] James Clark. Relax NG homepage, September 2003. Available at http://www.relaxng.org/.

[13] Robin Cover. SGML overview, July 2002. Available at http://xml.coverpages.org/sgml.html.

[14] Dan Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema, February 2004. Available at http://www.w3.org/TR/rdf-schema/.

[15] Mark Pilgrim. Should atom use RDF?, August 2003. Available at http://www.xml.com/lpt/a/2003/08/20/dive.html.

[16] Bob DuCharme. Converting XML to RDF, September 2004. Available at http://www.xml.com/pub/a/2004/09/01/tr.html.

[17] David Harrison, Michael Donn, and Henry Skates. Applying web services within the AEC industry: enabling semantic searching and information exchange through the digital linking of the knowledge base. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at http://itc.scix.net/data/works/robots/w78-2003-145.htm.

[18] Paul Prescod. Second generation web services, February 2002. Available on-line at http://webservices.xml.com/pub/a/ws/2002/02/06/rest.html.

[19] Sean McGrath. Would the real source of metadata please stand up?, 2004. Available on-line at http://www.itworld.com/nl/ebiz_ent/05042004/.

[20] Tim O'Reilly. Inventing the future, September 2002. Available on-line at http://www.oreillynet.com/pub/a/network/2002/04/09/future.html.

[21] Website eConstruct project, 2000-2002. Available on-line at http://www.econstruct.org/.

[22] Jeff Stephens. Jeff Stephen's eConstruct demo, including many screenshots of the actual software demo., Januari 2002. Available at `http://vanrees.org/research/papers/presentations/borrowed/demo_jeff.ppt`.

[23] G. Antoniadis and K. Menzel. Virtual pools of resources eliminate idle or missing equipment in AEC companies. In Attila Dikbaş and Raimar Scherer, editors, *eWork and eBusiness in architecture, engineering and construction*, pages 409–414. ECPPM, Balkema, September 2004.

[24] bcXML taxonomy format, September 2002. Available at `http://bcxml.net/formats/taxonomy/`.

[25] Frits Tolman, Reinout van Rees, and Michel Böhms. Building and construction extensible mark-up language (bcXML). In Gustav Coetzee and Frances Boshoff, editors, *IT in Construction in Africa*, CIB-W78, pages 38.1–38.10. CIB, CSIR, May 2001. Available at `http://itc.scix.net/data/works/att/w78-2001-67.content.pdf`.

[26] Reinout van Rees, Frits Tolman, and Rèza Beheshti. How bcxml handles construction semantics. In *Conference proceedings - Distributed knowledge in building*. CIB-W78, 2002. Available on-line at `http://vanrees.org/research/papers/Cib78ConferencePaper2002`.

[27] Shailesh Jain and Godfried Augenbroe. A performance-based approach for product selection from e-catalogues. In Žiga Turk and Raimar Scherer, editors, *Conference proceedings - E-work and e-business in AEC*. ECPPM, Balkema, 2002.

[28] Marc Bourdeau, François Giraud-Carrier, Yacine Rezgui, and Alain Zarli. Knowledge management in the construction industry: the e-COGNOS approach. In Brian Stanford-Smith and Enrica Chiozza, editors, *E-work and E-commerce*. IOS Press, 2001.

[29] Reinout van Rees. E-cognos project's final meeting in Paris, October 2003. Final project presentation transcript available on-line at `http://vanrees.org/weblog/1065185980`.

[30] Celson Lima, Bruno Fiès, C Ferreira da Silva, and S Barresi. Setting up the open semantic infrastructure for the construction sector in europe - the FUNSIEC project. In Attila Dikbaş and Raimar Scherer, editors, *eWork and eBusiness in architecture, engineering and construction*. ECPPM, Balkema, September 2004.

[31] Reinout van Rees. ceXML - an XML vocabulary for building and civil engineering. Master's thesis, Delft University of Technology, August 2000. Available on-line at `http://vanrees.org/research/econstruct/afstudeerverslag`.

[32] Reinout van Rees. ceXML - an XML vocabulary for building and civil engineering (chapter 6). Master's thesis, Delft University of Technology, August 2000. Chapter 6 available on-line at `http://vanrees.org/research/econstruct/afstudeerverslag/messagevocabulary.html`.

[33] Frits Tolman, Michel Böhms, Celson Lima, Reinout van Rees, Joost Fleuren, and Jeff Stephens. eConstruct: expectations, solutions and results. *ITcon*, Special issue on European projects, 2002. Available on-line at `http://itcon.org/2001/13`.

[34] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview, February 2004. Available at `http://www.w3.org/TR/owl-features/`.

[35] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL reference description, December 2001. Available at `http://www.w3.org/TR/daml+oil-reference`.

[36] Protégé ontology editor website. Available on-line at `http://protege.stanford.edu/`.

[37] Reinout van Rees. Clarity in the usage of the terms ontology, taxonomy and classification. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at `http://vanrees.org/research/papers/Cib78ConferencePaper2003`.

[38] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003. Available on-line at `http://www.cs.man.ac.uk/~horrocks/Publications/download/2003/HoPH03a.pdf`.

[39] Nicola Guarino. Formal ontology and information systems. In Nicola Guarino, editor, *Formal ontology in information systems*. IOS Press, 1998. Available on-line at `http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/FOIS98.pdf`.

[40] Per Christiansson. Next generation knowledge management systems for the construction industry. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at `https://www.cs.auckland.ac.nz/w78/papers/W78-69.pdf`.

[41] Reinout van Rees and Frits Tolman. Semantic web technologies applied to building specifications. In *Conference proceedings - CIB world building congres Toronto*. CIB, 2004. Available on-line at `http://vanrees.org/research/papers/`.

[42] Anders Ekholm. A conceptual framework for classification of construction works. *Electronic journal of information technology in construction (itcon)*, 1(2), March 1996. Available on-line at `http://www.itcon.org/1996/2/`.

[43] English wikipedia website. Available on-line at `http://en.wikipedia.org`.

[44] Free Software Foundation. GNU free documentation license, November 2002. Available on-line at `http://www.gnu.org/copyleft/fdl.html`.

[45] Creative Commons. Creative Commons website, 2004. Available on-line at http://creativecommons.org/.

[46] Michael Spearpoint. Properties for fire engineering design in New Zealand and the IFC building product model. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at http://itc.scix.net/data/works/robots/w78-2003-333.htm.

Captain, I found some numbers that don't add up.
They don't do this in a very scary way.

Commander Kevyn Andreyasn

5

Revisiting the research questions

This chapter revisits the initial research questions to re-focus for the next chapters. The goal is to take the analysis of the previous chapters and prepare it for the solution concepts chapter.

The *introduction* places the chapter in context and follows with a discussion on the *NG Internet conclusions*. The next section deals with the *market adoption chances* of new instruments, followed by the analysis' preliminary *conclusions on traditional Specifications*. The *reformulated research questions* close off the chapter.

5.1 Introduction

In the introduction (chapter 1) the following three initial research questions were raised. (1) How can new instruments that apply Next Generation Internet (NG Internet) technology help to improve information and knowledge sharing between all stakeholders, especially in the various chains? (2) How can the new instruments be brought to market, *i.e.* which type of standardisation is required (if any) and how should the software vendor community react? (3) What is the role of

traditional Specifications in the future?

In the problem description (chapter 2), it was concluded that increasing the industry's information and knowledge sharing capacity can help change the current static nature of the process into a more dynamic structure that is much more tolerant for change and does not require early agreement about all the details that really cannot be overseen beforehand. In a dynamic system, shortening and strengthening the feedback loop is instrumental in improving the efficiency of the whole system.

This chapter revisits the original research questions, based upon the conclusions from the previous chapters. Every original research question is discussed separately in the following sections.

5.2 New NG Internet instruments: help for information sharing?

As mentioned in the problem description (chapter 2), the need for solving the social and juridical sides of the problem stands undiminished. To ensure a dynamic BC, however, ICT implementation is a sub-problem of like magnitude.

The analysis of current information and knowledge sharing in the BC industry (chapter 3) concluded that the next generation instruments should promote transparent ways of storing information; provide for re-use of information and knowledge; support co-operation between applications; and have improved market acceptance.

The analysis of new applicable NG Internet technologies (chapter 4) resulted in the following conclusions, partly answering above problems:

Information availability The Internet is near-universally available and accepted. So much so, that for BC information to be really sharable, it has to be available over the Internet.

Usable data The NG Internet provides ways to make data understandable and usable for both humans and computers. Currently, many mechanical and boring tasks must be performed

by human participants in the process—purely because the data isn't structured enough for computers to jump in. Also, many innovative, time-saving and common-errors-preventing technologies are handicapped just by lack of usable data to operate on. NG Internet technologies are widely and cheaply available, which makes it more interesting for BC than specialistic solutions.

Involving applications Web services, providing either information stored in applications or results from calculations or services, are a good way to make even more information and knowledge available and accessible.

National BC Semantics BC Semantics has not received the attention it required. One reason has been the difference between objects in BC in different countries. The European eConstruct project showed that objects like 'inner doors' are different in each country. It seems therefore inevitable that detailed modelling efforts in an international setting failed and that successful developments start on a national scale.

Generic accessibility of information is provided by the Internet. First, Internet as such is available practically everywhere. Some building sites have started to use wireless Internet on-site (WiFi) and there are multiple projects using mobile Internet access. With the Internet, you can access all needed data from everywhere. Second, NG Internet makes the exchange of computer-readable data, instead of only human-readable data, possible. Third, web services allow applications to participate fully in the Internet-based flow of information and knowledge.

If BC intends to improve information and knowledge sharing, the Internet as such seems mandatory.

Use of Ontologies—capturing Semantics separate from the rest of the system—is assumed, as previous efforts at directly integrating the wide variety of BC objects in a communication standard have not met with success. For these Ontologies to be successful—which means actually being used by data—they profit greatly from generic

Ontology support in data formats. The NG Internet provides this Ontology support, but its suitability needs to be demonstrated.

For a new, dynamic BC process, improved information linking seems needed. Activities like weighing alternatives, receiving feedback, discovering alternatives, connecting chosen solutions with the original requirements: in a much more dynamic BC, these activities have to be done much more often. On an information level, it are all basically links between both knowledge and information. Without the links, the activities are not possible. The NG Internet has a built-in generic linking mechanism that presents itself as the suitable candidate. Here, also, research is needed as to the suitability for BC.

Analysis of new applicable NG Internet technologies (chapter 4) adds the conclusion that BC Semantics has not received the attention required. One reason has been the difference between objects in BC in different countries. The European eConstruct project showed that objects like 'inner doors' are different in each country. It seems therefore inevitable that detailed modelling efforts in an international setting failed.

If the generic linking mechanism works well, this opens opportunities to create Ontologies in a more distributed—and perhaps more dynamic—manner. Further research is needed, possibly combined with research for the generic Ontology support.

5.3 How can new instruments be brought to market?

The analysis of current information and knowledge sharing in the BC industry (chapter 3) concluded that next generation instruments should promote, amongst others, an improved market acceptance. Current PDT research lacks adoption, which is needed if BC *as a whole* should be able to use the new instrumentation.

Of critical importance to market adoption is support of new instruments by the software vendors, as they will be the most direct link to BC companies.

If a new, more dynamic, way of working is to fully take root in

BC, the social and juridical and instrumentational all have to be in place. The stronger and more attractive each part is, the higher the success chances for the whole. A possible scenario is that the social and juridical aspects are in place and that the instrumentation is more or less forced upon BC. This does not seem like an attractive or safe scenario.

Instead of juridically forcing a new instrumentation, a more desirable scenario occurs when the instrumentation is—economically—attractive in its own right. This attractiveness should be aimed for, to improve the chances for new, dynamic BC. Additionally, economic attractiveness probably is a better incentive for software vendor support.

When designing the new instruments, the desired market acceptance must be kept in mind. The important question is what advantages are available for software vendors: they are the ones that have to create the instrumentation for a dynamic BC.

5.4 What is the future role of traditional specifications?

In the description of the problem (chapter 2), a conclusion was that increasing the industry's information and knowledge sharing capacity seems to help change the current static nature of the process into a more dynamic structure that is much more tolerant of change and does not require early agreement about all details that cannot be overseen beforehand. In a dynamic system, shortening and strengthening the feedback loop is instrumental in improving the efficiency of the whole system.

Chapter 3 concluded that Classification and Specification systems are in wide-spread use and seem to fulfill an important information-ordering role in the BC industry. Another conclusion was that one single Classification system is not enough (section 3.2.1 on page 25), a more ICT-based approach might be able to present multiple Classification views to the same information, using Classification as a user-friendly view on the data.

Because of their importance and their central role in the information process, traditionally between the design phase and the construction phase, and having a large number of links to other information sources, Specifications are well suited to demonstrate the new instrumentation.

A more dynamic BC means for example more interaction between the design phase and the construction phase; phases that lose their strict boundaries. This might both threaten and strengthen the Specification's position and thus seems a worthwhile research subject.

At the same time, research on the Specification's role might shed some light on the future of BC as a whole when BC's processes are renewed. How will the future of the BC industry look like if the market picks up the ideas forwarded in this thesis and what will be the consequences for all parties involved?

5.5 Reformulated research questions

After the three analysis chapters it is now possible to reformulate the initial research questions into a more elaborate form:

1. How can state-of-the-art web technology possibly help to realise new, dynamic BC processes that adequately serve the Client's needs (provide value for money) and, at the same time, serve the needs of the construction companies (provide money for value)?

2. How to use the NG Internet to provide both parties (and their application software) with the precise and Semantically rich online BC object definitions?

3. How can software vendors take advantage of such a development?

4. What is the possible role of traditional Specifications in newly innovated BC construction processes?

In the end, the desired end result is a marketplace-developed *de facto* standard for information and a marketplace-developed, gener-

ally available and generally affordable instrumentation for dynamic Building-Construction.

Semantic Web, proper noun: An attempt to apply the Dewey Decimal system to an orgy.

Devil's dictionary

6

Solution concepts

This chapter describes the solution concepts. The problems facing Building-Construction (BC) and their clients are complex and involve many aspects (cultural, social, legal, contractual, financial) that are out of scope for a single PhD research project. What can be done however is to try to develop a solution concept with a scope just large enough to involve all the stakeholders, and flexible enough to co-operate with most solutions for the other aspects.

The *introduction* places this chapter in context and revisits the research questions that must be answered here. The next section deals with *BC Semantics*, followed by *web services*: a new generation of application software that is better equipped to support the communication and co-operation needs of BC. *Specifications*—as an important illustration of the concept—are detailed next. The next section describes the *business model* proposed as part of the solution concept, as it is important that this research finds its way into practice. The ways in which the concept allows *innovation* in BC is next, followed by the *conclusions*.

6.1 Introduction

The main problem targeted in this PhD research is the question how to help the BC industry to improve its value adding performance. In chapter 5, 'Revisiting the research questions', the question has been restated as:

1. How can state-of-the-art web technology possibly help to realise new, dynamic BC processes that adequately serve the Client's needs (provide value for money) and, at the same time, serve the needs of the construction companies (provide money for value)?

2. How to use the Next Generation Internet (NG Internet) to provide both parties (and their application software) with the precise and semantically rich, on-line BC-object definitions?

3. How can software vendors take advantage of such a development?

4. What is the possible role of the traditional Specification in the newly innovated BC processes?

For BC to become a more value adding industry, knowledge integration seems a priority. At the moment, knowledge is either present in experts or codified in regulations, codes of practice, brochures and booklets. Also computer applications contain a lot of knowledge, the problem is that those tools only serve the experts. Knowledge that is directly available to non-experts is hard to find.

A second part is the strict Internet-availability and computer-readability of the data. Being able to access BC information in a generic way is a huge difference with current practice. Currently, most information is only available textually (or as 2D lines) and only available through application-specific methods.

6.2 Building-Construction Semantics

BC uses many technical terms, estimates go up to 300k [1]. Most BC terms describe components, materials, connections, equipment

and such. Other terms identify verbs like 'painting', 'driving' and again others describe properties, units and such. Formally defining so many terms is not a simple task. Still that is what is needed if computers will ever be able to help to enhance the industry's capacity to communicate and co-operate.

6.2.1 Desired properties of the Semantics

As a first point, note should be taken of the comments in Chapter 3 on standards and standardisation processes. The analysis showed a low success rate for committee-developed standards. This does not, however, exclude the eventual standardisation of a *de facto* market-developed set of Semantics by a standardisation institute, should the need arise. The analysis shows that development of a *de jure* standard in a standardisation process does neither guarantee marketplace adoption nor results as such. For this research, a desired end result is a marketplace-developed *de facto* standard [2].

As analysed in chapter 3 and chapter 4, BC Semantics is a basic requirement for improved communication. For this basic requirement, it is necessary that adoption is not hindered by costs. The lower the cost, the higher, normally, adoption. With low costs, the cost of actually collecting the fees might easily be higher than the amount collected. Likewise, a fee collection process means a controlling organisation, which leans towards the committee-based standard development that was discussed above. The BC semantic basis, therefore, would ideally be free (gratis). (The economic basis will be discussed in more detail in section 6.5 on page 126).

In Chapter 4, the success factors for Internet-based developments have been analysed. BC Semantics need to be generally available and simple to allow the knowledge chains to unambiguously express their knowledge. The BC Semantics end-result should therefore be available in a simple, generally understood form that can be used without needing applications that are not generally available or affordable. Access should be unrestrained, both technically and juridically.

In practical terms, the desired end-result should be BC Semantics that are available on-line, for free, in a generic computer-readable

format.

As of the size of the problem domain, it is imperative that a healthy 'ecosystem' develops, in which every interested party can contribute and cooperate. The end-result should therefore allow for a way to do distributed development and maintenance of BC Semantics, without having a centralised organisation serving as a potential choke-point. There is nothing wrong with one organisation having a more dominant role, the Semantics should however not be dependent on that organisation: no monopolisation.

From the analysis, it becomes clear that it is probably a dream to make one single set of BC Semantics that will be used all the time, everywhere. This means that the end-result should be flexible: the definitions should be referenceable, it should be possible to use the definitions as a basis for communication by referencing them. This way one can either use the Semantics directly, or one can use them to communicate between two differing Semantics. The end-result should not force itself to be built directly into everything that uses it; it should be referenced only.

The next section discusses a development-friendly clustering of terms.

6.2.2 Domain specific clusters of terms

While focusing on the object definitions themselves, the first problem is to find a solution for ordering so many BC terms used by the stakeholders active in BC: civil engineering, house construction, infrastructure, architecture, waterworks and other sectors.

The previous section concluded that the end-result should allow for distributed development and maintenance of the BC Semantics. Distributed development calls for a subdivision of the problem to split up the work. A second reason is to provide a measure of local organisation to coordinate the work.

Which demands should be satisfied by a clustering of terms? The overarching goal is to achieve a complete set of terms, capable of supporting BC. Smoothing and supporting the actual development of that set of terms should therefore be the top demand. A simple and

clear process allows the greatest participation. A clustering should be clear and simple for the people adding and maintaining the terms.

A first subdivision of the problem area is to restrict the development to a national scale instead of an international scale, at least initially. Not having to deal with translation issues and local practices makes the process much simpler for most participants.

A second possibility is to cluster by work discipline or sector: bridges, installations, roofs. The resulting clusters are quite big, but the majority of the terms will be familiar to the participants. The clusters are even bigger if you cluster by ground/waterworks, buildings, equipment, etc.

A third possibility is to cluster by supplier industry. The suitability depends on the level of participation of suppliers, but in any case the resulting clusters will tend to be smaller than sector clusters. This might make it more manageable on the one hand, but too small for meaningful interrelations on the other hand.

For these sets of term definitions, the word *Ontology* is often used. An Ontology[1] is a conceptualisation (meaning definitions of terms) of a specific domain (a certain cluster), made from the viewpoint of that domain [3]. The term itself acknowledges the fact that an Ontology is restricted to a certain domain and it is made from the specific viewpoint of that domain.

The word used from now on to indicate BC Semantics, so the definitions of terms, is 'Building-Construction Ontology'.

The technical implementation of subdividing the problem area into several Ontologies will be done using the NG Internet Ontology Web Language (OWL) standard (analysed in chapter 4). Every cluster will be given its own OWL file. For unique identification of the terms in an OWL Ontology, OWL uses the NG Internet mechanism of giving every term its own Uniform Resource Locator (URL). Every OWL

[1]The way the term *Ontology* is used here, it implies that you can say *Ontologies* to indicate a number of domain conceptualisations. This is consistent with the way the term is used most often nowadays. *Ontology* can also be understood as meaning just the science field, in which case there can only ever be one Ontology. This is not the way the term is used in this thesis. Just trying to comfort some well-trodden toes...

Ontology should get its own 'base' URL. This way, no Ontology can overlap another, at least not on a technical level. An English door Ontology might start all its term identifications with `http://en.bcoweb.org/doors/` and a German bridge-building Ontology with `http://de.bcoweb.org/bruecke/`.

The next section shows how these cluster-based Ontologies can be related to each other in a network.

6.2.3 Building-Construction Ontology web

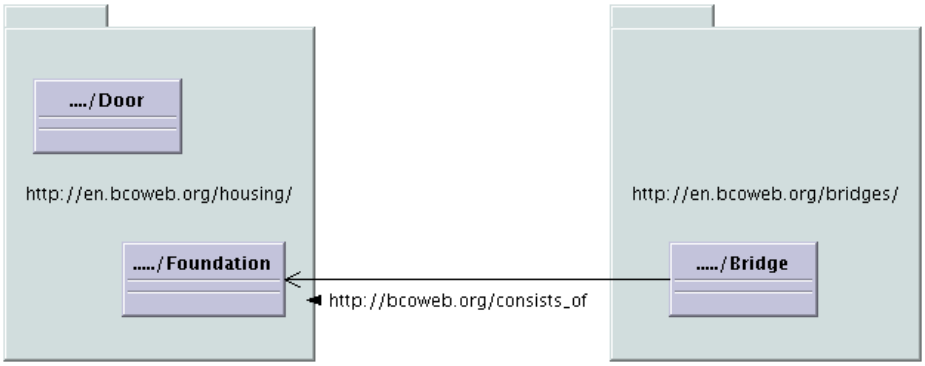
The clustering explained above greatly helps to make the development process simpler for the participants. BC as a whole, though, should not be limited or hindered by the development-friendly clustering. What is needed is a way to be able to see all the different Ontologies as one whole or as one connected network, so that BC as a whole can be supported.

A second reason to desire a connected network is the duplicated effort the participants need to expend if every cluster has to develop their own solution for common items. For example, foundations occur under houses, but also under stadiums and bridges[2]. Every cluster would need to enter the same information; information that should preferably be shared. Figure 6.1 on the facing page shows a simple schematic example.

Instead of one single unified Ontology, multiple separate Ontologies are proposed. This allows an individual company or industry sector to create and maintain their own Semantics. It also obviates the need for a central controlling entity. To allow generic usage and to allow improvements (which might be needed in such a less-controlled, decentralised approach), a specific open source license (GNU Free Documentation License (FDL) [4] or Creative Commons (CC) [5]) on the content is part of the concept—as was recommended as a very interesting new effort at creating a useful basis for web-based communication in Chapter 4.

[2]There are differences in the technical details of the various foundations, but there are more similarities than differences. Alternative foundation types that are not relevant to for instance houses can be left out by a restriction mechanism.

Figure 6.1: *UML diagram illustrating two cooperating Ontologies. A bridge Ontology's 'Bridge' class references a housing Ontology's 'Foundation' class.*



The distributed approach necessitates a linking-together of the Ontologies on their contact points: a network of partially cooperating Ontologies is envisioned.

The OWL Ontology format is build on RDF, the linking mechanism of the NG Internet. This allows you to link from one item in an Ontology (identified by a URL) to another item in another Ontology (also identified by a URL). This makes it possible to say that one of the 'parts' of a 'bridge' (in a Civil Engineering Ontology) is a 'foundation' (which is found in a Housing Ontology).

In this way, a network of Ontologies can provide BC with the Semantics needed. The drawback of having the Semantics spread out over multiple Ontologies is negated. In keeping with the Internet, the Ontology network will be called a web: the *Building-Construction Ontology Web (bcoWeb)*.

6.2.4 Structuring the object definitions in the Ontology

Object definitions can be structured in many ways, see the analysis in chapter 3 and chapter 4.

As supporting a more dynamic BC is a base requirement, the

interaction between Demand and Supply is important. The General AEC Reference Model (GARM)[3] Functional Unit (FU)/Technical Solution (TS) model supports the Demand/Supply distinction on the Ontology level and is proposed as a basis, see section 3.3.4 on page 44. The FU/TS distinction provides a natural contact point to link together Ontologies in a network.

Following ideas expressed by Van Nederveen [6] and Gieling [7] terms are divided in 'functional units' (called 'functions' by Van Nederveen) and 'technical solutions' (called 'function performers' by Van Nederveen). Demanding parties express their demands in functional terms, and supplying parties express their supply or offering in technical terms. For each functional unit a number of technical solutions (function performers) can be provided.

A specific goal is to help close the gap between Demand and Supply. Therefore the starting point is to take the selection of a particular technical solution for a given set of functions into the core of the metamodel. Each object, large or small, can be viewed from a functional perspective (what) and a technical perspective (how). The functional characteristics of an object are collected in what is here called a Functional Unit (FU). The technical characteristics are collected in what is here called a Technical Solution (TS). Selecting the appropriate TS among a number of alternative TSs is represented in figure 6.2 on the facing page.

Once the basic FUs and TSs have been made available in the Ontology web, new TSs can be added by the Supply side and subsequently be found and evaluated by the Demand side. Also note that the Unified Modeling Language (UML) model says that a TS can satisfy multiple FUs at the same time, *i.e.* a particular type of 'outer wall' can fulfill both separation requirements (temperature, noise, safety) and structural safety requirements. Note also that a particular FU can be satisfied by the technical characteristics of more than one TS.

The second key construct is the fact that each TS is made up of a set of lower objects that again can be viewed from both a functional and technical perspective. Figure 6.3 on the next page formalises this

[3]When looking up 'GARM' in google, you mostly find references to a hell hound from Norse mythology...

Figure 6.2: *UML diagram describing the information required for the selection of a particular TS for a FU*

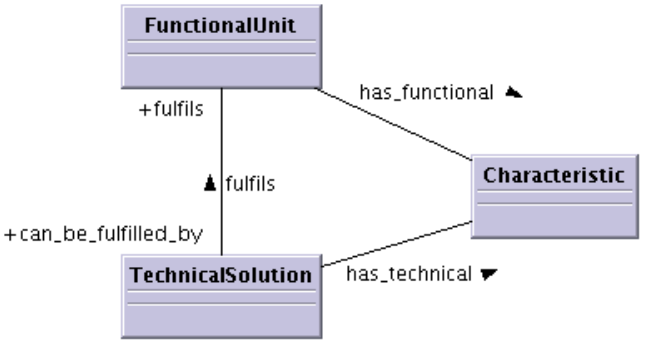
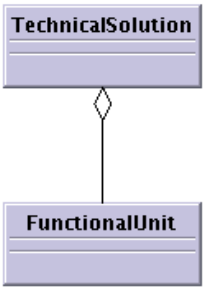


Figure 6.3: *UML diagram expressing the fact that TSs can be seen to consist of lower order FUs.*

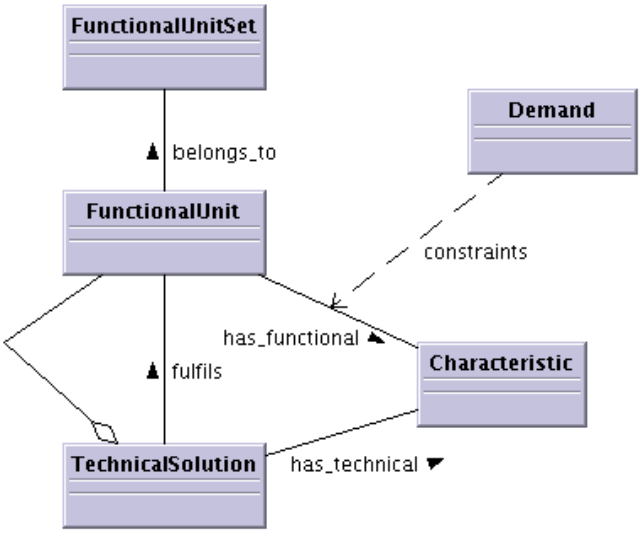


fact.

Besides the simple FU and TS, the concept suggests one other object type: the FU-set[4]. The FU-set is a collection of objects that

[4]In the implementation, a second object type is added: the abstract TS. The abstract TS supertype's only purpose is to aid in development. The intention is to never ever bring the end-users into contact with this abstract object. During development it is sometimes handy to group a set of fulfills/consist_of relations that occur a lot in one place. This set of relations can then be used by the real technical solutions, saving you the effort to add those relations to each and every one of those technical solutions. The abstract TS supertype is always postfixed with the word "Solution". For example the abstract TS super type 'beam solution' is used to describe the common characteristics of all the TSs that are able to fulfill the functional requirements of the FU 'beam'.

Figure 6.4: *Demand/supply-oriented meta model. The core is that a TS consists of multiple FUs and that a FU is fulfilled by one or more TSs. Both a TS and a FU have characteristics: the characteristics of a FU are constraint by the demand. Often, FUs come in groups of common solutions: an FU set.*



collects a set of FUs. The reason for its introduction is that it can be used to express that objects like 'water crossing' are more a grouping of FUs than that they are a FU themselves. The FUs contained in the set 'water crossing' can then be 'tunnel', 'bridge', 'ferry', and such. Modelled like this the FU 'bridge' can then be matched to various types of bridges, like 'suspension bridge'.

Figure 6.4 shows the complete meta model. Note that there is also an object called Demand which constrains the functional characteristics to become a demand (height < 30.0cm).

At this point, the model is really a simple version of the original GARM model: a lot of things are missing, like restrictions and properties. The simpleness of the model has attractive sides: it is relatively easy to explain and easy to work with. As an example, one graduate student added some 1000 concepts including relations in three months time [8].

Properties normally should be integrated into the model, but the use of OWL allows the possibility of keeping the properties external. In the external case, the FU/TS data is combined with one or more OWL files defining the properties before being used by data. Using OWL's possibility to combine several classes into one new class, the various source classes can be kept separate without cross-pollution.

An advantage of keeping the properties out of the FU/TS data is that the model stays simple and focused. A disadvantage is that properties are very important: keeping them external might make the FU/TS data less practice-oriented, as adding properties to a concept makes you think harder about that concept's reality. External properties allow the use of various abstraction levels. Not everybody is interested in the same properties or the same amount of properties. External properties invite you explicitly to take this into account, which might lead to results that are more practice-oriented.

Including properties directly in the FU/TS data makes the model less simple, but the importance of properties warrants that. A problem is the variance in abstraction levels at which users of the system think: for ordering a window, five properties might be enough, but, for making a window in a Computer Aided Manufacturing (CAM) setting, 85 properties are needed [9]. It seems clear that to keep the system workable, at least some properties have to be kept external. This is supported by OWL, so it presents no technical problems.

Apart from properties, restrictions are also part of the original GARM. Functional units are generic building blocks that can be re-used in other parts of the model, for instance a FU 'foundation' that is used by bridges, houses, stadiums. These FU building blocks are sometimes too generic: not all technical solutions for that FU are applicable for a certain object, so the non-applicable TSs have to be filtered out.

As an example, a 'bungalow' in the Netherlands indicates a house with just a floor on ground level and with a flat roof. For the technical solution 'bungalow', it would be handy to re-use FU 'foundation', FU 'roof' and so on. But FU 'roof' includes TSs that are not allowed for 'bungalow'. The simplest way to solve this is to restrict the list of TSs for a certain FU when some object higher up in the hierarchy occurs.

This restriction mechanism does not directly cover restrictions on properties or more complex interactions, but it provides a basis for further elaboration. It should be implemented in such a way, using OWL's extension mechanisms, that allows others to add more restrictions. The restrictions can be a good way of adding *knowledge* to the system.

A final word about instances of a FU/TS model. The tree structure invites you to use a tree structure on the data side (=instance), too. For many uses this might be a good solution, as bcoWeb aims to keep the FUs and TSs close to practice, something reinforced by the model.

When looking at the so-called *objecttree* solution, a difference is that an objecttree consists of per-project individual objects or sets of objects [6]. A bcoWeb-based instantiation tree uses general definitions instead of locally (un)defined objects. A combination of the two might be attractive: instantiated bcoWeb objects when possible, interspersed with locally defined objects. The locally defined objects, when useful, can be considered for addition[5] to bcoWeb, either as a separate Ontology or added to existing ones.

6.2.5 Cooperation with other initiatives

Other initiatives to create BC Ontologies exist (see chapter 3 and 4) and for instance Classification systems have a fixed place. The acknowledgement that cooperation with other initiatives is needed is inherent, in a way, to a concept that itself consists of a Web of cooperating Ontologies.

Interaction can take place in three ways: directly on the Ontology level, on the data level and 'via the data'. On the Ontology level means direct mapping from the Ontology to a Classification system or a Product Data Technology (PDT) standard. On the data level, the second way, the data is linked to the Ontology and to another system. In this case, the Ontology needs to know nothing, but it means double work on the data level. A third way is to do the linking

[5]When such a project instantiation is available over the Internet, mining tools can select the most promising candidates for inclusion automatically.

on the data level, but to try to extract an Ontology-level mapping out of the data level links later on. This can be done automatically, but is highly dependent on the accessibility of the data and and the availability of a sufficient data quantity.

Classification systems can be very important for Ontologies as they summarise and organise existing knowledge [10], connecting Ontologies to existing experience and perhaps making them more instantly familiar.

When dealing with Classification systems, the easiest approach is to directly classify the BC Ontology classes in the Classification's classes. All BC Ontology 'door' objects should end up in STABU's 'doors and windows' category, for instance. The nature of BC Ontologies and of Classification systems is an advantage here. BC Ontologies will have relatively many object classes, as it should be detailed enough for meaningful computer-based information exchange. Classification systems, on the other hand, have relatively few classes, as it would otherwise become unclear and unmanageable for the human actors that it is intended for. Classifying a detailed BC Ontology in the appropriate Classification classes will leave just a few border cases that are difficult to classify.

With regard to PDT efforts like Industry Foundation Classes (IFC) or 12006-3, Ontology-level cooperation is more difficult because the level of detail does not differ that much. This similarity in detail makes it unlikely that most classes fit exactly, because of the difference in viewpoints of the Ontologies. This makes for a lot of corner cases: partly overlapping classes.

The actual BC objects described are the same, so on the data level it is clearer which Ontology classes (object definitions) should be used. Though it is more work, you could reference more than one Ontology when needed. This is a data-level mapping. In certain cases, a data object could be seen as an instance of two Ontology classes.

The BC Ontologies proposed are each a domain-specific cluster of terms. Within individual clusters, a more Ontology-level mapping could be made if another Ontology is also used a lot in that domain. Another option is to try to recast another Ontology that is popular in a certain domain into a more demand/supply viewpoint by providing

a thin wrapper.

The BC Ontology concept proposed provides an additional angle for cooperation. As the proposed concept is based on a supply/demand viewpoint, there are two possibilities for cooperation: on the functional side and on the technical side. If another Ontology is more function-oriented, cooperation just on the demand side is a good opportunity, for instance.

Mappings made on the data level can be automatically turned into partly Ontology-level mappings if enough data level mappings are available. The next section, on web services, shows that data can be automatically mined for useful information. One of the possible applications is the generation of a first-effort mapping, which can later be enhanced.

6.2.6 Conclusion on Building-Construction Semantics

A network of national domain-specific object definitions should be created. By making them on-line available, they are referenceable and can be used to improve the communication between all stakeholders, including the general public. The concept is called Building-Construction Ontology Web (bcoWeb).

A clustering in national domain-specific Ontologies allows for a distributed development, making the concept developer-friendly. The drawback of separate Ontologies is negated by placing them in a NG Internet-supported network.

The Ontology structure is based upon a Supply/Demand distinction by using the FU/TS principle from the GARM, which has not been used on such a level. This principle has the advantage that it specifically targets the interaction points between objects and thus communication itself. It provides BC with a growth opportunity towards further integration and knowledge sharing.

6.3 Web services

Web services[6] expose application data via the Internet, as discussed in Chapter 4. It is a standard way to allow applications to interact with other applications via the Internet. This opens up a lot of extra possibilities and it gives computer applications a more active, more value adding role in the BC process.

Documents in BC are currently created after-the-fact and then passed on to the next phase in BC, without much possibilities for change or interaction. A good example is the building Specification: a static document, signalling the end of the design phase. This passive crop of documents should be replaced by a more active type of document, a smart document that 'understands' its purpose and its content and is able to see that it gets things done. This vision in fact means that smart documents, like robots and intelligent manipulators, are also treated as stakeholders in the process.

In the same way, applications can be treated more as participants, instead of just as tools used in the background by the actual human participants. If applications are conceptually given a more participating role, they can provide a lot of value to the process. BC web services provide a way for documents and applications to become participants.

6.3.1 Different types of BC web services

Web services for BC can take on various forms. Project databases (like big IFC Express databases) can be opened up to Internet access. Calculations can be done on-line, for instance strength calculations based on an IFC model. Service can be provided, by allowing building Specifications to be edited on-line, based partly on input from a project database for instance, with the Specification data in its turn being used as basis for other applications. Rule checking is another promising area, checking solutions against regulations or best-practice

[6]Technical note: It probably bears repeating that in this thesis, 'web services' is not intended to be equal to 'SOAP', but that the so-called REST web services style is advocated, using just HTTP+XML+RDF.

databases.

These different types of web services make integration an *a priori* possibility. Many of the examples mentioned above are now done by hand and after the fact. When the building design is finished, there are three days left for making the building Specification. Only then can a real cost estimation be made. *A posteriori* integration, as analysed in Chapter 2, limits BC greatly.

Web services are part of the solution concept, as they make integration a possibility early on in the process. Web Services—on a conceptual level—state that every computer program can be integrated into the BC process. This 'yes, it can be integrated' enables a greater interaction earlier in the process, leading to less friction (and friction costs) later in the process. Current practice means one-off integration afterwards, mostly by hand—leading to high friction costs.

An important part of the concept is to take a deeply distributed approach at information storage. In principle, information can be stored everywhere, as long as the information items are available using the Internet. Web services, as intended here, also include a simple putting on-line of data files as the simplest way of web services.

Tightly coupling applications and databases is impractical, as the coupling might never be used again. One-way referrals from one application or document to the needed item in another application or document are much more practical. Identify what is needed in order to be able to search for updates to this information.

Existing applications can be fitted out with a web service interface, for the common desktop applications this will mostly allow them to consume data, not make them into an information source themselves. But, for instance, TNO's IFC browser [11] could be made into a web service with relatively little effort, exposing the rich IFC data for consumption by other value adding applications.

Also project extranets (see 2.3 on page 20) could be turned into an information provider. It is quite natural to start swapping small pieces of functionality of a project extranet for more semantic alternatives: one at a time. Such a swap is a good measurement point, too: it should be better, not worse, than the original solution.

Much additional benefit will be possible if the web services use

the bcoWeb Semantics, as without common Semantics the use that can be gotten out of web services will still be limited. The solution concept assumes the use of bcoWeb in combination with web services.

Entirely new applications are made possible by web services. The value-price-cost model [12] [13] needs frequent re-calculations during the process in order to be practical. Applications that perform such calculations are made possible in a web services setting.

The next section introduces the knowledge sharing possibilities that are also newly made possible.

6.3.2 Knowledge sharing

Web services lower the costs (in time and effort) of integration of applications in the BC process. This allows an increase in computer-provided information and—more specifically—knowledge.

Chapter 4 analysed some of the knowledge representation possibilities in the NG Internet. These are also the possibilities that can be integrated into the BC process using web services. The simplest form is knowledge contained in the BC Ontology, like the common parts of a bridge or stadium. Or the different kinds of roofs. More involved is restriction or rule-based knowledge. For instance, the fact flat tarred roofs must be re-done every 10 years.

Often, a (theoretical) subdivision is made in data, information and knowledge. Data is the 'raw' data; information is data tagged with meaning so that it is usable; knowledge is 'active' information, information that is being used, information that is consciously acted upon. Using this common subdivision for this thesis, Ontologies are used to tag data with meaning, turning it into information.

Knowledge goes one step further—this chapter provides two ways. On the one hand, the use of the FU/TS-mechanism as the Ontology structuring mechanism brings the dynamic supply/demand, question/answer, wish/possibility back-and-forth directly into the Ontology, which facilitates the expression of knowledge. On the other hand, web services interacting with the information and the represented knowledge can themselves provide knowledge by acting on it.

Knowledge provision specific to web services is the knowledge that

requires access to multiple information sources, coupled with calculations, lookups in huge databases, etcetera.

This specific kind of web services based knowledge provision opens up huge possibilities for parties in BC that have a lot of knowledge. Examples include standardisation institutes, research institutes (like CSTB, CSIRO, TNO, universities) and, for instance, Contractors that have made knowledge formalisation a priority.

The knowledge present in those institutions is currently only transferable either by consultancy or by paper-based documents. Web services open up the way to make the knowledge available more widely by allowing it to be integrated more directly in the BC process.

Compared to paper-based documents, smart documents also allow a more detailed and comprehensive transfer of knowledge. Paper-based documents are restricted in size because they must be manageable by humans and because they must be economically viable. Web services based electronic knowledge support of the BC process knows no such limits.

The integration of knowledge into the process opens a *knowledge market* in BC, allowing also smaller participants to offer their—possibly highly innovative and specialised—knowledge in an economically viable manner.

6.3.3 Conclusions on web services

Web services open up new possibilities for value adding in BC. They open up project databases for generic application access, allow service providing, rule checking and a much increased knowledge integration opportunity. The knowledge integration possibility opens up a new *knowledge market* in BC, making it a more value adding industry.

Human actors can receive much more support during the process, as web services provide *a priori* integration instead of current after-the-fact by-hand integration. Web services automate what can be automated and support what has to be looked into further by human actors. In both cases, however, the feedback must be speedy.

Web services provide application integration and support knowledge integration. In combination with bcoWeb's network of BC

Ontologies—allowing the applications to talk the same language—a big step to an increase in BC's value adding capacity is taken.

6.4 eSpecification

As a major illustration of the bcoWeb and web services possibilities, an eSpecification is part of the concept. Part of the solution concept is the notion that BC documents are participants in the process: you can ask a Specification-in-progress for information; you can add information to a Specification; a Specification can request information.

As analysed in Chapter 3, Specification texts are central documents in the BC process, being part of the Tender Documents. They have a lot of links to other information sources (like the specification drawing, regulations) and others have links to the Specification (cost calculations, etc.). A Specification traditionally is the hinge between the design phase and the construction phase and is as such an important research target when trying to improve the value adding capacity of BC by making the process more dynamic: the current Specifications are static.

An important reason why current Specifications are static is that they are essentially paper documents: a representation of the real data (see section 3.2.3 on page 34).

eSpecifications are made part of the concept in order to put bcoWeb through its paces. With an eSpecification, a computer-supported translation from design to construction, that complies to the regulations, must be made possible.

How far can eSpecifications be taken? The stated goal is to automatically generate a Specification from bcoWeb-enabled design 'eDocuments'. The extend in which this goal can be achieved is a measure of the suitability of bcoWeb for improving BC's value adding process.

BcoWeb data can be instantiated in a Specification system. While keeping the Specification's Classification structure, the various parts of the tree can be placed in the Specification text. A powerful notion here is the formal distinction between functional and technical, allowing a more gradual development of the Specification. It also

makes early versions of the Specification valuable for feeding into the project's communication, as the earliest high-level functional descriptions can already be used for knowledge-addition, searching for alternatives, coupling with known solutions and so on.

6.5 Proposed business model

What has been discussed in this chapter till now are solution concepts for construction Semantics, the web services that build upon the Semantics, and Specifications as an important illustration of the concept. This section proposes a business model as integral part of the concept: the concept is intended to be applicable in practice.

A desired end-result of the concept solution is a healthy BC information ecosystem. This section introduces the proposed business model for the concept. The concept is not intended to be a theoretical model, but to be *applicable in practice*, solving real-world problems and improving BC's value adding.

The two main components of the concept, the Building-Construction Ontology Web (bcoWeb) and the web services will be dealt with separately with regard to a business model. An important basis for the potential success of both is formed by their use of standard NG Internet possibilities that are widely available.

6.5.1 bcoWeb business model

Part of the solution concept introduced in this chapter is to use an open source development model [14] for bcoWeb. The analysis in chapter 3 showed that current development efforts have failed to produce widely-accepted results, thus making an alternative approach an attractive option. Open source itself has been analysed in chapter 4 (section 4.6 on page 83, see also section 6.2.1 on page 109).

With an open source development model, the following points are part of the characteristics:

Zero costs The cost for use of bcoWeb is zero. This means that there are no monetary restrictions for adoption, making it inher-

ently more attractive for users and developers when compared to alternatives that require payment [15]. The downside is that there is no *direct* monetary profit to be gained from building bcoWeb. This means that development and, for instance, web server space, must be financed through other means [16]. Possible incentives to do this will be covered later in this section.

Cooperation BcoWeb does not need to be developed by just one single organisation or company. Open source licences allow you to develop bcoWeb cooperatively, with multiple partners developing parts of bcoWeb without needing elaborate legal frameworks to do so. It is possible to build upon each other's results and to re-use previous results. The clustering proposed helps to organise cooperation as the group within which cooperation takes place is of a limited size [17].

Small contributions Small contributions to bcoWeb are already valuable. If an individual supplier wants to include his—limited—generic product type information in bcoWeb, it is an addition that can be used. On its own, the information is not marketable or sellable. The open source development model of bcoWeb allows contributions like this to be used.

Possible reasons to finance bcoWeb either with money, effort, information or web server space include the following. Profits reaped indirectly from bcoWeb by deploying bcoWeb-based web services can be used to further bcoWeb's quality and quantity. Government or trade organisation's subsidies can be sought; a good selling point is that the open source license makes the subsidy's results generally and permanently available which might make it a more attractive subsidy target than funding proprietary developments. Self interest of BC companies is also a possible investment reason, as bcoWeb might improve a company's value adding process, even when only implemented locally for that company; with a limited degree of completeness, bcoWeb should already provide profitable results.

As no open source BC Ontology effort has been attempted, it cannot be guaranteed that this approach will work. A prototype

effort has to be undertaken to get a better understanding. What *is* possible now is to discuss two common objections encountered often [18] when discussing early open source Ontology ideas in this field.

Objection 1: it is not possible to earn money with it This point is correct, in so far that one cannot *directly* earn money with the BC Ontology web. The bcoWeb improves the value adding process, so money earned could be spend on improving the bcoWeb to enable more value adding.

When money is earned directly with the Ontology, it means logically that a certain organisation is paid for the the use of the Ontology. The consequence is that that company has a lot of influence on all communication in BC based on the Ontology. It seems questionable whether BC as a whole is willing to submit to such a single-organisation influence. On the other hand it does not seem likely that other stakeholders than public authorities become early involvers.

Objection 2: nobody will maintain the Ontology for free The argument goes that something that is free cannot be good, as there is no incentive to provide quality and maintenance. This argument can be disproved by the success and quality of a lot of open source products. Commercial web server and mail server programs form a minority on the Internet; the open source alternatives are used more often and are often even of higher quality.

Perhaps a better example is formed by the wikipedia on-line open source encyclopedia effort [19]. Despite occasional problems with political issues and personal agendas of some participants, it continues to grow and provides often surprisingly high-quality content. The ease by which interested volunteers can create new content and improve existing content is instrumental in both growth and quality, as the efforts of many small contributors are combined into one useful whole.

A Building-Construction Ontology Web being run as a collaborative open source project. That is the foundation for the best chance

of success for instrumentation that supports and enables a much more dynamic BC.

6.5.2 Web services business model

Web services itself are not a piece of infrastructure that has to be build, it is essentially a way of building and dealing with applications. So, the question becomes 'what is the business model behind building bcoWeb based web services applications'.

Web services provide additional functionality for the user-facing applications. They provide interaction with calculation software, Specification software, regulations, project databases. So there are two different kinds of business models: those for the web services producers and for the consumers.

For applications that consume web services, it opens up a way to offer more functionality with relatively little effort (compared to developing the used functionality inside the application itself). It offers better service to the customers by better and more natural integration with other applications. For example, a bcoWeb-based Specification web service is more attractive to develop for than adding an import/export function for a Specification system's data format, as the latter only works for customers that have that Specification system installed. A per-use fee based Specification web service increases the likelihood that the functionality offered will be used. The market increases enormously, which makes it more attractive to develop.

For web service providers, new markets open up and existing markets get bigger. An example of the increased market was given in the previous paragraph. New markets are opened up for applications that are too small and specialised to warrant separate desktop applications. Too few people get enough use out of it in order to buy the applications, making them economically unviable. With web services, the functionality can be offered to other applications without needing to install software on the client's computer. Innovative and highly technical niche applications, for instance, get a much bigger chance this way; the barrier for adoption is lowered.

What makes this interesting business-plan-wise is that, in most

cases, cooperation between just two applications already seems to profit from a web services approach. Every additional application joining in adds to the advantage. This is not a ecosystem that needs total market adoption before bringing in fruits [20], it can bootstrap[7] itself from a humble beginning.

6.6 Illustrating the changes to the BC information system

With the concept explained and business cases for development discussed, this section discusses the changes this can bring to BC.

By the proposed development of a national bcoWeb and related BC web services, the BC Information System can finally and gradually change from its current paper-based nature into an electronic document based Information System. Paper-based documents and 'dumb' electronic documents will be produced, if desired, as representations (mark-ups) of BC Ontology data.

bcoWeb and web services allow innovative changes in BC. These changes take place at the system, project, company and task level. The changes at these levels are discussed below.

System level innovation On the system (or industry) level, the actual mechanisms for projects that fulfill Clients' wishes are in place. In this system, strengthening the knowledge chains is important. BcoWeb enables the industry to get better use out of existing knowledge and information: knowledge and experience from previous projects; improved utilisation of suppliers' knowledge; improved capture of Clients' requirements, coupled with a better retaining of these requirements; a contractor that stores knowledge for reuse in later projects; knowledge of the research institutes that is, currently, only transferred through

[7]Bootstrapping is a term from the *Baron von Münchhausen* story, where the Baron pulls himself out of the swamp with the straps of his own boots. In ICT it is used to indicate the point where an idea or technology can stand on its own without needing outside support, it can fuel further growth on its own.

regulations, which is only a weak transfer mechanism as paper documents can only hold so much information. BcoWeb enables a promising start for system level innovation.

Project level innovation On the project level, bcoWeb allows for more cooperation possibilities. Information and knowledge sharing among project partners is made easier and more powerful. Likewise, the cooperation between the various phases can switch more from a sequential mode of cooperation to a more concurrent one. Within a project, possible alternative solutions can be compared and weighed more often and more easily, possibly resulting in better solutions. Web services, for example, provide a way to compare the costs and the value, providing rapid feedback on design decisions or change requests, which is vital for innovative kinds of contracts.

Company level innovation On the company level, the innovation is mainly regarding knowledge. On the one hand, a company can sell their own (practical) knowledge, something not currently feasible except for specialised consulting companies. On the other hand, they can get more specialised support through web services, integrated with their own applications. This might lead to a more knowledge-driven BC with a higher value adding capacity.

Task level innovation On the level of the individual task, an improved information dissemination and information access possibility allows tasks to be more knowledge-intensive. Workers can be provided with better instructions, tailor-made for the task at hand[8]. Workers on their part can, through the Internet-accessible information systems, provide much more feedback on tasks and on the design. Likewise, their progress reports can be much more instantaneous, removing a few hours or a day of

[8]Imagine having an A4 format paper placed each morning at each work location with the relevant list of materials (as a checklist), the relevant part of the drawings and a work description. All enabled by the generic and ubiquitous information access. Except for perhaps the top 1% companies, this is currently just a dream.

time lag out of the system. Good companies can use this to innovatively reduce failures and rework. The knowledge gained at the task level can be used in further projects.

6.7 Conclusions

The hypothesis presented in this chapter is that the value adding performance of the BC industry can be much increased by:

- Stimulating the development of domain specific BC Ontologies containing definitions of BC terms following the structuring mechanisms described above.
- Relating the different BC Ontologies in a national bcoWeb.
- Stimulating the development of co-operative BC web services that 'understand' the BC terms and can be used as references to support communication between all the stakeholders—including the general public—where computer applications and building documents are also seen as stakeholders.

If, like in the Netherlands, the government wants to have a more open, value adding and innovative BC sector, participation in initiatives like the one experimented here is necessary. Without instruments to improve communication and to raise the value adding capacity of BC, all the other efforts don't work out in practice. *Without the means to implement new ways of working, those new ways won't come about.*

Bibliography

[1] Frits Tolman. Product modeling standards for the building and construction industry: past, present and future. *Automation in Construction*, 8(3):227–235, February 1999.

[2] Jim Waldo. Why standards?, 2003. Available on-line at `http://www.artima.com/weblogs/viewpost.jsp?thread=4840`.

[3] Reinout van Rees. Clarity in the usage of the terms ontology, taxonomy and classification. In Robert Amor, editor, *Construction IT bridging the distance*, W78 international conference. CIB-W78, University of Auckland, 2003. Available on-line at `http://vanrees.org/research/papers/Cib78ConferencePaper2003`.

[4] Free Software Foundation. GNU free documentation license, November 2002. Available on-line at `http://www.gnu.org/copyleft/fdl.html`.

[5] Creative Commons. Creative Commons website, 2004. Available on-line at `http://creativecommons.org/`.

[6] Sander van Nederveen and Frits Tolman. Neutral object tree support for inter-discipline communication in large-scale construction. *itcon*, September 2001. Available on-line at `http://itcon.org/2001/3/`.

[7] Wim Gielingh. General AEC reference model (GARM) an aid for the integration of application specific product definition models. In Per Christiansson, editor, *Conceptual modelling of buildings*, W74+W78 workshop, Lund, Sweden, October 1988. CIB. Available on-line at `http://itc.scix.net/data/works/robots/w78-1988-165.htm`.

[8] Noor Hellemans. Instrumentatie voor het afstemmen van vraag en aanbod in de bouw - bcoWeb: Building-Construction ontology web initiatief. Master's thesis, Delft University of Technology, December 2004. Available on-line at `http://vanrees.org/research/phd/noor/`.

[9] Personal communication of the author with a CAM vendor at 2002's ecppm conference, 2004.

[10] Anders Ekholm. Harmonization of ISO 12006-2 and IFC - a necessary step towards interoperability. In Attila Dikbaş and Raimar Scherer, editors, *eWork and eBusiness in architecture, engineering and construction*, pages 67–75. ECPPM, Balkema, September 2004.

[11] Peter Bonsma. IFC browser website, 2004. Available on-line at http://www.ifcbrowser.org/.

[12] Hennes de Ridder. Process and system innovation in the building industry. In *Conference proceedings - CIB world building congres Toronto*. CIB, 2004.

[13] Hans Schevers. *Demand Support by Virtual Experts - Supporting the Client during the Inception Phase of a Building and Construction Project*. PhD thesis, Delft University of Technology, June 2004.

[14] Eric Steven Raymond. *The Cathedral and the Bazaar*, chapter The Cathedral and the Bazaar. O'Reilly, 2001. Chapter available on-line at http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

[15] Tim O'Reilly. The open source paradigm shift, 2004. Available on-line at http://tim.oreilly.com/articles/paradigmshift_0504.html.

[16] Eric Steven Raymond. *The Cathedral and the Bazaar*, chapter The Magic Cauldron. O'Reilly, 2001. Chapter available on-line at http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/.

[17] Eric Steven Raymond. *The Cathedral and the Bazaar*, chapter Homesteading the Noosphere. O'Reilly, 2001. Chapter available on-line at http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading/.

[18] Personal communication of the author at STABU, 2004.

[19] English wikipedia website. Available on-line at http://en.wikipedia.org.

[20] Tim Bray. Technology prediction success matrix–12: 80/20 point, January 2004. Available on-line at http://www.tbray.org/ongoing/When/200x/2004/01/14/TPSM-8020.

... engineers apparently regard any complex system that's functioning properly as a temporary anomaly.

Tim Bray

7

Technical implementation of the prototype

This chapter describes parts of the prototype implementation of the concept described in the previous chapter. The goal is to provide the technical background of the prototype applications build to demonstrate and test the solution concepts. This might provide valuable insight for other implementers.

After the *introduction*, the second section opens up with a discussion of *programming and development tools used for implementation*, this also includes a short discussion on web-based software development. The third section of this chapter deals with the *Ontology development tool* and the fourth with *web services* developed to test, demonstrate and disseminate the ideas presented in previous chapters. The *conclusions* close off the chapter.

7.1 Introduction

The solution concept described in the previous chapter first requires an instrument that *implements* the bcoWeb model that supports the Ontology developers and the software vendors that want to create bcoWeb compliant web services. Implementation follows the Internet recommendations from chapter 4.

The next chapter discusses the cases that were researched with help of the tools developed in this chapter.

The software implemented is available under an open source license.

7.2 Internet-based software development

This section provides background to web-based development and the specific tools used. First, the (programming) tools are presented: note that this section is not an analysis of all available tools[1], it describes the tools that were used. The tools used are felt to be instrumental in getting good and quick results, though. Second, web-based development as such is explained more fully.

7.2.1 Python, Zope and Plone

Python and Zope are attractive for web programming. Both Python and Zope have a big community that creates a lot of add-ons and modules that—most of them are open source—can be freely reused ('free' meaning both freedom to change and re-distribute and free of charge) [1]. An added benefit is that it provides a platform to build upon that can both be extended and, being open source, is safe to develop for.

Python Python [2] is a high level, object-oriented, dynamically typed language which is regarded as both elegant and powerful,

[1] For some personal observations as background for the choice, see `http://vanrees.org/research/phd/choice`.

suitable for programs both big [3] and small. It is platform-independent (windows, unix/linux, macintosh; recent versions of macintosh OS X even ship it as part of the operating system).

Python code is reportedly three to five times smaller, measured in number of lines of code, than an equivalent C or Java program [4]. Assuming equal suitability to the task, this results in quicker development. And with number of bugs normally being roughly proportional to the amount code written, the number of bugs is lower than in an equivalent Java or C program [5]. These advantages make it a good language to program in, even more so when you need a first version quickly for testing purposes.

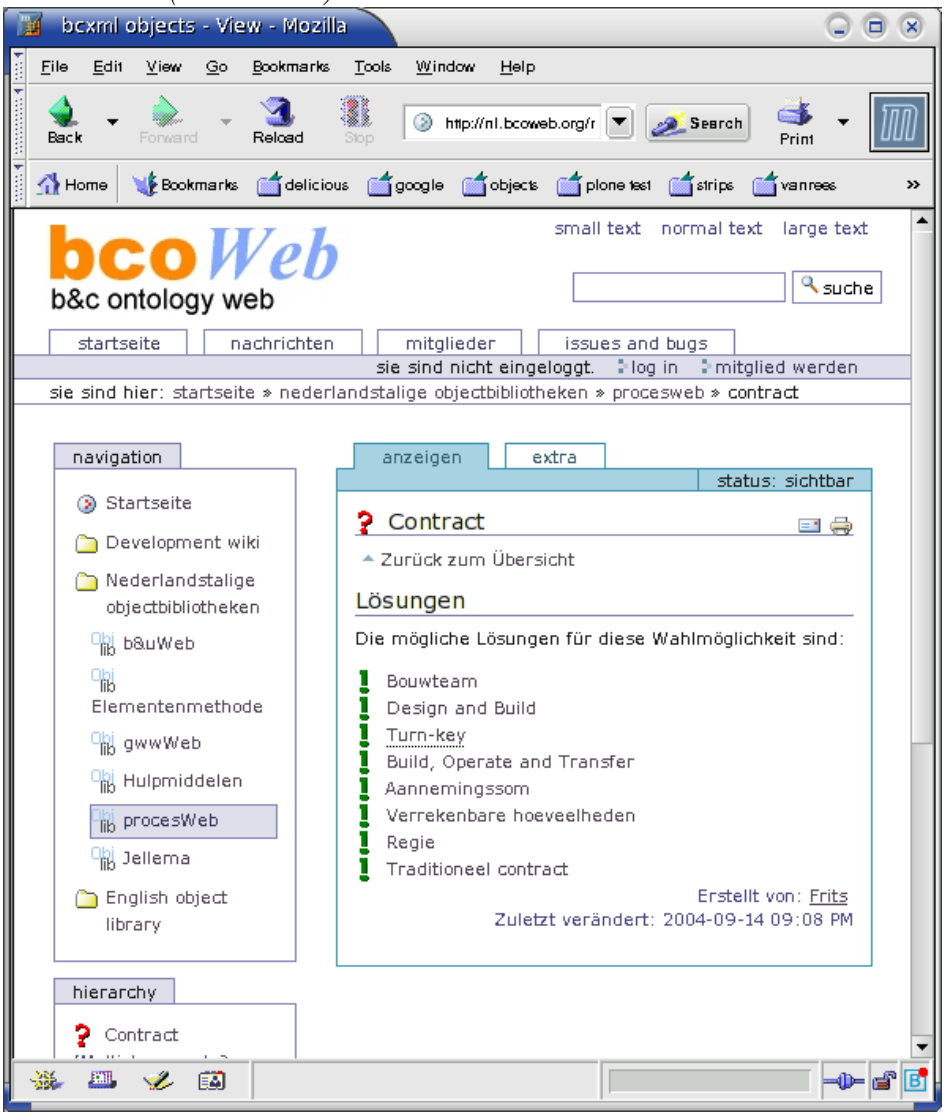
Zope Zope [6] is a web application server (written in Python) with a lot of built-in extras that save a lot of time and effort: a built-in object database; user management and flexible password protection; a through-the-web management interface, so no need for changing files on the file system when this is not desirable.

Built into Zope (and Plone) is the standard internationalisation (translation) mechanism `gettext` [7]. This allows the translation of the complete user interface into all languages. For the bcoWeb editor, at the moment English and Dutch are supported 100%. German (see figure 7.1 on the next page) and French translations have started and are mostly complete. Note that this is just the user interface, not the BC related contents.

Plone Plone [8] is an attractive (but changeable) user interface on top of Zope's. With little effort a good looking and usable result can be obtained (ideal for a time-strapped researcher). Recently, the possibility to partially generate web applications from UML diagrams added even more attractiveness to this solution [9].

By using the Plone user interface you can achieve a good-looking and well-working user interface even with an early prototype. The reason is that a large amount of user interface work is handled by Plone, instead of being handled explicitly by the

Figure 7.1: *Screenshot of the bcoWeb interface, localised for the German language. Note that just the user interface has been translated, the actual contents are (in this case) still Dutch.*



prototype itself [10]. In cases where the default placements are not enough, a thought-out extension mechanism allows you to adapt just those parts that need adapting. This way you don't lose the rest of the automatic user interface generation that did work.

A word of advertisement for open source software is in order here. A small error in a corner case was encountered during development and a 15 minute inspection of the UML generation code (ArchGenXML) exposed the error. In the local version, the small problem was fixed and development continued. Upon submitting the fix to the maintainers, an updated version incorporating the change was issued within two days [11].

The end verdict about this set of technologies is definitively positive. The only drawback is that the Zope/Plone combination has a steep learning curve [12]. The return-on-learning-investment is well worth the cost, though. With an incredibly modest amount of work, a good-looking, attractive user interface for, in this case, a FU/TS-modeller has been made. A standard mechanism for translations is in place, inviting a translation into multiple languages. Generating a lot of code from a UML diagram saves a lot of effort.

Next, web-based software development is discussed. Web-based such as enabled by Zope and Plone.

7.2.2 Web-based software development

Software is increasingly being deployed as a web interface instead of as a desktop application [13]. There are three distinct positive points to web-based applications. (1) There is no need to install applications locally. This can be a huge advantage for less-technical users and users without install privileges. Also the regular updating of applications is done away with. (2) A web interface runs everywhere. There is increasing interest in mobile applications and those devices don't run windows XP with 1600x1200. Also, you don't leave out the increasingly popular linux and macintosh. (3) Due to HyperText Mark-up Language (HTML)'s restrictions, the user interface is typically simpler and less involved. While an elaborate interface is needed for the

likes of ArchiCad, a simpler user interface is normally preferred by the user [14].

The drawback of web-based applications is on the developer's side: he needs a skill-set that is, it appears, not too wide-spread, at least in the BC research community[2]. Much development is done in Rapid Application Development (RAD) tools where the principal focus is on the Graphical User Interface (GUI), which ties it all together. Development for the web requires more by-hand interaction with the code. Also, running a web server permanently (or at least semi-permanently) can be more involved than a desktop application.

Web-based development has an additional advantage for prototype-stage research developments that are client-server based. Client-server requires that the data model on the server side corresponds with the model as the client understands it, resulting in a compatibility problem. Changes in the server's model do happen from time to time in an early exploration phase—like bcoWeb is in now. This leads to back- and forward-compatibility problems. In the worst case, which often happens in exploration-phase research prototypes, the server and the client have to be kept strictly synchronised, both having the correct version.

This compatibility problem can be solved by deploying applications as a web-based interface. The application runs on the server and only interfaces with the client using web pages. With the server and the application both running on one machine, coordination between the two is easy.

7.3 The bcoWeb development tool

Goal of bcoWeb is to allow multiple people to collaboratively edit the same FU/TS-based Ontology. The chosen technical basis is implementation in Python/Zope/Plone and as a web-based application, both explained in the previous section.

[2]As a measurement, a count was made of the CAD/GUI/visual basic developments and the server-related developments for the presentations at 2003's w78 conference. It was approximately 2:5 server:GUI.

First, the UML model of the application is introduced, followed by the application structure. Third, the web services capability of bcoWeb itself. See also the next chapter for a walk-trough of the application.

The open source application source code is available at [15], licensed under the GNU General Public License (GPL).

7.3.1 UML implementation model

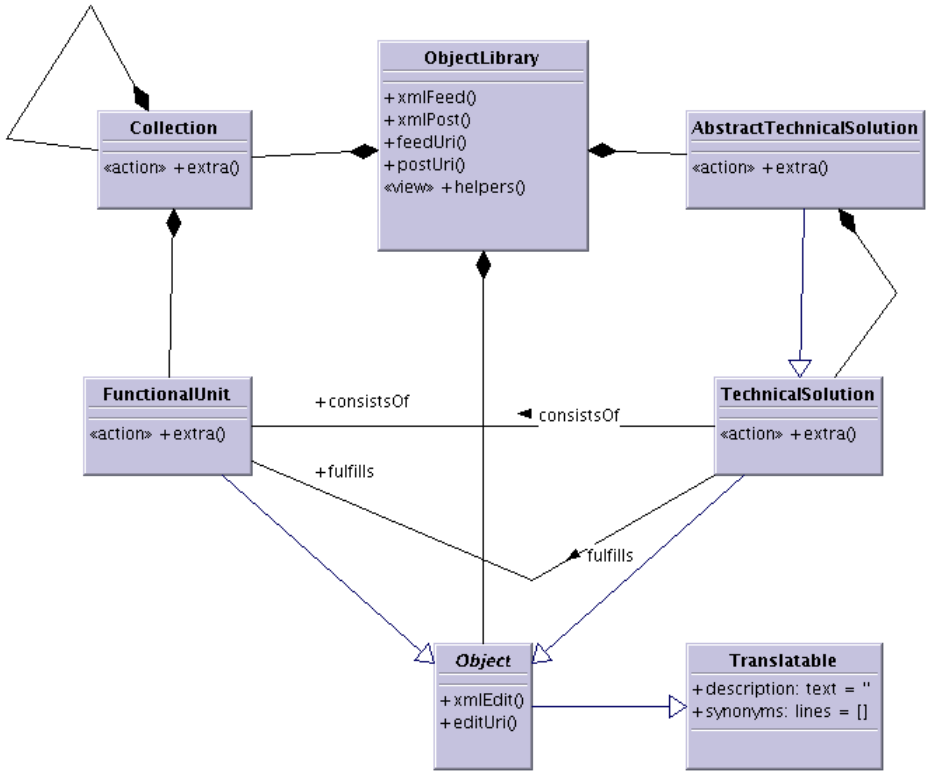
This section discusses the inner workings of the bcoWeb application by looking at the UML model, figure 7.2 on the following page. From this UML model a large part of the web application was generated. As this model is used as the starting point for a code generator, the needs (and occasionally wards) of the code generator weigh more heavily than 100% neat model correctness.

The starting point is the *object library* (Ontology[3]), which can contain all other items. It functions as a folder in which the other items can be placed. Main inhabitants are the *functional unit* and the *technical solution*. The FU/TS-typical *fulfills* and *consists of* relations are implemented on the technical solution, which means that—in the generated user interface—the technical solution is the place where you can add or change these relations.

Both *collection* (functional unit set) and *abstract technical solution* are implemented as subfolders of the object library. The collection is just a grouping mechanism for *functional units*, grouping them under one common name. Abstract technical solutions can contain technical solutions, but implementation-technical also inherit from (sic) the technical solution class. This provides an easy and intuitive way to associate technical solutions with an abstract technical solution, simply by putting the solution in the abstract's folder. The *fulfills* and *consists of* relations from the abstract technical solution to functional units are inherited by the contained technical solution.

[3]In this chapter, 'object library' is used as a synonym for 'Ontology'. Ontology is the more correct term, but object library was chosen in the beginning of the implementation as a seemingly more user friendly term.

Figure 7.2: *BcoWeb UML diagram. This model has been used to generate the Plone product that is the core of bcoWeb. Through the abstract baseclass 'object', the object library can contain both FUs and TSs. These can have 'consists of' and 'fulfills' links to each other. A collection can be used to group common sets of FUs, likewise 'abstract TS' groups common sets of TSs.*



7.3.2 Application structure

Each of the UML classes gets a basic web page auto-generated by the Plone mechanism[4]. For most of the classes, small changes were made by adding explanations and filtering some information.

The classes, implementing the core functionality, are stored in the Zope Object DataBase (ZODB). In this manner, they are both available in the running Zope web server and stored reliably for later use. This is a real advantage of the ZODB, and it does away with the need for a separate storage layer. In this sense, the ZODB is one of the few object databases seeing real use.

For visualising the running objects, Zope page templates are used. An example of such a template is provided in figure 7.3 on the next page. The Zope templating system provides a clean way of separating the actual object's code and the visualisation, always a goal of good programming.

For extra actions that do not really belong in the core classes, the possibility exists to add small Python scripts that can be called on the classes. An example is provided that calculates the FU/TS objects displayed on the starting pages of an object library (see figure 7.4 on page 145). The starting page is meant to be a good starting point for browsing through the object library. The best starting points seem to be the top-level items in the FU/TS hierarchy. Such an item is a functional unit that is not contained by a technical solution. So the database must be searched for such functional units. Afterwards, they should be sorted with the functional unit having the most technical solutions first, as that will put the least-developed functional units near the end.

7.3.3 Web service: exporting the data

Support for NG Internet is build into bcoWeb. The most important feature is the ability to download all contents of an Ontology as an Resource Description Framework (RDF)/OWL file. This file contains the functional units, technical solutions, their names and

[4]Technically: Plone's `archetypes` mechanism.

Figure 7.3: *Extract from the Zope page template that visualises a functional unit.*

```
<h2 i18n:translate="sols">Solutions</h2>
<div tal:define="solutions
                 python:here.getBRefs('fulfills')">
  <div tal:condition="not:solutions">
    <span i18n:translate="nosolutions">
      No solutions defined.
    </span>
  </div>
  <div tal:condition="solutions">
    <p i18n:translate="choicesolsexpl">
      Possible solutions for this choice include:
    </p>
    <div tal:repeat="object solutions">
      <a href="#"
         tal:attributes="href object/absolute_url"
         ><img tal:attributes="src object/getIcon"/>
         <span tal:content="object/title"/>
      </a>
      <span tal:condition="object/description">
        (<span tal:content="object/description"/>)
      </span>
    </div>
  </div>
</div>
```

Figure 7.4: *Python script that calculates the most interesting top-level FU/TS hierarchy items. These items are displayed on the starting page of an FU/TS Ontology.*

```
# context is the object we're called upon
# Note: 'getBRefs' means 'get back references',
# so: reverse relations.
def numberOfTechSols(functionalunit):
    return len(functionalunit.getBRefs('fulfills'))

def compare(fu1,fu2):
    # A higher number of technical solutions means
    # higher up in the list.
    return numberOfTechSols(fu2) - numberOfTechSols(fu1)

collections = context.objectValues('Collection')
functionalunits = context.objectValues('FunctionalUnit')
toplevelItems = []
# Filter out those which are contained, as
# they aren't top-level items
for functionalunit in functionalunits:
    if not functionalunit.getBRefs('consistsOf'):
        toplevelItems.append(functionalunit)
# Sort by number of available technical solutions
toplevelItems.sort(compare)

return [] + collections + toplevelItems
```

their fulfills/consist_of relations. This way, every bcoWeb Ontology is available in machine-readable format on the Internet.

A possibility added in a later stage was to download information on the individual objects, so that a client application could rapidly get at the information it needs without needing to download the entire Ontology. For some classes of applications, this can make all the difference. Just imagine the possibilities presented to mobile phone applications in this way.

7.4 bcoWeb compliant web services

First, the implementation aspects of web services are introduced, followed by a first test of dealing with semantic integration. Third, bcoWeb model checkers that, as web services, help maintain bcoWeb content. Web services that consume, use and augment bcoWeb content are described last.

7.4.1 Introduction

As the proof of the pudding is in the eating, the next chapter presents cases. BcoWeb is an important ingredient, but such an Ontology network needs to be *used* in order to show its value. So this section looks into applications of bcoWeb. An interesting type of applications are the so-called 'web services'. Web services are computer applications that usually reside on the machine of their owner and produce some service on data send to them over the Internet. Ideally there is no human activity involved (though human involvement still will be required in many cases).

Within a week of adding the Ontology download option (see figure 7.5 on the next page for an example output) to bcoWeb, a fellow researcher created a small Java desktop application [16] that could download this file and present its contents in a Java user interface. By working client-side, the responsiveness of the user interface was much better. This work was not known to the bcoWeb authors until finished, showing that NG Internet allows unthought-of new usages of the information to spring up [17].

Figure 7.5: *Example of a bcoWeb Ontology file.*

```
<FunctionalUnit rdf:about=
    "http://en.bcoweb.org/bridges/RoadCrossing">
  <rdfs:label>Road crossing</rdfs:label>
</FunctionalUnit>
<TechnicalSolution rdf:about=
   "http://en.bcoweb.org/bridges/overpass">
  <rdfs:label>Overpass</rdfs:label>
  <fulfills rdf:resource=
    "http://en.bcoweb.org/bridges/RoadCrossing"/>
  <consistsOf rdf:resource=
    "http://en.bcoweb.org/bridges/center_support"/>
  <consistsOf rdf:resource=
    "http://en.bcoweb.org/bridges/BridgeSection"/>
  <consistsOf rdf:resource=
    "http://en.bcoweb.org/bridges/support"/>
  <consistsOf rdf:resource=
  "http://en.bcoweb.org/bridges/protection_construction"
  />
  <consistsOf rdf:resource=
    "http://en.bcoweb.org/bridges/bridge_road"/>
</TechnicalSolution>
<FunctionalUnit rdf:about=
    "http://en.bcoweb.org/bridges/BridgeSection">
  <rdfs:label>Bridge section</rdfs:label>
</FunctionalUnit>
<FunctionalUnit rdf:about=
    "http://en.bcoweb.org/bridges/support">
  <rdfs:label>Support</rdfs:label>
</FunctionalUnit>
<FunctionalUnit rdf:about=
    "http://en.bcoweb.org/bridges/bridge_road">
  <rdfs:label>Bridge road</rdfs:label>
</FunctionalUnit>
```

An interesting feature of web based software for users is that they don't necessarily have to buy a license for application software. Providers can allow them to only pay for actual usage of software (mostly a modest sum). For software providers it might be an interesting addition to existing business, because casual users and small users will usually not buy an expensive license.

Web services allow applications to use functionality provided by other applications. So the advantages of web based software are extended from humans to computers.

One of the most important features of this type of web services is that, if based on a common Ontology like bcoWeb, by definition, they can communicate and co-operate. Until now application software has been developed in isolation. System designers selected the objects of interest most applicable for the application and developed complex data structures including the definitions of the objects of interest. Co-operation with other applications could not really be considered during the design. So, integration was left to be accomplished (if possible) afterwards.

The next paragraphs present a first test at creating software for a semantic integration of Specifications and Ontologies.

7.4.2 Semantic integration testing

Chapter 8 describes the case in which a first test was done on integrating Specifications and Ontologies. The aim was to generate a Specification from an Ontology-based project description, with project description and Ontology both being available on-line. The Specification should also be available for the project tool for display to the user. In the end, the on-line part was foregone[5], but the rest of the test remained and gave useful insight in the semantic integration process.

Starting point for development was a text-based STABU Specification which was converted into an eXtensible Mark-up Language (XML)-formatted file (using a variety of unix text conversion tools).

[5]The files serving as input were downloaded from the Internet, but the program itself was just running locally.

The XML was transformed (using eXtensible Stylesheet Language for Transformations (XSLT)) into two separate semantic web RDF files, one for the Specification's chapter structure and one for the actual Specification items content. The separation was done to reflect the difference in semantic character between Classification data (chapter structure) and Specification data. Note that this is also reflected in the UML diagram of STABU's Specification system, figure 3.3 on page 33. The Classification file contained links to individual Specification items, so that the Specification could be re-assembled.

The project description was an RDF file pointing to concepts in the Ontologies used. To generate a Specification from the project description, the correct way is to do the coupling via the Ontologies. For this coupling, a mapping file was created plus a program to perform the mapping based on the mapping file. Figure 7.6 on the next page provides an overview.

A file was created that specified the mapping between concepts in the Ontologies and Specification items. For example, 'House' is the starting point. The mapping file specifies that for the opening section of the Specification, naming the project, the address and the description of the house have to be extracted from the project file. This mapping was, of course, done in RDF.

The mapping used both 'push' and 'pull'. The Specification 'pulls' the info needed in the opening section (the house's address, the house's description) from the project file. But it just reacts ('push') on a lot of other items. It only includes a section on brick walls when the project file 'pushes' the brick wall to the mapper.

In a way, this simple solution mimics XSLT's behaviour. For the program that performs the mapping, experience with the behaviour of XSLT processors could be re-used.

The result of the mapping process is an RDF file with just Specification items, but without chapter structure. A small program adds the relevant chapters from the separate chapter RDF file, generated previously. This is converted to HTML at the end. Figure 7.7 on page 151 tries to give an impression.

An interesting addition to this process was the conversion of the Dutch SfB Classification table (nl-SfB, *elementenmethode*) to a simi-

Figure 7.6: *Walk-through of specification generation. A hand-made mapping file maps Ontology items to Specification concepts. An instance file describing a project is converted to raw Specification items and subsequently represented in two different Specifications using two different Classification structures.*

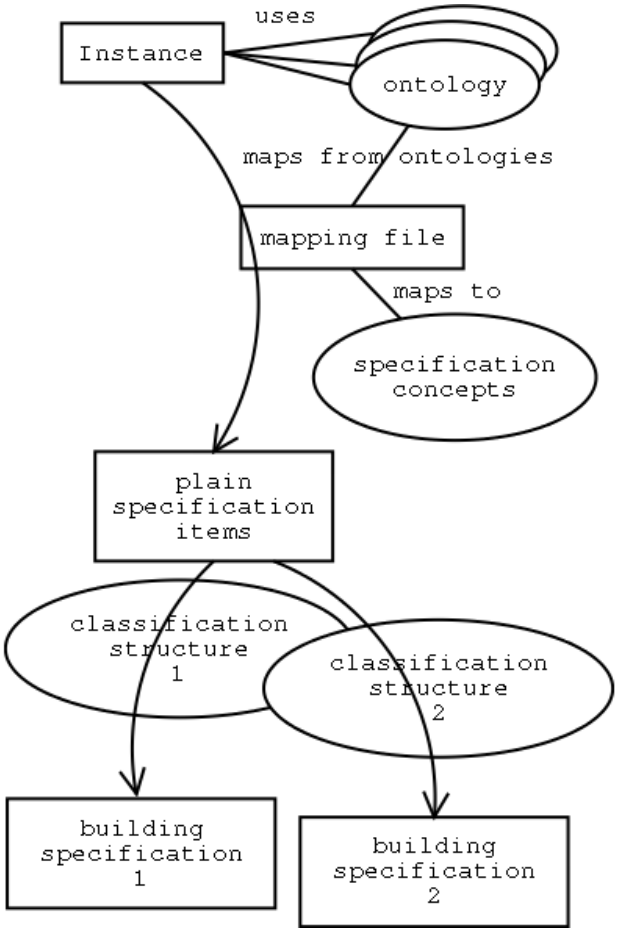
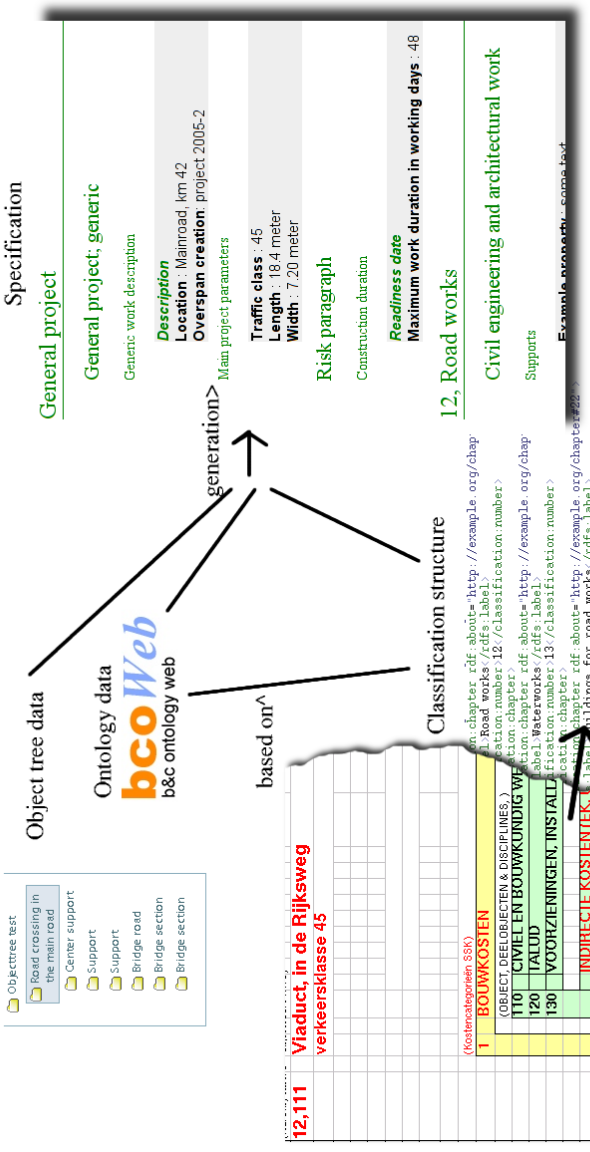


Figure 7.7: *Process of specification generation, see figure 7.6 on the preceding page. Object tree data (based on bcoWeb) serves as instance data. It is combined with a bcoWeb-using Classification structure into a HTML-formatted Specification representation.*



lar chapter structure file and adding the links from chapters to Specification items. Without changing anything regarding the original data, this second chapter structure could be combined with the resulting Specification items and transformed to HTML as an alternative building Specification.

Concluding, linking information with RDF works well. You can connect items together in a one-directional way, without needing mutual coordination. It is possible to flexibly add extra information (like, for instance, a second Classification structure) to existing information. The implementation, however, was small-scale and not itself accessible on-line, so apart from the above points, no extra conclusions can be drawn.

7.4.3 bcoWeb model checkers

There is another advantage to supporting a NG Internet way of working. The FU/TS model—as implemented—is very simple. This does not mean, however, that the need for quality and coordination vanishes, which could be built directly into a more elaborate model. By being able to access the data externally, external supporting programs can check the available data for inconsistencies and omissions. Three of those programs have been implemented: to check for double names, for possible use of available abstract technical solutions (see figure 7.8 on the next page) and for duplicate names (which means candidates for merging).

Upon the simple model explained earlier in this chapter, it is easy to build additional tools. Those tools are, in fact, what makes the simple model possible. The problem as such remains hard, but the hardness should be *transferred* as much as possible to the programmer, not to the volunteer that works with bcoWeb.

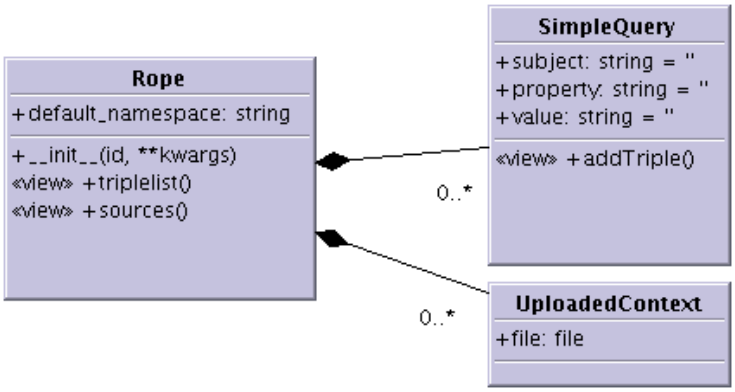
7.4.4 RDF integration

At the core of a bcoWeb approach is the Internet-based exchange of meaningful data. Within the prototype, RDF is used as the format in which to exchange most of the data. Therefore a means to read and

Figure 7.8: *Python script that detects possible candidates for inclusion in an abstract TS.*

```
# context is the object we're called upon

# Ok, look for the FUs we relate to (fulfill
# or consistsOf).
consistsOf = context.getRefs('consistsOf')
fulfills = context.getRefs('fulfills')
# For all those FUs, look what they relate
# to the other way around.
backlinks = {}
for FU in consistsOf:
    backlinks[FU.UID] = FU.getBRefs(
        'consistsOf')
for FU in fulfills:
    backlinks[FU.UID] = FU.getBRefs(
        'fulfills')
# Now we've got a list of back-linked TSs,
# find the ones present in all lists.
candidates = []
if len(backlinks.values()) > 0:
       candidates = backlinks.values()[0]
notAccepted = []
for controlSet in backlinks.values():
    for candidate in candidates:
        if candidate not in controlSet:
            notAccepted.append(candidate)
results = []
for candidate in candidates:
    if ((not candidate in notAccepted)
        and (candidate != context)):
        results.append(candidate)
# Return resulting list
return results
```

Figure 7.9: *UML diagram of Rope, an rdflib+zope combination. This diagram was used to generate an add-on product for the Plone CMS. Rope is the actual RDF storage itself, to which you can add simple queries and to which you can upload RDF files. Rope is used as a basis for the object tree and the catalog application.*



interpret and query RDF is needed.

A choice was made to use Python's `rdflib`, a simple RDF store that parses, stores, queries and exports RDF files [18]. It has the added benefit of being able to use the Zope Object DataBase (ZODB) for storage.

For this research, a Zope product was created that turned rdflib into an integrated part of Zope. The UML diagram is shown in figure 7.9. A difficulty that needed to be overcome was to adapt rdflib, which uses some new Python techniques[6], to Zope, which couldn't yet handle the new techniques.

`Rope` can be used directly, allowing you to upload RDF files and to add simple queries. As this direct functionality is quite restricted, a common way will be to use `Rope` through subclassing it from another product. This route has been chosen in `SimpleCat`, described below, for instance.

[6]The problem were the so-called new-style classes, which at the time couldn't be stored in the ZODB. An amazingly similar product, 'Zemantic', targets the newer Zope version and doesn't have this problem. The latest developments even allow use of rdflib directly.

The name chosen, `Rope`, reflects the combination of rdflib and Zope. The software is available at [19].

7.4.5 Integrating properties

For the showcase in the next chapter, the Ontology's data needs to be enhanced with properties like width and length. Adding properties to the Ontology can be done in a number of ways. Allowing addition of properties into the core bcoWeb editor is a possibility, but you can also do it separately. For interoperability, the addition of properties directly to bcoWeb seems best. For the prototype development, the properties were added by hand in a separate file (see figure 7.10 on the next page for the contents of that file). For the way the prototype works, there is no material difference, though. One RDF file more or less doesn't make a difference.

7.4.6 Catalog application

`SimpleCat` is a simple prototype for a bcoWeb-based catalog application. Its initial aim is to both consume and provide bcoWeb-based data in order to demonstrate bcoWeb's feasibility. Figure 7.11 on page 157 shows the result.

As the base for the implementation a Plone shopping framework, `PloneMall` [20], was used. PloneMall provides the main mechanisms for catalogs, categories, catalog items and even shopping carts (figure 7.12 on page 157), tax handling and payment systems. Note that the shopping cart and payment systems didn't receive any attention during the implementation, but it does strengthen the expressibility of the prototype.

As a technically interesting point, PloneMall uses ArchGenXML UML code generation. This way, there is always an up-to-date UML model that documents the product. As `SimpleCat` also uses ArchGenXML code generation (figure 7.13 on page 158), this was a good fit.

SimpleCat uses Rope (see section 7.4.4 on page 152) to read bcoWeb Ontology files. These files are used for connecting the catalog

Figure 7.10: *Properties added to TS 'overpass'.*

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns........>
  <rdf:Property rdf:about=
      "http://en.bcoweb.org/bridges/width">
  <rdfs:label>width</rdfs:label>
  <rdfs:domain rdf:resource=
      "http://en.bcoweb.org/bridges/overpass"/>
  </rdf:Property>
  <rdf:Property rdf:about=
      "http://en.bcoweb.org/bridges/length">
  <rdfs:label>length</rdfs:label>
  <rdfs:domain rdf:resource=
      "http://en.bcoweb.org/bridges/overpass"/>
  </rdf:Property>
  <rdf:Property rdf:about=
      "http://en.bcoweb.org/bridges/trafficClass">
  <rdfs:label>traffic class</rdfs:label>
  <rdfs:domain rdf:resource=
      "http://en.bcoweb.org/bridges/overpass"/>
  </rdf:Property>
</rdf:RDF>
```

Figure 7.11: *Screenshot of the catalog application, including a link to the bcoWeb definition*

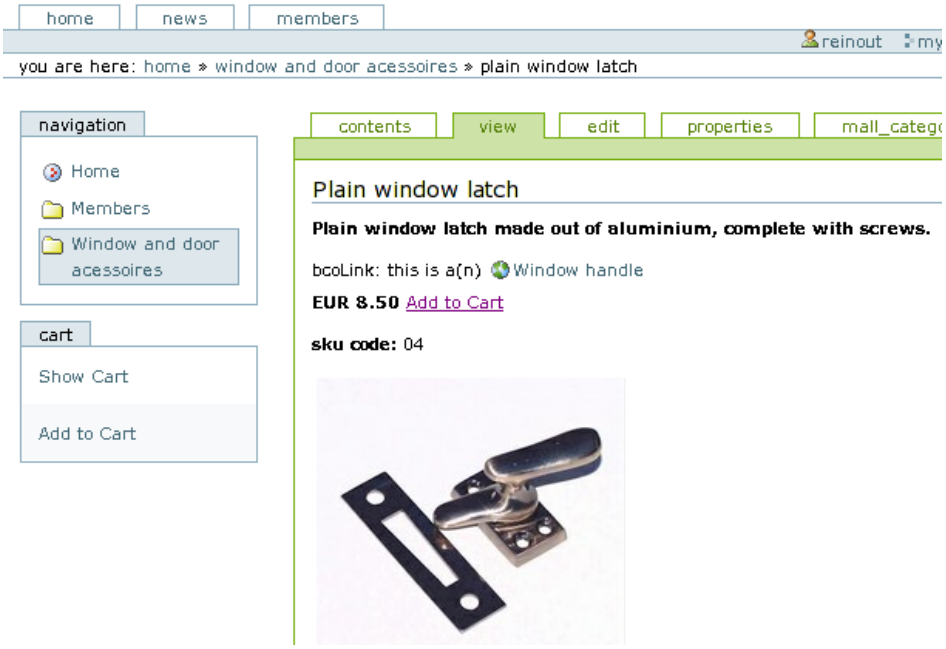


Figure 7.12: *Screenshot of the shopping cart.*

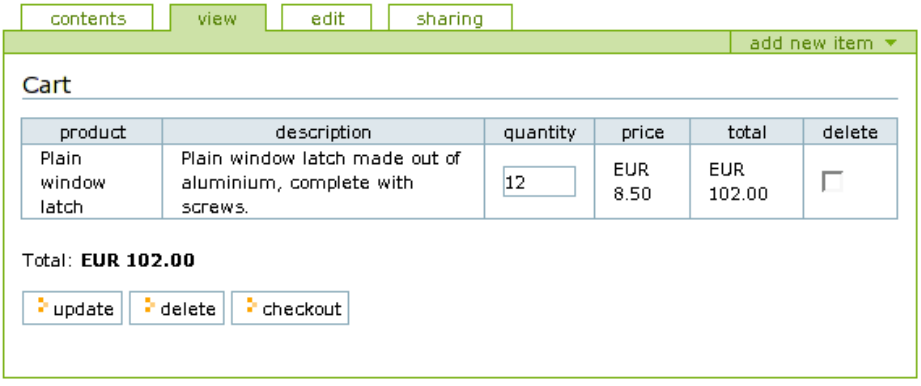
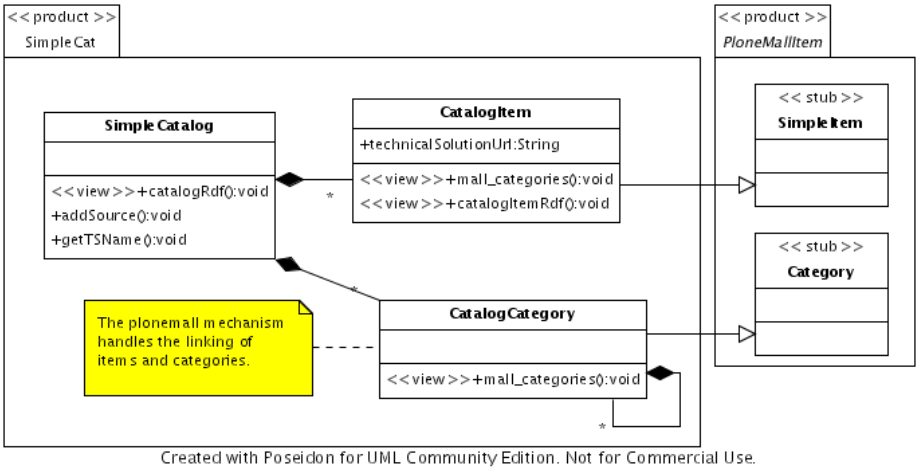


Figure 7.13: *UML diagram for catalog web service example.*



items to bcoWeb Ontology items and for getting the names and properties of the Ontology items.

The connection between catalog items and Ontology items is the RDF/OWL subclass relation. An Ontology can contain a FU 'window latch' which has as a TS 'window handle'. A catalog offering various 'window handles' is not really offering a completely new technical solution, its window handles are more a 'kind of' TS window handle. That is why in this case the subclass relation has been chosen, which offers precisely the 'kind of' Semantics desired. *If* the catalog item really is a new TS, it can be declared as such and used as a separate TS next to the other TSs.

The catalog application can, of course, export its contents as RDF again: figure 7.14 on the facing page.

7.4.7 Tree instantiator

The goal of `ObjectTree` is to demonstrate the use of bcoWeb and web-based data in a project setting. `ObjectTree` allows you to specify a project as parts that consists of subparts and so on. Those parts can be just locally used items or they can be taken from a bcoWeb

Figure 7.14: *RDF example of a catalog item.*

```
<rdf:RDF xml:base="http://simplecat.org/windows/latch04">
  <simplecat:CatalogItem rdf:about=
       "http://simplecat.org/windows/latch04">
    <rdfs:subClassOf rdf:resource=
    "http://en.bcoweb.org/test/englishtest/WindowHandle"
    />
    <rdfs:label>sdfsdfsdf</rdfs:label>
    <simplecat:price>20.0</simplecat:price>
    <simplecat:currency>EUR</simplecat:currency>
    <simplecat:skuCode>04</simplecat:skuCode>
  </simplecat:CatalogItem>
</rdf:RDF>
```

Ontology.

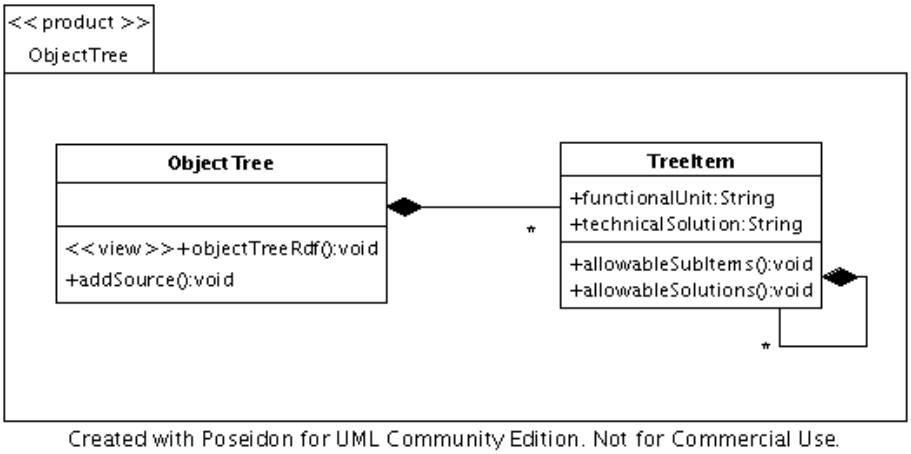
Like the catalog application, the tree instantiator uses the `Rope` RDF library as a back-end. This way, the application has access to bcoWeb Ontologies that can be used to assist in creating the object tree. Figure 7.15 on the next page shows the UML diagram that was used to generate the simple application with ArchGenXML.

It is possible to add TreeItems without using the bcoWeb functionality, in this way an object tree as described in [21] can be made.

To use the bcoWeb-provided Ontology functionality, you have to link a TreeItem to a Functional Unit (FU), signifying that the TreeItem tries to answer that particular functional demand. The RDF library is queried for possible solutions, which can then be chosen. For a chosen Technical Solution (TS), the FUs that the TS consists of are displayed as possible subitems for the object tree. Clicking them adds them as a subitem, see figure 7.16 on page 161.

Using bcoWeb in this manner brings home the interactiveness of the underlying FU/TS model. It is a very natural model for supporting design and specification work. A question is provided along with possible answers, answers in turn lead to further questions. Multiple levels of detail can be covered in this way, without forcing anyone to

Figure 7.15: *UML diagram for object tree instantiator web service example.*



specify every detailed item, as the interaction can be broken off at any level.

ObjectTree makes its data available as an RDF file, so that other applications can always download the up-to-date project information: figure 7.17 on page 162. This possibility is used a few times in this thesis, see figure 7.7 on page 151.

7.5 Conclusions

Creating software in the form of bcoWeb compliant applications and web services results in a new generation of co-operative software tools that, by definition, are able to work as a team and in a team including human partners, and that enforce the re-use of existing information and knowledge to the Client's advantage. Because of the availability of the bcoWeb Semantics, it is possible to precisely describe relevant information and knowledge for re-use. Web services do not necessarily imply human activities and therefore may well be available at lower cost. Also the fact that users (1) only pay for the actual usage and (2) don't need to buy a license, nor (3) need to follow a course, makes

Figure 7.16: *Example of the object tree application. The 'road crossing in the main road' is connected to the bcoWeb FU 'road crossing'. Of the available TSs, 'overpass' has been chosen. The properties you see come from a hand-made RDF file (see figure 7.10 on page 156) that defines those properties and attaches them to the 'overpass' TS. In practice, this property information should come from an Ontology somewhere. Through the bcoWeb connection, the object tree application can show the suggested FU subitems, like 'bridge section' and 'center support'.*

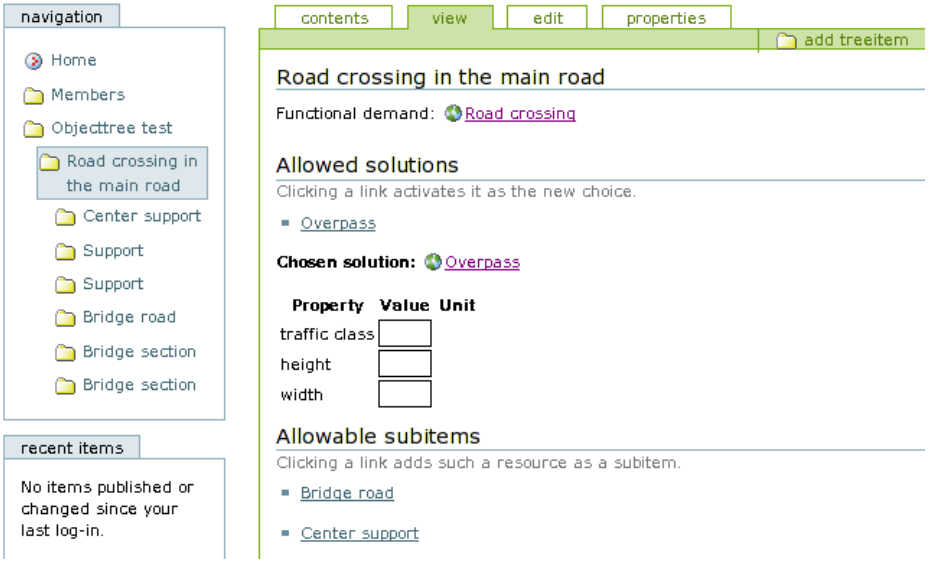


Figure 7.17: *Example of an object tree RDF export.*

```
<ObjectTree rdf:about="http://objecttree.org/OT">
  <node>
    <Node rdf:about="http://objecttree.org/OT/root">
      <rdfs:label>overpass</rdfs:label>
      <functionalUnit>
        http://en.bcoweb.org/bridges/RoadCrossing
      </functionalUnit>
      <technicalSolution>
        http://en.bcoweb.org/bridges/overpass
      </technicalSolution>
      <node>
        <Node rdf:about=
           "http://objecttree.org/OT/root/Support">
          <rdfs:label>Support</rdfs:label>
          <functionalUnit>
            http://en.bcoweb.org/bridges/support
          </functionalUnit>
          <technicalSolution></technicalSolution>
        </Node>
      </node>
      <node>
        <Node rdf:about=
           "http://objecttree.org/OT/root/Support2">
          <rdfs:label>Support</rdfs:label>
          <functionalUnit>
            http://en.bcoweb.org/bridges/support
          </functionalUnit>
          <technicalSolution>
            http://en.bcoweb.org/bridges/abutment
          </technicalSolution>
        </Node>
      </node>
    </Node>
  </node>
</ObjectTree>
```

the approach attractive. Also the fact that the software vendors run the software on their own web servers is interesting, because many problems with software distribution and maintenance disappear.

The initial development of bcoWeb based web services already clearly demonstrates the power of the new information and knowledge sharing paradigm. Because bcoWeb contains the Semantics of the BC industry itself, it is not a difficult task to create a web interface for an existing tool that requires semantic communication and turn it into a web service. This feature will provide an easy growth path from existing tools to a future Ontology-based alternative.

With all the Semantics available it also becomes possible to create applications in the form of web services that can do very smart things, like for example code and regulation conformance checking, common-design-error database searching, constructability rules enforcement and much much more.

Bibliography

[1] Open Source Initiative. Opensource.org website. Available on-line at `http://opensource.org/`.

[2] Python website. Available on-line at `http://python.org/`.

[3] Guido van Rossum. Python status update, January 2006. Article mentioning the heavy use Google makes of python. Available on-line at `http://www.artima.com/weblogs/viewpost.jsp?thread=143947`.

[4] Bill Venners. Programming at python speed, January 2003. Available on-line at `http://www.artima.com/intv/speed.html`.

[5] Paul Graham. Succinctness is power, May 2002. Available on-line at `http://www.paulgraham.com/power.html`.

[6] Zope website. Available on-line at `http://zope.org/`.

[7] Plone internationalisation team, 2005. Available on-line at `http://plone.org/development/teams/i18n/`.

[8] Plone website. Available on-line at http://plone.org/.

[9] Jens W. Klein. ArchGenXML manual - generating archetypes using UML, March 2003. Available on-line at http://plone.org/documentation/archetypes/archgenxml-manual/.

[10] Archetypes documentation, 2004. Available on-line at http://plone.org/documentation/archetypes/.

[11] Reinout van Rees. Archetypes bug patch, January 2004. Available on-line at http://sourceforge.net/tracker/index.php?func=detail&aid=877505&group_id=75272&atid=543432.

[12] Matt Hamilton. Plone for a beginner, June 2004. Available on-line at http://www.under-score.org.uk/pipermail/underscore/2004-June/030148.html.

[13] Paul Graham. *Hackers and painters*, chapter The other road ahead. O'Reilly, May 2004. Chapter also available on-line at http://www.paulgraham.com/road.html, highly recommended for developers.

[14] Adam Bosworth. Evolution in action, October 2004. Available on-line at http://www.adambosworth.net/archives/000028.html.

[15] Reinout van Rees. Subversion repository for bcoWeb editor source code, 2005. Available on-line at https://svn.vanrees.org/websvn/wsvn/opensource/bcoweb/bcowebedit/trunk/, see https://svn.vanrees.org/ for instructions.

[16] Peter Willems. Desktop bcoWeb browser and instantiator. Available on-line at http://www.xs4all.nl/~pettil/bcoWeb/.

[17] John Udell. Nobody expects the spontaneous integration, December 2002. Available on-line at http://www.infoworld.com/articles/op/xml/02/12/19/021219opwebserv.html.

[18] Reinout van Rees and Frits Tolman. Semantic web technologies applied to building specifications. In *Conference proceedings - CIB world building congres Toronto*. CIB, 2004. Available on-line at `http://vanrees.org/research/papers/`.

[19] Reinout van Rees. Subversion repository for Rope (rdflib+zope) source code, 2005. Available on-line at `https://svn.vanrees.org/websvn/wsvn/opensource/Rope/trunk/`, see `https://svn.vanrees.org/` for instructions.

[20] Fabiano Weimar dos Santos. Plone mall e-commerce excellence website, September 2004. PloneMall is an e-commerce framework for Plone 2 CMS, available on-line at `http://plonemall.com/`.

[21] Sander van Nederveen and Frits Tolman. Neutral object tree support for inter-discipline communication in large-scale construction. *itcon*, September 2001. Available on-line at `http://itcon.org/2001/3/`.

Whatever works is good enough. If you can understand what the customer wants, you win. Of course, if your customer doesn't know what he wants, then it is like arguing with your wife: no matter how right you are, you are still sleeping on the couch.

8

Case studies

The goal of this chapter is to verify and demonstrate the solution concept developed in this thesis, and to help the reader to get a better understanding of the theory, the next generation instruments and their possible benefits. Note that section 4.4 on page 65, on eConstruct, can also be considered a case.

The *introduction* places the chapter in the context of the thesis. The case studies involve the development and application of (a part of) an *experimental Building-Construction Ontology Web, called bcoWeb*. Also included is the use of a number of 'smart' computer applications implemented as prototype *web services* that are based on the bcoWeb. Lastly, a *showcase* has been added to evaluate and demonstrate the feasibility of the approach and illustrate the value of the technology for improved information and knowledge sharing, and the realisation of much more dynamic versions of the traditional BC process. *Conclusions* close off the chapter.

8.1 Introduction

Chapter 6 concludes that BC Semantics should be made available online in a form that is both understandable by humans and computer applications. A shared network—or web—of definitions of BC terms (name, property names, units) enriched by relations, constraints, and various expressions of knowledge, as an interface between demand and supply, and between service providers and their clients, seems a good way to improve the information and knowledge flows through the chains, and consequently improve the value adding performance of the BC industry. Based on such a network it seems possible to develop 'smart' applications that are able to act as if they really understand us and our technical communications, and to participate in the various BC processes on an equal basis. Computers are masters in fast and faultless communication of complex data. By elevating the level of Semantics used in their communications to the same level used by us humans, 'smart' computer applications can in the future become professional and reliable co-workers with useful added skills.

Some of the core solution concepts (the use of Internet and XML) were demonstrated by the EU project eConstruct's case, see section 4.4 on page 65. eConstruct's case showed that XML and the Internet are a good fit for BC; that REST web services are handy; that a single big Taxonomy doesn't work that well.

This chapter describes the development of the Building-Construction Ontology Web (bcoWeb). At the core, bcoWeb aims to provide a basis upon which *BC process innovation* can take place. Starting point is the idea that object definitions can be organised in many ways, *i.e.* alphabetic, according to a Classification system or a number of Classification systems, system wise (system, sub system, aspect system), in *is-a* or *part-of* hierarchies, etc., but that a more elaborate Ontology web should aim:

1. To increase the matching process of demand and supply (or offering). Demand, transaction and supply innovation requires a much better insight in each others information and knowledge.

2. To support capturing and re-use of BC related information and

knowledge.

3. To facilitate the development of 'smart' computer applications that behave as if they 'understand' the meaning of the language(s) used in BC.

4. To involve 'smart' computer applications to directly (without human interference) participate in the information and knowledge processing tasks.

8.2 An experimental bcoWeb

This section starts off with a first experiment for creating an OWL-based Ontology, followed by an improved approach using the FU/TS paradigm: bcoWeb. The goal is to see if this works better than the single big Taxonomy from eConstruct's case.

8.2.1 Experimenting OWL: a first effort

A first test for a web-based Ontology was done in co-operation with graduate student Wouter van Vegchel [1]. The aim was to get more familiarity with web-based Ontology technologies, work that was partly started in eConstruct (see section 4.4.2 on page 66). The prototype was implemented to support the owners of private dwelling houses to consider the feasibility of the realisation of an extension of their living space. The idea was to create a tool that can support the on-line evaluation of alternative technical solutions.

A reason to start with an OWL-based Ontology was the wish to experiment multiple separate, cooperating Ontologies instead of one single huge Ontology, as was done in eConstruct. OWL uses the NG Internet's Globally Unique ID (GUID) mechanism described in section 4.2.1 on page 55. With this mechanism, items in various Ontologies can be connected—thereby making it possible to use a divide-and-conquer approach at creating Ontologies (see figure 8.1 on the next page).

Figure 8.1: *Code example of two cooperating Ontologies, analogous to the graphical diagram in figure 6.1 on page 113.*

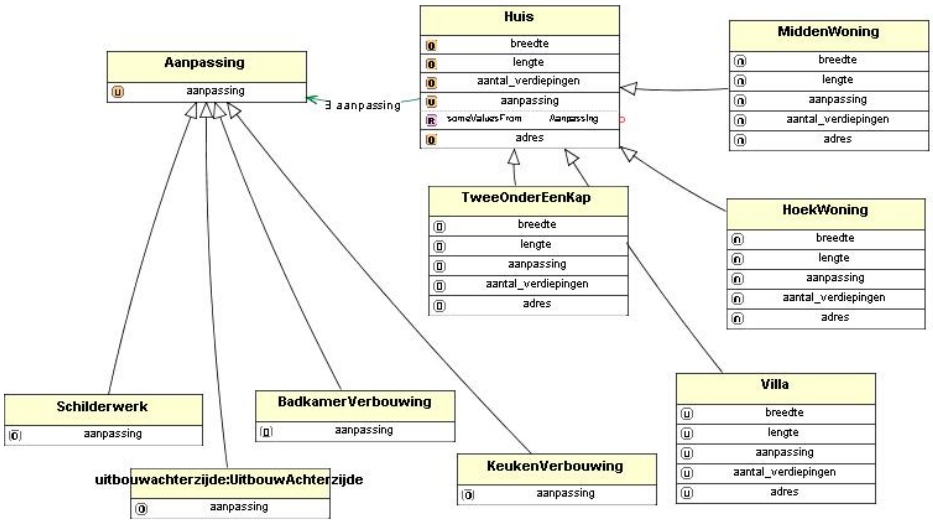
```
...file with base URL http://en.bcoweb.org/housing...
<owl:Class rdf:ID="Door"/>
<owl:Class rdf:ID="Foundation">
  <!-- More information -->
</owl:Class>

...file with base URL http://en.bcoweg.org/bridges...
<owl:Class rdf:ID="Bridge">
  <bcoweb:consists-of rdf:resource=
      "http://en.bcoweb.org/housing#Foundation"/>
</owl:Class>
```

The mechanism was used in this first OWL test, resulting in several small Ontologies of about 10 classes each. The Ontologies could be connected without any problems, validating the OWL approach in that respect.

Protégé, the tool used to create the Ontologies, is a very powerful tool that already exists for a decade. Protégé is extremely powerful as an information modeling tool, though it basically mainly supports the subclass relation as object structuring mechanism. This experiment's implementation added a limited part-of relation, as it was needed. Using Protégé, the main structure of the home extension Ontology consisted of small two-level hierarchies. Each two-level hierarchy focused on a particular end-user choice. The subtypes were mainly technical solutions for a more generic concept. The more generic concept was often used as the target for either a relation or a part-of relation.

Protégé contains a lot of functionality, but it is therefore intimidating for casual users. The Ontologies developed in this first scenario were mostly created with a UML-like plugin for easy graphical editing, like seen in figure 8.2 on the facing page. For large-scale Ontology

Figure 8.2: *Example of the structure of the small Ontologies. Image by Wouter van Vegchel.*



creation with cooperation from BC-expects that are not ICT-experts, a more restricted editor seems more suited. In this respect, web-based editors are normally easier and simpler.

The Ontologies created were ad-hoc. The only objective was to support the selection of alternative technical solutions. The small modeling needs of the case study provided the classes for the Ontologies.

Using the subclass-relation worked fine for structuring similar kinds of objects, like different kinds of houses or different kinds of doors. Those small hierarchies were mostly just one level deep, perhaps two. Extending this small-scale structuring mechanism to the scale of an entire, BC-spanning Ontology appears to be much more difficult, as shown in chapter 3.

8.2.2 Improved approach: bcoWeb

There are several alternatives for structuring Ontologies. The FU/TS paradigm proposed in the GARM [2], see section 6.2.4 on page 113,

seems attractive and useful, but it needs to be tested with a sizable set of data. This also directly draws attention to the part-of relation Semantics. Taking just eConstruct's Ontology as example, it was never clarified whether the parts indicated in the Ontology were a definitive list, just a suggested list of common parts, or a set of obligatory parts. FU/TS includes the decomposition of a technical solution into a few smaller functional units and so could be a good basis for a clarification.

As discussed in section 6.2.4 on page 113, the GARM provides a meta schema approach to information modeling that nicely suites the needs of the BC industry. Though the original model has not yet been applied to Ontology building, the core constructs can easily be used to capture the separation between what is required ('what') and what can be provided for ('how'). This distinction between functional and technical perspective can be used to improve the information needed for and provided by both the Demand side and the Supply/Offering side.

The implementation of the bcoWeb editor is discussed in the previous chapter, section 7.3 on page 140. BcoWeb was created, content-wise, with help of graduate student Noor Hellemans. The original content was all in Dutch. The English content as seen here was made especially for the thesis and carries much less weight in numbers than the Dutch content.

It was felt that creating something simple would be the best course of action [3], likewise something that was dynamic by nature. So a simple version of the GARM was made with just TSs and FUs and their relations. You have a desire for something, you want a function, you want something you can describe the function of, a FU. This can be fulfilled by one or more things that are an answer to your question, a solution for your problem, a TS. Such a technical, physical solution itself normally consists of parts that can be exchanged for other parts. These parts can again be seen as a FU for which you need a TS, starting the sequence afresh. By using the GARM as the basis for this development the road towards greater functionality, if needed in the future, is more or less clear.

A second subject of interest regarding Ontologies is the speciali-

sation hierarchy. This mechanism is very useful, especially for data reduction. From a scientific viewpoint it is interesting to try to combine the specialisation hierarchy with FU/TS decomposition using multiple, smaller, Ontologies as shown in this research.

8.3 Using Ontologies and bcoWeb: web services

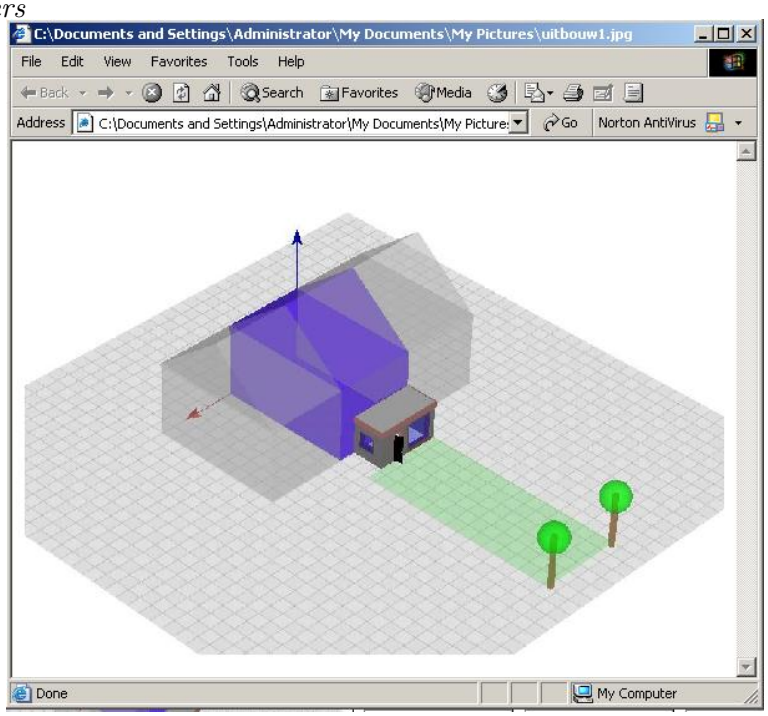
As discussed in chapter 6, the true value of a well-structured web of domain-specific Ontologies for BC is the support for a new mechanism for information and knowledge sharing. If our computer applications are able to behave as if they understand the language we use to communicate among ourselves and between the various stakeholders, BC processes can be much improved. 'Smart' computer applications, including software and hardware (intelligent equipment and robots), that are able to cooperate with us humans like co-workers can provide us with a fast and faultless communication and large-scale re-use of existing information and knowledge.

One of the most important implementations of the new information and knowledge sharing mechanism is the Ontology based Web Service. In the definition of this thesis a Web Service is a recognisable service electronically provided by one or more independent, cooperative computer applications over the Internet. The user sends his data and questions over the Internet to the Web Service and receives (as fast as possible, preferably in real time) the requested answers also over the Internet. The next sections will illustrate the idea.

8.3.1 Experimenting Ontology-based applications

With the concepts (of the first experiment, see section 8.2.1 on page 169) stored in the Ontologies, a prototype web application was made that supported the Client in his awareness and decision process. The main focus of the effort was to use only the Ontologies—as much as possible.

Figure 8.3: *Visualisation of a house with an extension. Image by Hans Schevers*



The starting point, a one-family house, is fixed. But the application retrieves the available subclasses of 'House' and presents them as choices to the user. The selected class is then polled for attributes, which are presented to the user (flat or tilted roof, for instance). The design can be used as the basis for a visualisation following the research of eConstruct [4] where VR shapes were generated as mark-ups from bcXML content files (see figure 8.3).

As a test case some of the applicable rules in the building regulations have been added to the Ontology. As these rules are simply of an if-then-else type OWL has no problem with them. In fact it seems that OWL Description Logics (DL) and OWL Full provide a strong basis for this type of applications and opens up a whole new market for knowledge vendors (including R&D institutions). Though time became rather limited we looked into the possibility to check the de-

signs against a knowledge base that contained knowledge to prevent construction errors. From what we did, it seemed that this type of services can be nicely build on-top of the Ontology network. As many other services spring to mind from costs, risks, to a multitude of analyses, the idea took shape that this might become a whole new way to market knowledge and software.

8.3.2 Specification integration

With the instance available, a coupling with the Dutch STABU Specification system has been made. A problem is that the Dutch Specification system does not really support such small projects executed without an architect (it is both an implementation and a legal problem: there are two differing legal frameworks, one for small works like rebuilding and one for bigger works with architects, sub-contractors). The work done is therefore just for test purposes.

An interesting addition to this process was the conversion of the Dutch SfB Classification table (nl-SfB, *elementenmethode*) to a similar chapter structure as used in the Specification and adding the links from these chapters to the Specification items. Without changing anything in the original data, this second chapter structure could be combined with the resulting Specification items and transformed to HTML as an alternative building Specification.

To be more accurate, the first Specification structure is a work breakdown structure, which is the structure of the Dutch building Specifications. The second Classification (nl-SfB) is normally used (in the Netherlands) for costing applications and Computer Aided Design (CAD) drawing layering. As a side note, work is underway at STABU to make the nl-SfB an alternative for the work breakdown Classification in the building Specifications.

Concluding: storing the base data in Ontologies worked well. Both the support application and the Specification generator could use it as a basis for communication. The possibility of subclassing (inheritance) was used well by both applications. The Client support application displayed different kinds of houses (subclasses of 'House'), the Specification generator needed a 'House' instance as a starting point,

but reacted also perfectly to an instance of a subclass. This might not seem like a big deal, but there are not many current applications that have such a thing build-in.

The real support of the Client by warning him for common pitfalls was not undertaken because of time constraints. In this way, only the generation of a simple Specification as a basis for a contract provides a bit of support. On the other hand, the Ontologies contained common solutions for house extensions, preventing possible omissions.

Regarding further work on the instance model, the next stage is obvious: a real interaction with CAD-based data. The only currently realistic option is IFC. When looking at a semantic web supported scenario, it makes sense to use ifcXML. As a baseline, ifcXML files should be downloadable on the Internet, but it makes a lot of sense to expose the normal IFC data store functionality on the Internet. `http://example.com/building42/2nd-floor` could return an ifcXML file with all elements on the second floor, `http://example.com/building42/doors` could return all doors in the model. See section 4.2.1 on page 56 for more details.

8.3.3 Involving supply: integrating catalog and Ontology in a project

Figure 8.5 on page 178 shows a small case where a catalog and a bcoWeb Ontology are integrated with a project's object tree. An overview of the three files at the basis of the integration is given by figure 8.4 on the facing page.

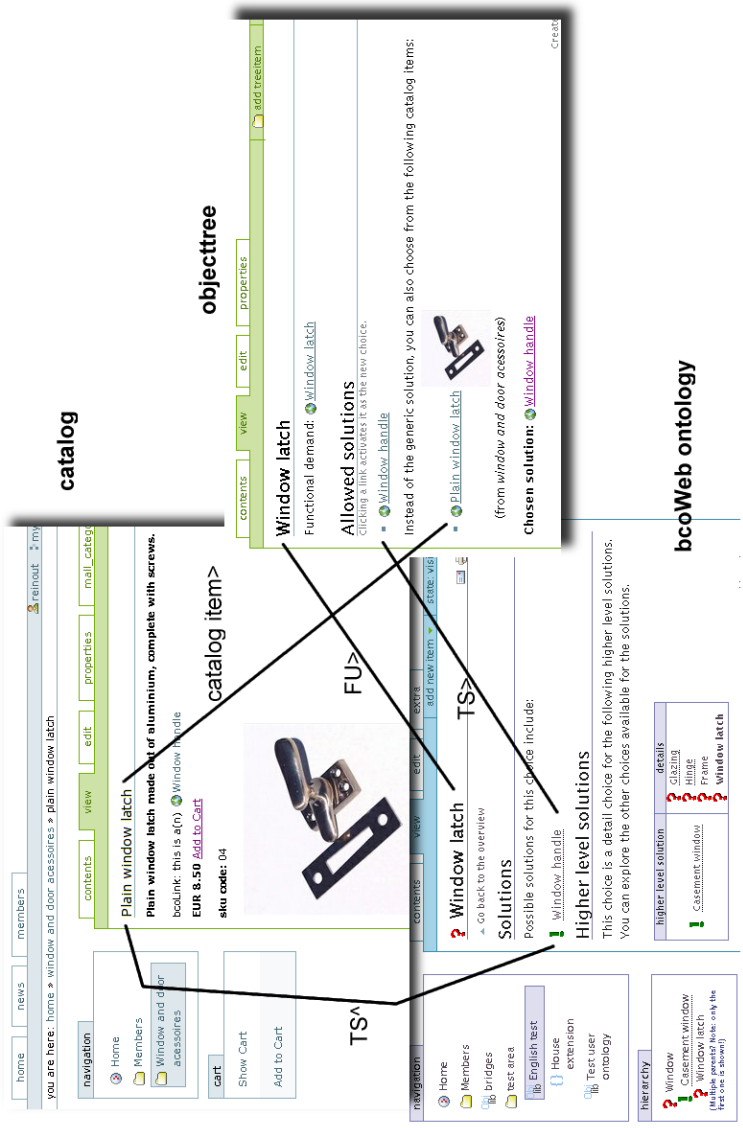
In the Ontology, the FU 'window latch' has a 'window handle' as a possible TS. In the catalog, an existing window handle uses the bcoWeb Ontology by connecting to the bcoWeb window handle. In the background, this is modeled as an inheritance relation.

The object tree application, at a certain level in the project, adds a new tree item fed from bcoWeb: a request for a window latch. Normally, the object tree application would only show the bcoWeb-supplied TSs, but when it has loaded the web-readable data from several catalogs, it will also display objects that are subclassed from the valid TSs.

Figure 8.4: *Technical detail: extracts from the three RDF files underlying figure 8.5 on the following page.*

```
<!-- bcoWeb content -->
<FunctionalUnit rdf:about
  ="http://en.bcoweb.org/test/englishtest/windowlatch">
  <rdfs:label>Window latch</rdfs:label>
</FunctionalUnit>
<TechnicalSolution rdf:about=
  "http://en.bcoweb.org/test/englishtest/WindowHandle">
  <rdfs:label>Window handle</rdfs:label>
  <fulfills rdf:resource
  ="http://en.bcoweb.org/test/englishtest/windowlatch"/>
</TechnicalSolution>

<!-- catalog content -->
<simplecat:CatalogItem rdf:about
   ="http://simplecat.org/windows/latch04">
  <rdfs:subClassOf rdf:resource=
   "http://en.bcoweb.org/test/englishtest/WindowHandle"
   />
  <rdfs:label>Plain window latch</rdfs:label>
  <simplecat:price>8.50</simplecat:price>
  <simplecat:currency>EUR</simplecat:currency>
</simplecat:CatalogItem>

<!-- object tree content -->
<Node rdf:about=
    "http://objecttree.org/OT/window/Windowlatch">
  <rdfs:label>Window latch</rdfs:label>
  <functionalUnit>
  http://en.bcoweb.org/test/englishtest/windowlatch
  </functionalUnit>
  <technicalSolution>
  http://simplecat.org/windows/latch04
  </technicalSolution>
</Node>
```

Figure 8.5: *Integration of bcoWeb results and catalog results in an object tree application.*



This serves as a good demonstration of the relative ease by which several information sources can be combined when you use semantic web technologies.

8.4 Showcase: demonstrating the feasibility of the concept

This section presents a showcase that demonstrates the feasibility of the bcoWeb and illustrates the reasoning behind the most important conclusions drawn in chapter 6: building a bcoWeb improves BC's value adding performance; national differences matter, so it should be done (at least *start*) on a national, not international, scale[1]; BC web services are essential in getting BC's building process innovation going.

8.4.1 Stakeholders and their views

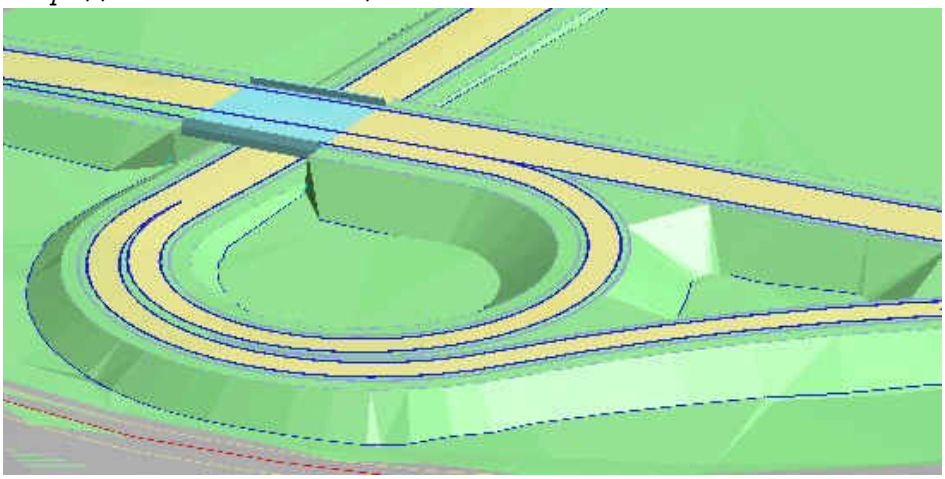
Some of the most important stakeholders in a typical overpass project are governmental bodies, authorities, the general public, main contractors and subcontractors, road designers, structural engineers, cost calculators, knowledge providers, etcetera. The case will show that many of them can join into the project in an earlier phase and in a more flexible and economically attractive way than currently feasible.

Each stakeholder has its own view on the construction project. The general public is not interested in most of the purely technical data. People living in the area are more interested in the level of noise to be endured during construction, or the traffic flow to and from their homes or work, etc. Figure 8.6 on the next page shows imaginary integration of semantic data with google maps ('imaginary' meaning that it is example data; this kind of data coupling *is* made routinely with google maps, see [5] for an example).

[1]Many European projects in the past tried to develop EU-standards from scratch (without national standards). All these projects failed, they could not agree on common definitions because there are no common definitions.

Figure 8.6: *General public oriented GIS interface based on google maps (`http://maps.google.com/`). Just an example based on current possibilities.*

Figure 8.7: *MOSS editing session integrated in MicroStation. Image from* `http://www.state.me.us/`.



An interest group's knowledge database accesses the data of an environmental planning group that provides them with a continuously updated GIS view. This view is transformed into a 4D view of the project and a good sound level estimate based on, for instance, the drilling method used for the foundation piles. This is a completely different view from the CAD view of the designer.

In order to make these views more effective, the environmental planner's GIS data should also be brought on a higher (semantic) level including objects like 'building', 'neighborhood', 'parcel', 'greenery'. These object definitions should be entered in the bcoWeb.

In the opening stage of the design and decision-making process the project road designers, for example using MOSS (figure 8.7), produce a gradual refinement from a broad concept to a more detailed design. Ontology data concerns for example objects like 'road radius', 'road section', 'gradient'.

Contractors and subcontractors retrieve a description of the project in terms of bcoWeb Ontology data. Their tools are able to understand that input without any problem and are immediately ready to do the calculation required. Among themselves and their partners they communicate using objects like 'contract', 'budget',

'bulldozer', 'time-slot'. Definitions of these objects are again collected in a separate part of the bcoWeb.

8.4.2 Cost

Especially important for this case is the ability to plug in cost knowledge, as infrastructural projects often have a sizable political and budgetary component. Ideally, a reasonably accurate cost estimation has to be available in every stage. The estimate will be less accurate in the early stages and more accurate in the final design stage, but the level of accuracy can itself be provided along with the financial figures.

8.4.3 An imaginary overpass project

The case[2] will be described in a chronological order. Figure 8.9 on page 184 shows the basic structure of a bcoWeb Ontology for overpasses (and potentially other road crossings and bridge-like structures). Figure 8.10 on page 185 shows the Dutch and English content in the bcoWeb interface.

It is interesting to note several national differences here. When traveling abroad by car it is apparent there are differences in road construction, like: highway exits, crossroads construction. Britain's roundabouts (see figure 8.8 on the next page). The Netherlands have roads with a pile foundation in certain areas of the country to keep the road from sinking away in the mud. You won't find American-style wooden tressle railway bridges in Europe. In Indonesia houses are constructed with bamboo. Norway's triple glazing insulation won't have much use in hot climates. Differences stemming for example from national differences in soil, elevation and curvature, climate, legislation and culture should all be taken as a fact.

Differences like above make it infeasible to create one single Ontology. If certain parts *can* be shared, it will be in a highly focused area. In highly international industries like oil drilling, there is of

[2]Some data used for the case was graciously provided by the *Bouwdienst*, the engineering service of the Dutch department of Road and Waterworks.

Figure 8.8: *Swindon, UK, has what they call a 'magic roundabout' with both clockwise and counter-clockwise rotation. Image from* `http://p.vtourist.com`. *See* `http://tinyurl.com/yerfon` *for a google satellite view.*

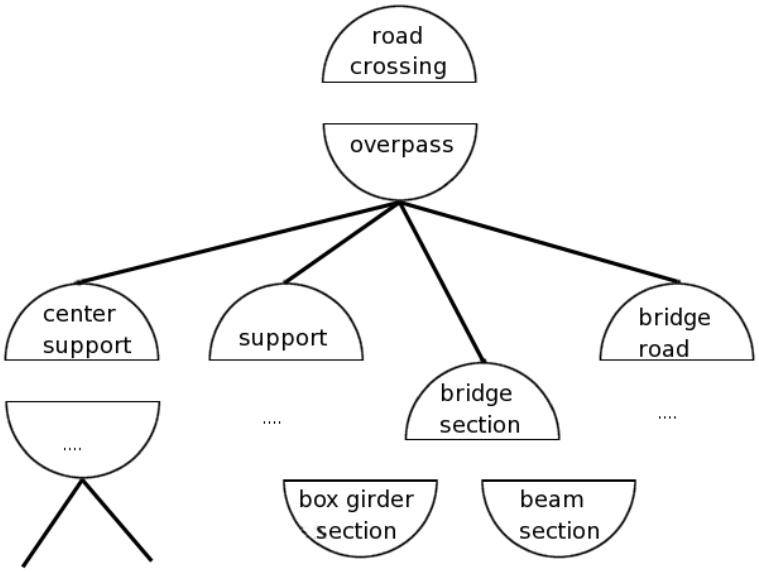
course a much better chance of internationally feasible Ontologies. Note though that it are exactly industries like this that already have existing (STEP-based) solutions, see section 3.3.1 on page 36.

8.4.4 Starting the process

Assumed is a political or technical wish for a new or improved road crossing. In the future BC process, bcoWeb can start assisting the process already in this stage by providing input to a decision support system. BcoWeb is organised around a demand-supply distinction or, phrased differently, a question-answer distinction. The demand for a *road crossing* is coupled with possible solutions like *overpass* or *level crossing*. A bcoWeb interface like in figure 8.10 on page 185 is more of an internal look in the underlying data, the intention is for the data to be used in existing or new end-user applications.

At this point in the process, an initial cost estimation is desired

Figure 8.9: *FU/TS hamburger diagram bridge example, showing the FUs as the upper halves and the TSs as the lower halves*



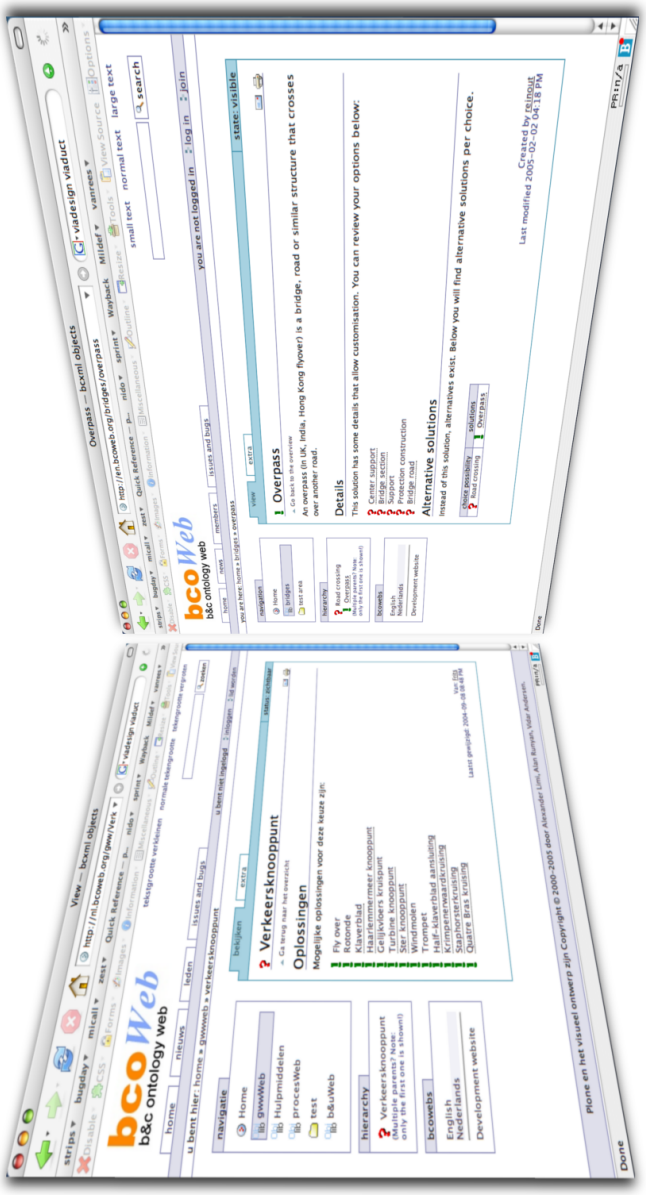
to provide feedback to the designer and the (political) Client. The only real data available is the wish for an overpass (the assumed desired solution) with some basic traffic engineering data like the needed number of traffic lanes and some basic situational data like the length of the span required to cross the lower road.

As described in chapter 4, web services allow an elegant way of handling this. Via the web, your application can connect with an intelligent automatic service provider elsewhere—gaining access to specialised knowledge. Instead of building detailed cost knowledge into each and every application, one single cost knowledge web service can provide it in one go for every application.

Figure 8.11 on page 189 shows how this process could play out. Spreadsheets are a common way to slowly gather, create and build up knowledge of a data-oriented nature. The figure shows a spreadsheet such as created by the *bouwdienst* for gathering information on cost estimations for early-stage overpass projects.

This generated valuable knowledge can then, by an IT expert,

Figure 8.10: *Dutch and English content: different. There is a need for Ontologies on a national scale.*



be turned into a web service—increasing knowledge integration, as the knowledge can now be used by a number of other projects, also outside the *bouwdienst*. The knowledge transfer that can be achieved with printing booklets or writing papers is quite limited. With such a web service, however, the knowledge can be applied directly and broadly.

The web service filters out the correct properties for the overpass and calculates—based on the spreadsheet—the price. This price can then be shown in, for instance, the object tree application. This is not the only option, as the web service can also be used by the financial department to get a running tally of the projects currently in the opening phase.

This is a web service of the kind that you see in many on-line examples: it receives information, calculates something and returns the value. Currently, the *bouwdienst* is creating spreadsheets to capture cost and estimation information, but without a system to tie it to. In the envisioned future it is normal to have and to make such data available over the Internet. Knowledge management applications assist in structuring information and in gathering available information. This can already help within a single company or government department, but it really shines when connecting companies with knowledge institutes (like TNO, CSTB, CSIRO), suppliers, legislators and so on.

How? In the envisioned future, it is as natural for cost estimation data to be available to your CAD-package as it is now to Google [6]. As a provider of cost knowledge, the easier your customers can use your data, the better for your business. Current practice is to deliver a stand-alone application to your customers in which they have to re-enter all their project data. Double work. In the future scenario, you can put your cost information on-line. Of course with password protection, as your service has real value to your customers—meaning paying customers in this case.

As a knowledge provider, you have a better and easier way of serving the customers. The provider of, for instance, the project management application also has an advantage. He can integrate your (and others') downloadable cost knowledge directly into his project management application, making it a more valuable tool for his cus-

tomers. A win-win situation by reducing the friction in the market and thereby increasing the market's size.

In the overpass scenario, the first cost estimation will likely come from the internal knowledge from the *bouwdienst* itself. Once you go deeper into the technical details, generic BC cost estimation services take over.

8.4.5 More parties joining in

With the choice for an overpass made, the next phase of the project starts. On the design level, choices must be made for the next level of detail, where bcoWeb again can lend a hand. We've got the *question/demand* for an improved road crossing; we've made the choice for an *answer/supply* of an overpass; next come the the subparts, which are *questions* again. An overpass consists of a foundation, supports, and so on. So we *want* a foundation, we have a *demand* for a foundation.

Important for building process innovation is that bcoWeb allows more parties to join in earlier. Instead of throwing a bundle with a fixed design over the constructor's wall, the constructor can tie in earlier in the process. BcoWeb is distinctly less strictly sequential than the traditional process.

Switching back and forth from demand to supply back to demand is a simple, elegant solution. It is also easy to follow. And every time, more participants can join in the project. BcoWeb provides, for the solution 'overpass', the various subparts as additional *questions* that need to be answered. There is a constant interaction from demand to supply to demand and back again. It is beneficial for the project to include the possibilities (and prices) offered by suppliers further down the line now that that has become a possibility.

Likewise, now that the lower levels are accessible through bcoWeb, information provided by knowledge institutes or internal best-practice databases can already be used to effect on the higher levels. A recent research's results, added to the internal knowledge database, might show a better chance for project success for underpasses compared to overpasses due to being less intrusive in the landscape. An underpass

solution, ruled out initially because of cost reasons, might thus receive attention anyway.

There is a need for new kinds of contracts and forms of co-operation to enable this kind of integration. A bcoWeb-like approach provides the technical possibility; the legal and social framework must be in place, too, to reap the full benefits.

Contrast bcoWeb's enabling of participation with normal information structuring solutions like classifications. In classifications, 'static' data is ordered; bcoWeb's structuring mechanism is much more actively involved in the process, it is more 'dynamic' in nature. Instead of a one-hierarchy-fits-everything approach, bcoWeb allows for differences in detail level and an incremental build-up of project data.

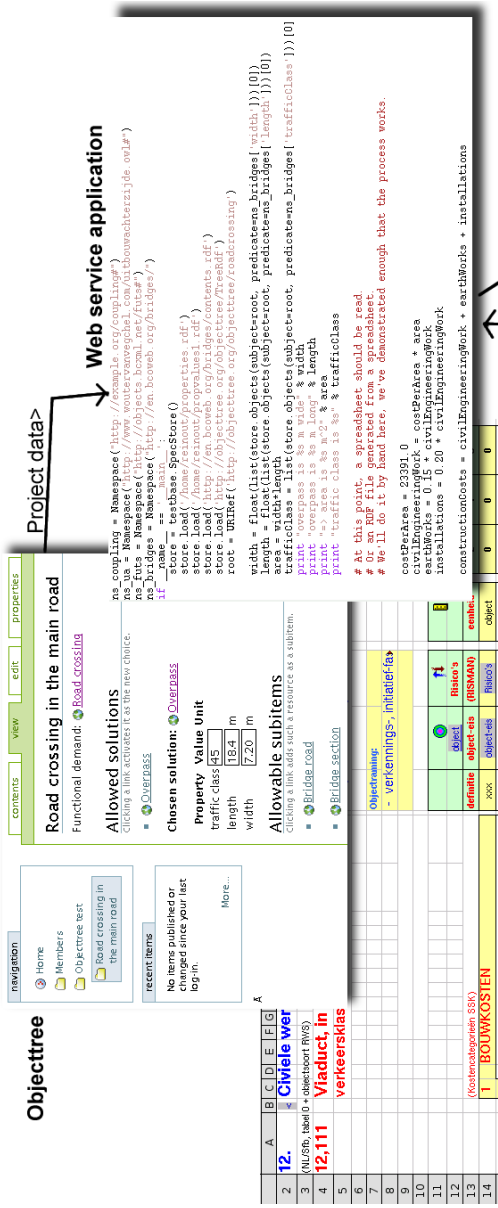
Another way of stating the difference between the current classification approach and bcoWeb: classifications order documents or CAD-layers, but they don't support the whole process. 'Support' is intended in its literal sense, here. The intention of classifications is to order information—for instance CAD drawings into layers of about the same size and importance—in order to keep the drawing manageable[3]. This subdivision, however, has nothing to do with actively supporting and improving the whole BC process itself.

For the overpass project, iterating between supply/demand or question/answer eventually yields a tree-like information structure. This can be supported by integrated project management applications or by, for instance, a focused web application. Behind the scenes, the bcoWeb data is available to both by means of a simple data download.

An object tree application like in figure 8.11 on the next page can be used for a further detailing of the initial 'overpass' choice. Iteratively, you can build up a tree view of your project. You are totally free to create your own subdivision, but bcoWeb contains the knowledge of common subparts for solutions. So when you tie, for instance, the toplevel 'overpass' object to the bcoWeb overpass, you have bcoWeb's knowledge of common subitems at your disposal. This is company-internal knowledge or you can use a common database. Knowledge institutes offer attractive and well-designed libraries of

[3]Note that ordering information in CAD drawings is useful, so classifications are useful. They are not a panacea for all woes, however.

Figure 8.11: *Web service that calculates the preliminary price for an overpass based on a spreadsheet with cost data and an object tree with project data. This example shows (cost) knowledge integration.*



common solutions. Also, Contractors can create small libraries with their innovative solutions—leading to improved supply integration.

As individual smaller bcoWeb Ontologies can so easily be integrated into the whole, a likely source of those Ontologies might become clear: cooperation within a single sector of the industry. (In Dutch: *brancheverenigingen*). Within a single sector, there will be a single focus and a mostly consistent agreed-upon vocabulary that makes creating a bcoWeb Ontology feasible. It gives the sector a good opportunity to tie in to the whole structure and to provide added value to their Clients.

In the application, the suggested or possible subparts for the overpass are shown, allowing the design expert to make his choices (or to allow subparts of his own). Since the data is computer-readable, all sorts of decision or design support systems are possible. For a subpart 'foundation', multiple solutions are shown like 'surface foundation' or a 'depth foundation'.

The depth foundation solution, in turn, consists of multiple subparts for which a design decision must be made: the foundation support and the piles. The piles are illustrative for the strength of bcoWeb's emphasis on computer-readable data. There are numerous kinds of piles with different strengths and weaknesses. These strong and weak points can be captured in a decision support system to be coupled with the design tool. Piles that are screwed instead of (noisily) driven can make a certain design much more politically acceptable as the irritation caused by the construction work's sound can be lowered significantly. See image 8.12 on the facing page.

Essential to grasp here is that the general public's input can be used (or enforced) much earlier in the process. A good designer will keep the public's wishes in mind at all times; the new process makes it more explicit, makes it more quantifiable. Of course, not all the general public's wishes can or should be granted. Perhaps a bcoWeb system can make the trade-offs as a whole more explicit, improving the decision making process's transparency and acceptability.

Figure 8.12: *Improved inclusion of stakeholders: from project overview website to project data to nearby-living citizens meeting to screwed pile foundation (Pile image from* `http://www.joostdevree.nl/`*, meeting image from* `http://www.mierlo-hout.nl/`*).*

Figure 8.13: *Construction of an overpass with a picture of the initial design (A30, the Netherlands). Photo by Laurens Jansen.*

8.4.6 Specifications

Apart from the design and the technical data, there is also contractual data, legal data, organisational data. This is mostly associated with the Specification. In the future, a Specification takes on a much more active role. A 'smart' specification application continually collects project data, keeping up with the various changes and readily produces a traditional specification on the fly. Increasing the height of a building from four to five stories in the design drawings? The Specification application knows that that means that you have the legal obligation (at least in the Netherlands) to add an elevator.

The big 'smart' Specification application's advantages are its flexibility and the fact that it is a real semantic Specification: it has knowledge about Specifications, it is not just a paper representation. Specifications used to be mostly paper-based, making them expensive and hard to make, so this was only done at the end of the design phase. Change is no longer expensive and no longer so error-prone, now a bcoWeb compliant 'smart' system means that the data is well-structured. A bcoWeb Specification ensures that the right in-

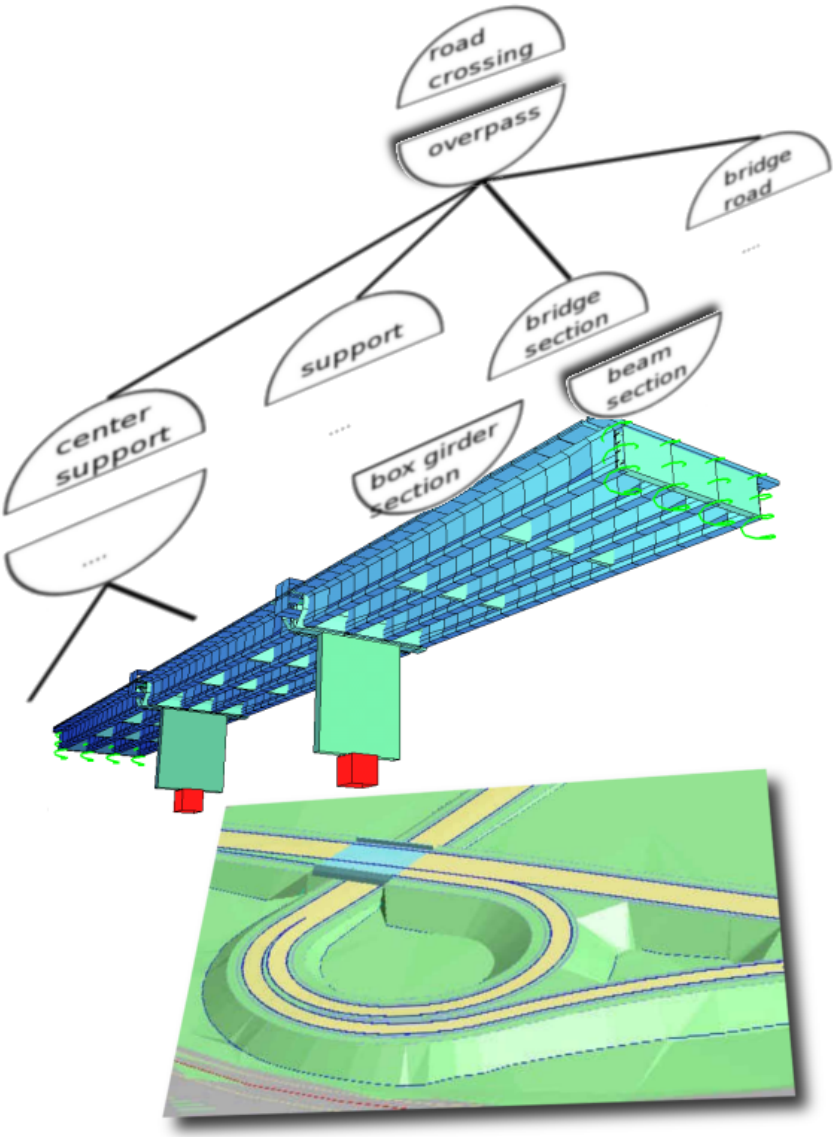
formation is available: for everyone, always, everywhere. It is easy to mis-lay a section or two of Specification text when copy/pasting inside a Word document, but automatic Specification checkers will not give you a chance to err in bcoWeb. Just like changing some walls in a CAD package is easy and cheap, just so is changing a bcoWeb Specification. Also the fact that the 'smart' Specification application is able to directly access the CAD data undoubtedly will result in substantial reduction of cost of failure.

Flexibility—non-rigidity—also means you can slowly build up the info to be contained in it. Early on in the overpass construction, order-of-magnitude contractual data are added. Perhaps one should say 'initial ideas on the type of contract' instead of 'contractual data', but that is something to be decided by the project itself. If, bcoWeb-desired, the Specification is available on-line in this early stage, potentially interested Contractors can pre-select projects on the contractual details in the Specification.

The further the overpass project evolves, the more details can be added to the Specification (figure 8.14 on the next page). The big advantage is that these details can be added the moment the decision is made. So the actual designer gets to make the decision whether a certain quality class is needed—not the maker of the paper-based Specification, traditionally only made right at the end of a project. The drawing-related parts are made with the designer of that part at least close at hand, but in the old paper-based process it is hard to reliably store that temporary knowledge reliably. This either means that duplication of effort (through thought work being re-done by the Specification specialist) or a sub-optimal Specification (for instance demands for quality being set too high or too low). Duplication of effort is costly; likewise work that is delivered to a quality standard that is too high or too low.

What follows from all the above is that the new-style Specification and its creator (a 'smart' Specification application) has a more central place than paper-based Specifications. A paper-based Specification derives its central place from the legal and contractual weight accorded to it. A bcoWeb Specification has, probably, the same legal and contractual weight, but is in addition a valuable source of

Figure 8.14: *Beam section choice for an overpass, integration of data of two different design software packages.*



well-accessible information.

8.4.7 Closing remarks

The case shows the advantages of bcoWeb for BC, the way in which it strengthens BC's value adding performance and attractiveness. bcoWeb allows many stakeholders to get involved earlier on in the process. Many currently static processes and documents, like Specifications, can take on a much more dynamic and active role in the whole process. 'Smart' applications, like 'smart' Specification applications, will be able to access data, information and knowledge provided by others.

The build-in focus on demand and supply allows various problems in BC's process to be tackled. Demand and supply between the Client and the Contractor or the Client and the designers, between the Contractor and the subcontractors and the suppliers. Also with regard to knowledge, either knowledge from internal departments accumulated from earlier projects, knowledge institutes and subcontractors or knowledge provided by the stored data in the project database or the Specification.

bcoWeb's use of 'smart' applications and web services allows available knowledge to be used more broadly, making knowledge more sellable and therefore more valuable and more attractive to generate—which is good for innovation. When simple web services, documents and programs can access each other's data, improved cooperation and effectiveness will be the result. Data integration currently costs money, making the whole *less* than the sum of the parts. With 'smart' applications and web services, the costs are lowered and new opportunities open up.

8.5 Conclusions

The bcoWeb contains open source reference definitions of BC terms (called objects) that are ordered with the intention to support decision making in a hierarchical decision process.

Figure 8.15: *Brand new overpass over the N99 at Westerland, the Netherlands. Photo by Jan-Simon Hoogschagen (from* `http://www.pagowirense.nl/`*).*

bcoWeb supports breaking down complex decisions (what type of power plant do we use) into simpler decisions (do we need a one or two story turbine building) and again in simpler decisions (do we use a concrete or steel frame) and even simpler decisions (what type of wall covering is desired). All the way down to the materials, connections and finishes. BcoWeb allows this on the Ontology level.

Object definitions are grouped in a natural way, *i.e.* the groupings map one to one to the languages of the various disciplines/stakeholders.

Looking at the current bcoWeb filling and the applications surrounding it, it is clear that the idea to build hierarchies of definitions of technical solutions (or function performers) provides BC with a new mechanism for electronic information and knowledge sharing in the various chains that has a number of advantages over earlier mainly paper-based mechanisms.

FU/TS hierarchies ensure that the definition of an object is not

separate from its context and from its actual usage.

- Computer applications can use the bcoWeb to share a common view on the project.
- bcoWeb provides the flexible, commonly accessible BC language required for the development of 'smart' computer applications that are able to behave as if they really understand the language(s) of Building-Construction.
- Smart computer applications (in the future including intelligent manipulators, equipment and construction robots) are able, like human co-workers, to complete tasks in close co-operation with other co-workers and other smart computer applications without human interference—though not without control (!!)—thus providing a promising new technology that seems able to drastically improve current BC processes.
- Providing Internet-based access to the data that underlie the textual document helps to integrate the documents in a greater whole. Catalogs can be used in object trees, an object tree's project data can be used in a Specification.
- Web services are a valid way to consume and to provide information.

The chapter shows that the open source, web-based, bcoWeb-structured, smart applications and web services-using approach can indeed be implemented and that it provides a new opportunity for the BC industry and society at large to increase the speed, controllability and quality of its processes and that it can help BC to become more dynamic, flexible, open, competitive, efficient and effective.

Bibliography

[1] Reinout van Rees, Wouter van Vegchel, and Frits Tolman. Practical use of the semantic web: lessons learned and opportunities

found. In Attila Dikbaş and Raimar Scherer, editors, *eWork and eBusiness in architecture, engineering and construction*, pages 329–336. ECPPM, Balkema, September 2004.

[2] Wim Gielingh. General AEC reference model (GARM) an aid for the integration of application specific product definition models. In Per Christiansson, editor, *Conceptual modelling of buildings*, W74+W78 workshop, Lund, Sweden, October 1988. CIB. Available on-line at `http://itc.scix.net/data/works/robots/w78-1988-165.htm`.

[3] Tim Bray. Technology prediction success matrix–12: 80/20 point, January 2004. Available on-line at `http://www.tbray.org/ongoing/When/200x/2004/01/14/TPSM-8020`.

[4] Frits Tolman, Michel Böhms, Celson Lima, Reinout van Rees, Joost Fleuren, and Jeff Stephens. eConstruct: expectations, solutions and results. *ITcon*, Special issue on European projects, 2002. Available on-line at `http://itcon.org/2001/13`.

[5] Adrian Holovaty. Chicagocrime.org, a freely browsable database of crimes reported in chicago, 2006. A combination of police databases and google maps. Available on-line at `http://www.chicagocrime.org/`.

[6] Tom Sweeney. I live in a google world but I work in a pre-google one, July 2005. AECbytes Viewpoint number 16, available at `http://www.aecbytes.com/viewpoint/issue_16.htm`.

The most absolute lie is the presentation of an irrefutable fact based on unassailable numbers.

L.E. Modesitt jr., Gravity dreams

9 Conclusions and recommendations

The goal of this chapter is to present the final conclusions of the thesis and to formulate a number of recommendations for the BC industry, the software vendors and the Clients of the BC industry.

The *introduction* recapitulates the research questions revisited in chapter 5. The sections following the introduction answer each of the research questions in turn. In the last three sections, the *contributions of this research*, *suggestions for further research* and *recommendations* are presented.

9.1 Introduction

The research questions targeted in this PhD research, as revisited in chapter 5, were:

1. How can state-of-the-art web technology possibly help to realise new, dynamic BC processes that adequately serve the Client's needs (provide value for money) and, at the same time, serve the needs of the construction companies (provide money for value)?

2. How to use the NG Internet to provide both parties (and their application software) with the precise and Semantically rich online BC object definitions?

3. How can software vendors take advantage of such a development?

4. What is the possible role of the traditional Specification in the newly innovated BC processes?

In the next sections these questions will be answered in detail.

9.2 How can web technology support dynamic processes?

As discussed in chapter 2, BC process innovation should focus on a substantial increase of the information and knowledge sharing capacity of the industry at large. Unlike industries like automotive and shipbuilding, BC projects are usually temporary joint ventures of companies that realise a unique project in a unique Virtual Enterprise. Information and knowledge developed in a project is not stored for re-use in later projects. Improvisation is the trick of the trade.

Starting point is the idea that definitions of terms of the BC industry can be entered into the computer as reference definitions that can be used by humans and computer programs as a kind of BC Esperanto. *i.e.* a common language understood by humans and computers alike. Smart computer applications are able, like human co-workers, to complete tasks in close co-operation with other co-workers and other smart computer applications without human interference—though not without human control.

Reference object definitions can be organised in many ways, *i.e.* alphabetic, according to a Classification, system wise, etc., but a more elaborate Building-Construction Ontology Web (bcoWeb) should aim to:

1. Increase the matching process of demand and supply (or offering). Demand, transaction and supply innovation requires a much better insight in each others information and knowledge.

2. Support capturing and re-use of BC related information and knowledge.

3. Involve computer applications and smart construction documents to directly participate in the information and knowledge processing tasks.

9.3 How to use NG Internet technology?

This subject has been discussed in chapter 4 and chapter 6, the main conclusions have been the following:

1. Development of an open, on-line available, human and computer accessible store of BC information and knowledge; this can be created by providing a common and open source Building-Construction Ontology Web (bcoWeb).

2. To integrate documents (like Specifications) and applications into the Internet-based information and knowledge exchange, they should be made available and interactable using simple HTTP+XML web services.

3. The NG Internet linking and Ontology possibilities should be used to improve the knowledge utilisation in BC, improving the value adding capacity.

The case in chapter 8 showed that bcoWeb provides the flexible, commonly accessible BC language required for the development of 'smart' computer applications that are able to behave as if they really understand the language(s) of BC.

9.4 How should the software vendors react?

Software vendors are key players in the realisation process of the ideas presented in this thesis. Without software that conforms to the bcoWeb proposed, nothing much will happen. The question is then: 'Why should software vendors invest into adapting their existing tools, and into the development of next generation bcoWeb compliant tools?'

The answer is, besides the obvious but naive observation that BC software vendors should try to improve the value adding performance of the BC industry, that it is also very good for the BC software market. There are a number of reasons why BC software vendors should adopt bcoWeb.

Open source The bcoWeb data is open source[1], so it provides a safe, non-proprietary platform to build upon: software vendors will not be dependent on one single organisation. Second, it means that bcoWeb can be extended and changed; additions can be given back to bcoWeb, so that local maintenance can be kept to a minimum. Third, use of bcoWeb is free (free of charge and free to copy, use and modify).

Semantically rich In great contrast with the majority of today's data, bcoWeb-based data is computer-readable. Of course, software vendors can continue to work with today's non-integrationable textual or binary data, but a semantically rich application foundation provides many more possibilities for innovative and value adding applications. The Semantics needed can be, if desired, contributed to bcoWeb, as the semantic needs of software vendors will be close to the needs of BC itself.

Web services Internet-based applications and documents are much more widely usable and accessible than today's applications. bcoWeb web services offer the opportunity to make the applications available on a larger scale than currently feasible. For

[1] This does not mean that the applications have to be open source too.

instance, small customers can now also be served: the kind of customer that wants to create just a single Specification per month. An entirely new market springs up for data-enriching, knowledge-providing applications that are barely in existence nowadays.

For all of this to come to pass, an ecosystem must be in place. A single software vendor that is the only bcoWeb user will not get added benefit out of it. Two communicating applications, however, already provide enough benefit, when compared to the extra cost and effort needed to cooperate using current means. It is the belief expressed in this thesis that bcoWeb will be able to bootstrap itself from a humble beginning. Interested software vendors are referred to [1].

9.5 What is the future role of the Specification?

Specifications translate designs into descriptions that include materials, components, work and constraints following from regulations. Specifications, part of the Tender Documents, are also legally binding contracts between Clients and Contractors. Therefore, one way or the other, Specifications will be with us for many decades to come, if not forever. But that does not mean that the traditional paper-based form will remain unchanged. The obvious electronic versions of 'dumb' paper-based Specifications are already used in the market. Their introduction did not result in much change, but what will happen if 'smart' electronic Specifications enter the scene?

The major drawback of Specifications is their rigidity. Clients have to define their requirements in Specifications, but requirements change once the project gets started. Clients see and learn unexpected new possibilities and the desire to change the Specification is quite common. In the traditional Specification-based process, changes in the Specification are costly, and even more or less prohibited. If new BC processes can be devised that increase the flexibility of the Specification, more satisfied Clients will result.

One way to fulfill that desire is to create dynamic Specifications. If controlled by agreed rules, computers can handle the consequences in responsibilities, costs and rewards.

A new-style Specification and its creator (a smart specification application) has a more central place than paper-based Specifications. A paper-based Specification derives its central place from the legal and contractual weight accorded to it. A bcoWeb Specification has, probably, the same legal and contractual weight, but is in addition a valuable source of well-accessible information.

The NG Internet provides the means for the Specification of the future to keep its position as a central document in BC, enabled by the much improved information structuring, information linking and knowledge providing possibilities of NG Internet. Such a semantically rich eSpecification both enables and needs juridical and social changes in BC.

9.6 How will Building-Construction look like in the future?

Predicting the future is difficult, especially in the overlap between BC and Information and Communication Technology (ICT). Too many 'failures' of the past prove this point. Literature studies on the subject of BC Semantics going back nearly thirty years show that the ideas about the goal of ICT in BC are largely unchanged. Much of the texts of papers written in the eighties can be recycled without shame. What, then, is it that fuels the enthusiasm expressed in this thesis and in ICT research groups that concentrate on the problems of the BC industry?

The answer, as explained in chapter 6, lies in the technical features of the NG Internet. The NG Internet (or semantic web, or web 2.0) not only suddenly makes it possible to share information and knowledge between everybody involved who has access to the Internet, but—and much more important—also makes the information accessible for computer applications.

Up until now Internet has been primarily used by humans. Web-

pages and email are directed at human readers. Computer applications are still simply not smart enough to understand what is said.

With the proposed solution concept, the role of computers will drastically change. In the future, computers will perform many tasks that are now done by human actors. Mostly of course concentrating on the information processing aspects. It is not that computers will become the Big Brothers as sometimes feared. On the contrary, computers will do what we humans do poorly, but with respect, not taking away our responsibilities and leaving us humans in charge.

Adopting the technology discussed in this thesis, BC can become much more efficient. All kinds of new players will enter the scene. Without human effort, computer applications can retrieve the data required and send their results to all parties involved. Computer applications can work together. Construction robots and intelligent equipment become feasible. Also a next generation of 'smart' documents will arrive. Smart documents will be able to perform tasks now requiring action by middle management, *i.e.* they know their goal and their Clients, and can be trusted to deliver the message on time, at the right place, in the right format.

9.7 Contributions of this research

This research made four main contributions to science and society.

1. The GARM was used as a basis for Ontologies. The FU/TS structuring mechanism received prominence over the customary subclass relation. This leads to a much more BC-centric style of Ontologies which are also much more dynamic. An added benefit is the more direct link to the money-making process by mimicking the demand-supply relation.

   The use of GARM also provides Ontologies with a context for the definitions. Definitions for a TS, in this way, are always in context of the enclosing FUs and vice versa.

2. A great emphasis on an *open source* bcoWeb as a requirement for success. This comprises both an open source license on the

bcoWeb content and an open source style of content development. Distributed, Internet-based development allows many interested parties to contribute, also small companies.

The open source license prevents vendor lock-in, a critical point as the aim is to structure the information of BC as a whole. The chance of BC, as a whole, to entrust all of their communication to one single vendor or organisation is very low.

The open source license, additionally, makes bcoWeb also an interesting and safe subsidy, funding and support target.

3. An emphasis on an explicitly *national* bcoWeb instead of the customary research for international information structuring standards. Perhaps EU research has a lot of influence in this regard. See the suggestions for further research in the next section, though.

4. To BC ICT research, a contribution[2] is the emphasis on the simple HTTP+XML+RDF style of web services instead of the SOAP style that is seen almost exclusively in BC ICT research. SOAP seems to be assumed as a given, when working with web services. The much simpler style (often called REST) severely reduces complexity: complexity which is the last thing BC needs, if NG Internet-based process innovation is to come about.

A possible big impact on society as a whole and BC in particular is bcoWeb's capability to reach the—until now—elusive goal of commonly usable and available BC Semantics. This is coupled with a vision for simple, cheap, available NG Internet-based information exchange.

The reason? The technological choices in this thesis were made with the goal in mind of *applicability in practice*. When trying to improve BC as a whole, the whole of BC must be targeted, not just the top 1% which have the big project databases. The choices must not fly in the face of economic and political acceptability. The oh-so-important software vendors must have a good reason to support the solution and the disincentives must be small or non-existent.

[2]First mention was in 2002 [2].

Hopefully, part of this mindset can also be seen as one of the contributions of this thesis.

BcoWeb opens a road to a new BC paradigm where man and machine communicate in the language of the industry.

9.8 Suggestions for further research

One single PhD research project is by nature restricted to what can be accomplished in four years time. An important result of such a project is a set of new questions that can be used as ideas for further research, as detailed research always unearths further questions.

One target for additional research is the knowledge component. This thesis showed an attractive approach for improving the data and information handling capacity of BC. Some suggestions have been given on how this approach also makes BC a more knowledge-driven industry. There are however many opportunities for much more detailed research in this area.

Though keeping a constant eye on the needs and uses of Specifications, there hasn't been much attention to the inner details of a Specification. Possible attention areas include the knowledge-intensive links to regulations and laws which are probably worth a couple of years of research. How to handle the differences between the generic properties attached to bcoWeb objects and the probably partly different properties as preferred by the Specification. This thesis showed some technical possibilities for multiple Classifications per Specification: what are the organisational, juridical and practical possibilities or difficulties to bring this to practice and to perhaps take it somewhat further by having different Specifications for different kinds of projects?

Additionally, a Specification can be seen as a model that describes the transformation from design to materialisation for a given project. The traditional (paper based or electronic) Specification will be a representation of this future model, just like the 3D geometry or 2D geometry is a representation of an IFC model.

ISO 12006-2 is partially Functional Unit (FU)-based [3]. There

was no time to investigate the possible connections with bcoWeb's GARM-like development. 12006-2 is quite formal and well-defined, so looking at both from a likewise well-defined OWL-viewpoint seems worthwhile.

Something that did not receive enough attention is the use of the subclass relation. Are catalog items subclasses or instances of bcoWeb TSs, for instance? For the prototype development, the subclass relation was used (which worked), but much more thought should go into this. Also the suitability of Ontologies for multiple viewpoints should receive more attention. Do you need multiple Ontologies or not?

In this thesis, bcoWeb was only filled with objects directly. There are, however, possibilities to mine catalogs and object trees automatically for data. Catalogs and object trees could have a local set of objects, which might be considered potential candidates for inclusion in the core bcoWeb. There is a lot of research to be done on this terrain. Perhaps revisiting neural networks and fuzzy logic is a worthy research target.

With an initial—and needed—focus on a *national* bcoWeb, there remains a desire for international cooperation, with many larger projects being international in nature and with many suppliers working internationally. Perhaps these international suppliers can be an important link in connecting various national bcoWebs. At the least, this is an area that can use further research.

Something that did not receive much attention in this thesis is the coupling between bcoWeb and geometry. Some work was done in eConstruct, demonstrating possibilities. This, too, is a worthwhile research target.

Web services themselves are also a potential research target. How can they be made acceptable? What are the limitations? Are certain areas more suited for web services than others? Can a strength calculation be reliably performed? What is the effect on financing and billing structures?

9.9 Recommendations

Open web of cooperating computer programs From this research effort it seems clear that increased matching of supply/offering and demand (*i.e.* a tighter integration of the supply chains), improved process control and increased flexibility of the processes all require increased information and knowledge sharing to such a level that only an open web of co-operating computer programs will be able to do the job. This open web of cooperating computer programs should be an implicit element of many research activities as it promises much better results than stand-alone developments.

BC itself must take action The bcoWeb experiment itself is no solution, it is merely a finger pointing in the right direction. The next steps have to be taken by the BC industry, their clients (government, large facility owners), the suppliers, the service providers, the software vendors and the knowledge institutions.

The recommendation to all parties involved is to raise the required funding and set up the organisation[3] to create a reasonable set of open BC Ontologies, each containing a few hundred objects, properties, units, relations and constraints. This effort should be large enough to create an initial bcoWeb that is rich enough for most current computer applications. As current applications are not yet really semantically rich, the required effort will be moderate, at least moderate considering what is at stake.

Initial software and pilot projects From that basis the software vendor community should be involved to provide an initial set

[3] 'The organisation' does not necessarily mean one single organisation. There is, however, development work to be done, web server space to be paid and content to be added. Manpower can be donated or volunteered, web server bills paid. Perhaps this is best handled through one channel, though open source developments have shown that looser forms are also capable of working. Material funding, manpower/data funding and financial funding are needed, though.

of bcoWeb compliant instruments.

At the same time, the demand and supply side should organise and execute a number of pilot projects providing feedback to their management and to the developers. At that time the funding required for the extensions of the bcoWeb should, at least partly, come from the market, because at that time it should be clear that improved information and knowledge sharing and process control indeed helps to reduce cost of failure and leads to increased end-user value. By that time, it should also be clear that an explicitly *open source* bcoWeb is a very attractive subsidy and funding target.

Integration in a larger effort As the transition of the BC industry involves more than information and knowledge sharing—it also involves cultural changes, legal aspects, financial and contractual aspects—the proposed development should be part of a larger effort, such as promoted by the Dutch *Regieraad* and *Proces- en SysteemInnovatie in de Bouw* (PSIB).

Bibliography

[1] Tim Bray. Technology prediction success matrix–12: 80/20 point, January 2004. Available on-line at `http://www.tbray.org/ongoing/When/200x/2004/01/14/TPSM-8020`.

[2] Reinout van Rees, Rèza Beheshti, and Frits Tolman. bcXML enabled VR project information front-ends. In Žiga Turk and Raimar Scherer, editors, *Conference proceedings - E-work and e-business in AEC*. ECPPM, Balkema, 2002.

[3] Anders Ekholm. A conceptual framework for classification of construction works. *Electronic journal of information technology in construction (itcon)*, 1(2), March 1996. Available on-line at `http://www.itcon.org/1996/2/`.

If you think the problem is bad now,
just wait until we've solved it.

Arthur Kasspe

Samenvatting

Gedurende vele jaren is vanuit de bouwinformatica onderzoek gedaan naar communicatietechnologiën die de bouw zouden kunnen helpen met het toepassen van computers en -netwerken. In 20 jaar is er vooruitgang geboekt, maar echte verbeteringen in snelheid en consistentie zijn uitgebleven. De hoofdreden is dat onze computerapplicaties niet in staat zijn om de taal van de bouw te begrijpen. Mensen moeten tekeningen lezen, er informatie uit destilleren voor hun applicaties, die applicaties hun werk laten doen en de uitkomst van de computer vervolgens weer terugvertalen in voor mensen bruikbare informatie en dat verder doorgeven. Dit soort transformaties voegt geen waarde toe. In tegendeel, ze zijn een bron van fouten en verwarring.

Met de komst van XML/RDF en het semantische web is er een nieuwe mogelijkheid gekomen om de communicatie tussen mens en computer en computers onderling maatgevend te verbeteren. Dit zou potentieel de missende technologie kunnen zijn die de bouw al zo lang nodig heeft. Wat als we met dezelfde terminologie die we onderling gebruiken met onze applicaties zouden kunnen communiceren? Zou de ICT communicatie tot dat niveau omhoog getild kunnen worden? Zou dat betekenen dat de mens, waar onnodig, uit het informatieproces gehaald zou kunnen worden? En dat de mens niet meer als vertaler hoeft te fungeren tussen de projectcommunicatie en die van de computerapplicaties? Zou dat een vermindering van communicatiefouten en faalkosten betekenen? Zou dat het proces versnellen? Wat zouden de consequenties zijn? Hoe zou deze nieuwe technologie geïmplementeerd moeten worden?

Het participeren van de computer in onze menselijke technische communicatie, zou dat de ontbrekende technologie kunnen zijn?

Het onderzoek is verricht aan de Technische Universiteit Delft en bij de stichting STABU, het Nederlandse besteksinstituut voor de bouw, dus bouwinformatica in theorie en praktijk. De eerste twee jaar vonden plaats binnen het EU project "eConstruct" wat zorgde voor een goede inbedding in het internationale bouwinformatica onderzoeksgebied.

De huidige eerste generatie ICT toepassingen concentreren zich voornamenlijk op één taak en niet op project-gerelateerde taken—waarvoor informatie- en kennisuitwisseling nodig is. Classificatie- en besteksystemen zijn wijdverbreid in de markt maar ze bieden te weinig informatierijkdom om informatieuitwisseling in de bouw adequaat te ondersteunen. En PDT-oplossingen zijn òf te gecentraliseerd òf effectief alleen beschikbaar voor de bovenste 5% van de bouwmarkt.

Het voor u liggende onderzoek heeft zich vooral gericht op het gebruik van innovatieve Internettechnologiën op het snijpunt van twee gebieden: (a) de bestaande bouwinformatica, vooral productmodellering en informatieuitwisseling en (b) de gewenste bouwprocesinnovatie in de huidige bouwpraktijk.

Internet heeft een ongekend aantal nieuwe mogelijkheden geschapen. Welke van deze mogelijkheden kan aan het begin staan van bouwprocesinnovatie? De vraag kwam ook op hoe dit soort mogelijkheden dan geïmplementeerd en gebruikt zou moeten worden.

De onderzoeksvragen werden na de eerste analysehoofdstukken als volgt opgeschreven. Hoe kan moderne webtechnologie helpen om nieuwe, dynamische bouwprocessen te realiseren die adequaat voorzien in de klantbehoefte (waar voor hun geld) en tegelijkertijd in die van de bouw (geld voor hun geleverde waarde). Hoe het nieuwe Internet te gebruiken om beide partijen (en hun computerapplicaties) te voorzien van precieze en semantisch rijke on-line definities van bouwobjecten? Hoe kunnen softwareleveranciers voordeel trekken uit zo'n ontwikkeling? Wat is de mogelijke toekomstige rol van het traditionele bestek in zo'n nieuw innovatief bouwproces?

Volgens de conceptoplossing kan het vermogen van de bouw om waarde toe te voegen aanzienlijk vergroot worden door een open en vrij toegankelijk "Building-Construction Ontology Web (bcoWeb)". Dat houdt in: (a) Het stimuleren van het ontwikkelen van vakgebied-

specifieke bouwontologiën met definities van bouwtermen, gestructureerd volgens het GARM-principe. BcoWeb's sterkste punt is de ingebouwde aandacht voor de interactiepunten in de bouw: daar waar vraag en aanbod elkaar raken. (b) Het in onderling verband plaatsen van de verschillende bouwontologiën in een nationaal en open source bcoWeb. Eerdere standaardisatiepogingen hebben tevergeefs geprobeerd om internationale standaarden voor informatieuitwisseling op te bouwen. Een van de conclusies die uit het Europese eConstruct project gretrokken mag worden is dat de bouw land-specifiek is (binnendeuren zijn in elk land verschillend), dus de ontwikkeling van internationale standaarden kan pas beginnen zodra nationale standaarden beschikbaar zijn gekomen. (c) Het stimuleren van het ontwikkelen van met elkaar samenwerkende *web services* die de bouwterminologie "begrijpen" en naar wiens informatie verwezen kan worden. Hierdoor kan de communicatie tussen alle betrokken partijen—inclusief het algemene publiek—ondersteund worden. Hierbij worden "slimme" computerapplicaties en "slimme" documenten ook als betrokken partij gezien.

BcoWeb heeft vele voordelen. Het fungeert als een collectie definities waarnaar verwezen kan worden door zowel mensen als computers. Het is open source, waardoor het veilig is om op te bouwen. Het heeft rijke semantiek, biedt context-afhankelijke definities en ondersteunt meerdere representaties, dus zijn er veel mogelijkheden voor innovatieve en waarde-toevoegende computerapplicaties, onder andere door ze als *web service* beschikbaar te maken.

Gedurende de implementatiefase is er een web-gebaseerde bcoWeb *editor* gemaakt waarmee bcoWeb deels is gevuld. Verder zijn er catalogus- en objectboomapplicaties gemaakt alsmede een eenvoudige besteksgenerator. De *cases* lieten zien dat zo'n bcoWeb-benadering zorgt voor de flexibele, algemeen beschikbare bouwtaal die nodig is voor de ontwikkeling van "slimme" computerapplicaties die zich kunnen gedragen alsof ze de taal van de bouw daadwerkelijk begrijpen.

BcoWeb (plus bijbehorende applicaties) heeft een aantal voordelen t.o.v. eerdere vooral papier-gebaseerde technieken. Computerapplicaties kunnen het bcoWeb gebruiken om op een gemeenschappelijke manier naar een project te kijken. Tekstuele documenten kun-

nen “slimme” applicaties worden als hun onderliggende data via het Internet beschikbaar wordt gesteld. Zo kunnen ze, net als mensen, in samenwerking met anderen (computer of mens) taken vervullen alsof ze de taal van de bouw begrijpen. Zonder verdere inmenging (maar niet zonder controle) kunnen ze vele informatie- en kennisbeheertaken vervullen: een aantrekkelijke nieuwe technologie die het bestaande bouwproces aanzienlijk kan verbeteren. Het primaire informatieverwerkingsproces kan, net zoals in de automobielindustrie, het beste aan het informatiesysteem overgelaten worden aangezien dat het aantal fouten vermindert.

Een bijdrage van dit onderzoek: het gebruik van GARM zorgt voor ontologiën die veel meer een bouwkarakter hebben en die ook direct op het economische vlak in de bouw inhaken omdat ze het vraag/aanbod proces simuleren. Een tweede bijdrage is de nadruk op de noodzaak van een open source licentie voor de ontologiën die als basis voor de volledige informatieuitwisseling in de bouw gebruikt zullen worden: daar is geen plaats voor beperkingen door individuele bedrijven of projecten. Ten derde, het gebruik van GARM maakt het mogelijk om voor verschillende bouwsectoren andere definities en andere gezichtspunten aan te houden. Definities kunnen vrijwillig gebruikt worden.

Een belangrijke bijdrage is ook de nadruk die op de praktische toepasbaarheid voor een gemiddeld bouwbedrijf wordt gelegd. Veel onderzoek richt zich op de bovenste 1% van de markt die zich de grote projectdatabases en de duurste CAD paketten kan veroorloven.

Bestekken zullen hun centrale rol in het bouwproces uitbouwen als het bcoWeb gebruikt wordt om van het bestek een waardevolle bron van goed-geordende en goed-toegankelijke informatie te maken.

Wat is de aanbeveling? Wat moet er gedaan worden? De bouw zou nationaal moeten beginnen met het vullen van bcoWeb bouwontologiën. Het begin mag klein zijn, gewoon een set ontologiën die tezamen voldoende informatierijkdom bieden om communicatie mogelijk te maken.

Een belangrijke rol is weggelegd voor de software- en informatieleveranciers. Zonder eindgebruikerssoftware is bcoWeb alleen maar een voorstel op papier.

Acronyms and definitions

AEC Architecture, Engineering and Construction. A common acronym for the entire industry. Can be considered a synonym to BC.

ArchGenXML Open source code generator. Generates Plone products from UML diagrams. Instrumental in creating some of the applications that demonstrate the concepts of this thesis. Since 2005 the author is one of the core developers. `http://plone.org/products/archgenxml`

BC Building-Construction. The Building and Construction industry. A common acronym for the entire industry. Can be considered a synonym to AEC.

BCCM Building-Construction Core Model. Development in the 1990s for an ISO-STEP standard for Building-Construction.

bcoWeb Building-Construction Ontology Web. Initiative to start a co-operative open source Ontology web for Building-Construction, based on a simple GARM-like model.

bcXML Building-Construction XML. eConstruct-developed format for Building-Construction data exchange, based on XML.

BIM Building Information Model. Vision of a central source for all building information, mostly associated with IFC-like developments.

CAD Computer Aided Design.

CAE Computer Aided Engineering.

CAM Computer Aided Manufacturing.

CASE Computer Aided Software Engineering.

CBC Co-ordinated Building Communication. Danish classification system.

CC Creative Commons. Collection of licenses for creative works that are made available gratis, with some possible standard restrictions, for general use. Note that this thesis is placed under a CC license, see page 233.

CIB International council for research and innovation in building and construction. Organisation to stimulate and facilitate international cooperation and information exchange in the building and construction sector. Formerly named *Conseil International du Bâtiment*.

Classification A grouping of entities according to some external criteria. The grouping is quite natural, as it is mostly made from a specific viewpoint. Classification is basically a set of boxes (with labels) to sort things into. It can be used as a user-friendly view on/in a Taxonomy or Ontology.

Client The party for who the Contractor carries out the work. *Opdrachtgever*.

CMS Content Management System. A web server that provides the ability to manage, secure, edit and publish the content of a website through the web.

Coding The process of classification of information.

Contractor The party who carries out the work as described in the Tender Documents for the Client. *Aannemer*.

CROW *Centrum voor Regelgeving en Onderzoek in de Grond-, Water- en Wegenbouw en de Verkeerstechniek*. Creator of the Dutch specification system for road and waterworks.

DAML DARPA Agent Mark-up Language. Ontology language build on RDF, extending it. Originally funded by DARPA (a USA defense research organisation that also effectively started the Internet, which was originally called ARPA-net). Merged into OWL together with OIL.

DIS Draft International Standard. The step in ISO's standardisation before becoming a formal international standard.

DL Description Logics. OWL DL is an OWL variant, described in section 4.5.2 on page 78.

DTD Document Type Definition. Earliest XML format definition.

eConstruct eConstruct is an EU research project that ran from January 2000 till January 2002. The author participated full-time in this project.

EPISTLE European Process Industries STEP Technical Liaison Executive

ERDL EPISTLE Reference Data Library.

ETIM ElectroTechnical and Installation Model.

EU European Union.

FDL GNU Free Documentation License. License that makes content available gratis and that allows modifications of the content, under the restriction that the modified content is made available under the same license. Used by, amongst others, the wikipedia project.

FTP File Transfer Protocol. Common way of uploading and downloading files from so-called FTP servers. It is a bit less common nowadays as HTTP has become more popular.

FU Functional Unit. One of the two core classes of the GARM.

GARM General AEC Reference Model. Influential reference model developed in the late 1980s.

GIS Geographical Information System. Digital maps (in short).

GNU GNU 's Not Unix. Project started by the Free Software Foundation (headed by Richard Stallman) to create a free unix system (free as in 'gratis' and free as in 'freedom to share and change'). GNU and the Linux kernel are the core of the current Linux systems.

GPL GNU General Public License. The most widely used free software license.

GUI Graphical User Interface. The nowadays-common user interface for computer

	programs. The opposite of a textual command-line user interface ('DOS screen').
GUID	Globally Unique ID. Identificator for an item that is guaranteed to be unique everywhere.
HTML	HyperText Mark-up Language. Simple language format for webpages.
HTTP	HyperText Transfer Protocol. The protocol used for communication on the web ('getting HTML pages'). Also the basis for web services, as you can likewise transport XML, RDF, etcetera.
IAI	International Alliance for Interoperability. Organisation that develops the IFC standard.
ICT	Information and Communication Technology. Generic term for computer- and communication-related technologies.
ID	IDentificator. Name or number used to identify something.
IFC	Industry Foundation Classes. IAI standard for building information models and model-based CAD exchange.
ifcXML	XML format for IFC data.
IFD	International Framework for Dictionaries. Effort to harmonise multiple Taxonomies/Ontologies, mostly for use by IFC.
Internet	Internet is, technically, the all-encompassing term for the TCP/IP protocol, FTP, HTTP, email, etcetera.
IS	Information System.

ISO International Standardisation Organisation. Largest standardisation institute.

NG Internet Next Generation Internet. The next generation Internet, formed by improved connectivity, semantic web technologies and web services.

OIL Ontology Inference Language. Ontology development from Europe. Now, with DAML, merged into OWL.

Ontology A set of well-defined concepts describing a specific domain. The concepts are defined using a subclass hierarchy, by assigning and defining properties and by defining relationships between the concepts. An Ontology's goal is to provide a common, referencable set of concepts for use in communication. It is quite common to use multiple Ontologies, each providing concepts for a particular domain, together forming a rich vocabulary for communication.

OWL Ontology Web Language. Standard Ontology format for the Internet. Best-practice merger of DAML and OIL. Uses RDF.

PDT Product Data Technology.

POSC-CAESAR Petrotechnical Open Standards Consortium - Caesar offshore program

Plone Open source CMS on top of the Zope application server.

PSIB *Proces- en SysteemInnovatie in de Bouw*. Dutch initiative at improving BC's value-adding and innovative capacity by

	innovating both the BC process and the BC system as a whole.
Python	Dynamic, object orientated, friendly language.
RAD	Rapid Application Development. Sophisticated tools that allow rapid application programming by providing 'wizards', code generation and, most-times, graphical design of user interfaces.
RDF	Resource Description Framework. A (most often XML formatted) standard way of describing subject, property, object triples, including basic terminology like specialisation.
REST	REpresentational State Transfer. Web services style based on plain HTTP and XML (or RDF). This is the 'opposite' of SOAP.
RNG	Relax NG. Format for XML format definitions, simpler and more powerful than XSD.
Semantics	The meaning of things. An example is probably the best way to explain this. If a computer file refers to something as 'door', what does that mean? What is the Semantics of 'Door'? Likewise, the semantic web tries to make data exchange more *meaningful* by providing mechanisms to make clear what is meant by certain terms.
SfB	*Samarbetskommittén för Byggnadsfrågor*. Swedish committee that created the SfB classification.
SGML	Standard Generalized Markup Language. More elaborate precursor of XML.
SMEs	Small and Medium Enterprises. Dutch: *midden- en kleinbedrijf (MKB)*.

SOAP Simple Object Access Protocol. Part of a large family of web services standards. SOAP is a way to encode web services requests inside XML files. This way, the actual web services are 'tunneled' over HTTP using XML. This is the 'opposite' of the REST style of web services.

Specification Textual document containing the technical and administrative specifications. Part of the Tender Documents. *Bestek*.

STABU Dutch specification institute for the building industry.

STEP STandard for the Exchange of Product data.

SVG Structured Vector Graphics. XML format for two-dimensional vector graphics.

Taxonomy A hierarchical grouping of entities according to data internal to the Taxonomy. 'Classification Taxonomy' or 'simple Ontology'. When used as a simple Ontology, the Taxonomy's hierarchy should be based upon a subclass hierarchy.

TCP/IP Transport Control Protocol/Internet Protocol. Protocols for layer 3 and 4 of the common 7-layer protocol stack. Effectively this can be considered to be the Internet.

Tender Documents The set of documents which the Client issues to a possible Contractor. *Bestek, aanbiedingsdocumenten*. Includes the Specification, the specification drawings and the generic administrative regulations.

TS Technical Solution. One of the two core classes of the GARM.

UML Unified Modeling Language. Diagram format for (mostly) object oriented design and programming.

UNETO/VNI *Ondernemersorganisatie installatiebranche en de technische detailhandel*. Dutch organisation for the electro-technical and installation sector.

URI Uniform Resource Identifier. As URL, but with a few added identification schemes like 'urn:'.

URL Uniform Resource Locator. Examples: `http://vanrees.org/` and `mailto:reinout@vanrees.org`. Uniform way of locating resources on the web. In this document they are used as equivalent to URIs.

VEH *Vereniging Eigen Huis*. Dutch house owners' association.

Virtual Enterprise A consortium of companies working together in a temporary construct. In BC, it is normally a number of Contractors working together, but functioning as one virtual Contractor from the Client's viewpoint.

VRML Virtual Reality Mark-up Language. Format for three-dimensional graphics. It has an XML-like format, a later version called X3D is 100% XML compatible.

W3C World Wide Web Consortium. The standards body governing most of the web's standards.

XML eXtensible Mark-up Language. Universal data format for the Internet. It's a standard way for storing information.

XSD	XML Schema Definition. W3C-developed XML-based format for XML format definitions.
XSLT	eXtensible Stylesheet Language for Transformations. XML format for transforming XML.
ZODB	Zope Object DataBase. The Python object database included with Zope. One of the few object databases seeing real use.
Zope	Web application server build in Python.

Index

He's not a fiery orator that yokes your ankles to a chariot of rhetoric and drags you around the arena three times.

Author index

Curriculum Vitae

Full name: Reinout van Rees
Born: 25 December 1972, De Bilt, the Netherlands
Email: `reinout@vanrees.org`
Website: `http://vanrees.org/`

Education

- MSc study civil engineering, September 1991–Augustus 2000 at Delft University of Technology. Specialisations: traffic engineering; planning and organisation; building informatics. Graduation thesis: *ceXML - an XML vocabulary for building and civil engineering.*
- *VWO* (high school, preparatory university education), September 1985–Augustus 1991. *College Blaucapel* in Utrecht (first three years) and *Overvoorde* in The Hague (last three years)

Work experience

- Programmer at Zest Software (`http://zestsoftware.nl`), April 2005–present.
- PhD student at the faculty of Civil Engineering and Geosciences at Delft University of Technology, October 2000–January 2007.
- Research assistant at STABU Foundation, October 2000–September 2004.

Liberty has become a crime punishable by death

Blake's 7

License

[4]Except several images that have other copyright holders, as indicated.

I'd sooner lower my [editor's note: body parts redacted for family IT publication] into a vat of boiling acid than even use a Windows computer again – let alone own one.

Colophon

This manuscript was typeset by the author with the LaTeX 2_ε Documentation System on a PC running Debian GNU/Linux ('woody') and another PC running Mandrake Linux (10.0). The last 18 months, all editing was done on an apple powerbook running OS X. The server that hosted the version control system (subversion) was running on Debian GNU/Linux ('sarge').

Text editing was done partly in *GNU Emacs* using the AUCTeX package and partly in *BackTalkBook* under *Zope*, running on aforementioned server. A lot of tasks were automated with *Python* and the Makefile-like *AAP*. The illustrations and graphs were created with *Dia* for diagrams and *Gimp* for screenshots and other images. The UML tools used were *Object Domain* and *Poseidon*. The cover layout was produced with Apple's *Pages* using images taken with a *Canon A530* during a 2006 holiday in the Vosges mountains.

The body type is 12 point Computer Modern Roman. Chapter and section titles are in various sizes of, also, Computer Modern Roman. The monospace typeface used for program code is Computer Modern Typewriter.

The final output was converted to a PDF file, which was then printed by Sieca Repro, Delft, on 80 grams offset white. The cover is 250 grams colotech.

The dissertation is also available on-line at `http://vanrees.org/research/phd`.